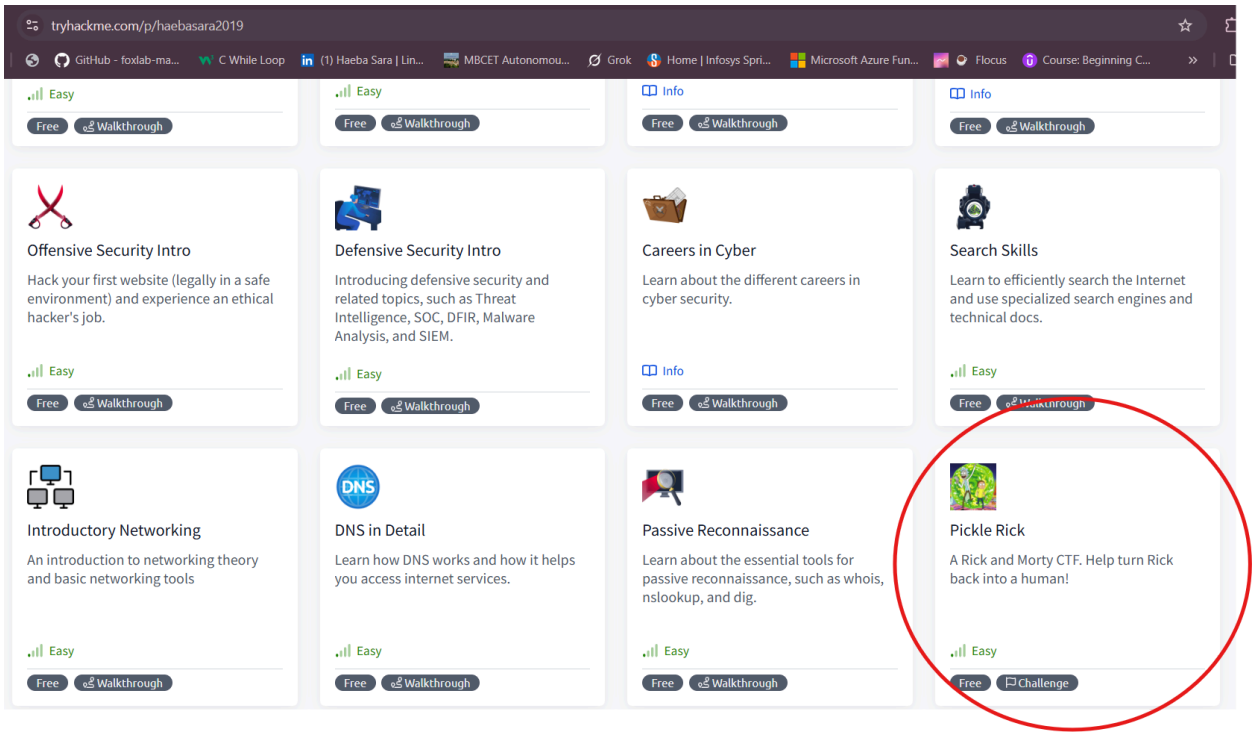
Cyber Security (OWASP Kerala x Mulearn)

Task 1: TryHackme CTF

By: Haeba Sara Alexandar
ECE, Mar Baselios College of Engineering and Technology

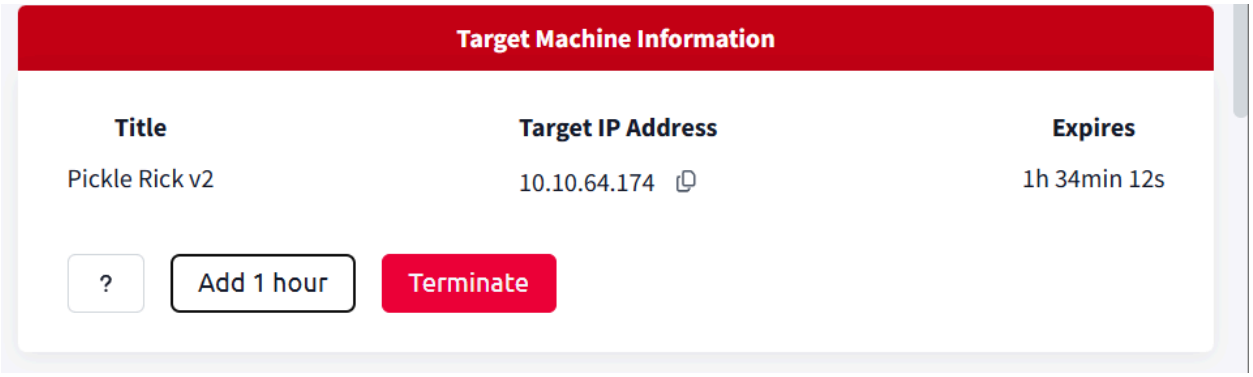
As a part of the Task 1 I have completed the Pickle Rick CTF in TryHackme

Attaching the screenshots of the same

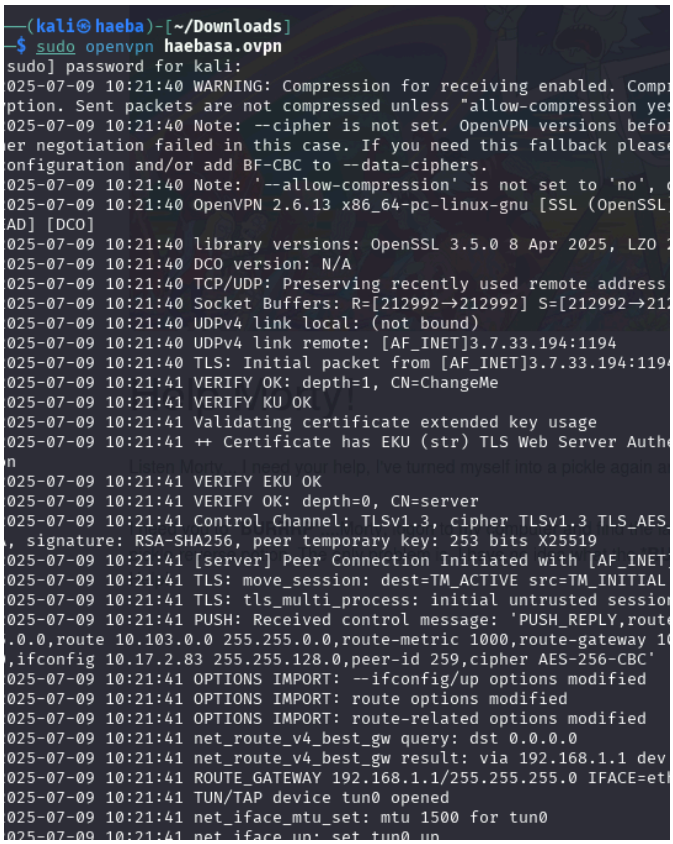


Step 1: Target Machine Deployment Overview

The screenshot shows the active target machine session for the challenge titled **"Pickle Rick v2"**. The machine has been deployed with the IP address **10.10.64.174**



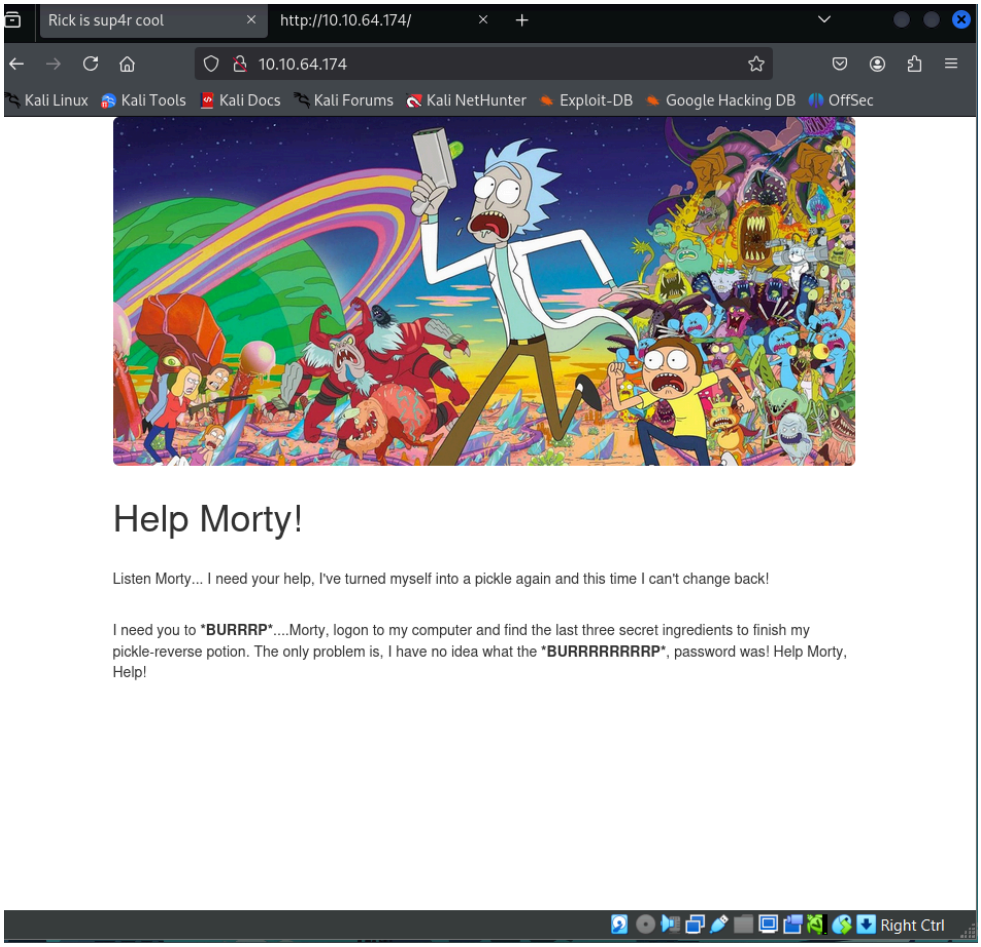
Now I established a connection in the ip above using open vpn in Kali Linux (VM)



Step 2: Initial Reconnaissance and Web Interface Inspection

As part of the initial reconnaissance phase, I performed a ping test on the *target IP address* `10.10.64.174` to verify network connectivity and confirm the host was responsive. Once I confirmed that the machine was online, I opened the IP address in a web browser. This loaded the main page of the website (as shown in the screenshot below).

Since there were no clear hints or links on the page itself, I right-clicked and viewed the page source code to look for any hidden information. From the source code, I was able to find the *username*: `R1ckRul3s`, which I planned to use for the login page discovered earlier.



```
  </div>

  <!--

    Note to self, remember username!

    Username: R1ckRul3s

  -->

</body>
</html>
```

Step 3: Web Directory Enumeration and Discovery

The screenshot below shows directory enumeration performed on the target machine (`10.10.64.174`) using the tool Gobuster v3.6 in a Kali Linux environment. The command used specifies:

- Target URL: `http://10.10.64.174/`
- Wordlist: `/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt`
- Extensions scanned: `.php, .txt, .js, .py, .html`

- Threads: 10
- Timeout: 10 seconds

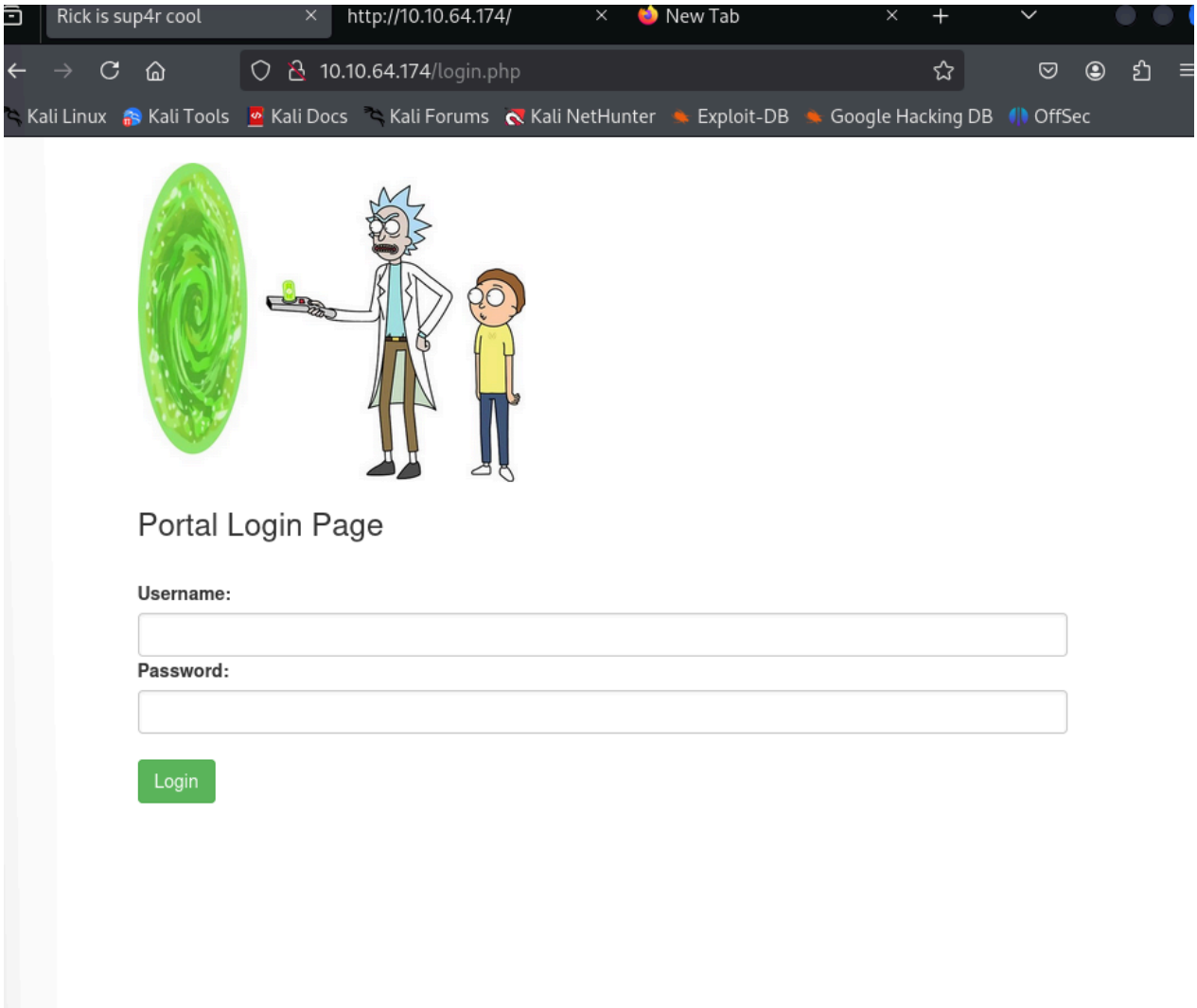
The scan revealed the following paths:

- `/index.html` (200 OK)
- `/login.php` (200 OK)
- `/portal.php` (200 OK)
- `/robots.txt` (200 OK)
- `/assets` (301 Moved Permanently)

Also, access to `.php` and `.html` at the root was forbidden (403). This enumeration step is crucial to identify accessible resources, hidden directories, and potential entry points on the web server for further exploitation.

```
┌──(kali㉿haeba)-[~]
└─$ sudo gobuster dir -u http://10.10.64.174/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x ph
p,txt,js,py,html
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.64.174/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php,txt,js,py,html
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.php                 (Status: 403) [Size: 277]
/.html                (Status: 403) [Size: 277]
/index.html           (Status: 200) [Size: 1062]
/login.php            (Status: 200) [Size: 882]
/assets               (Status: 301) [Size: 313] [--> http://10.10.64.174/assets/]
/portal.php           (Status: 302) [Size: 0] [--> /login.php]
/robots.txt           (Status: 200) [Size: 17]
```

Step 4: Login Page Access and Initial Enumeration



After identifying `/login.php` and `/portal.php` during directory enumeration with Gobuster, I accessed the **login interface** of the target web application hosted on `http://10.10.64.174`. Both `login.php` and `portal.php` led to the **same login page**, suggesting redundancy or aliasing of routes within the application's routing logic.

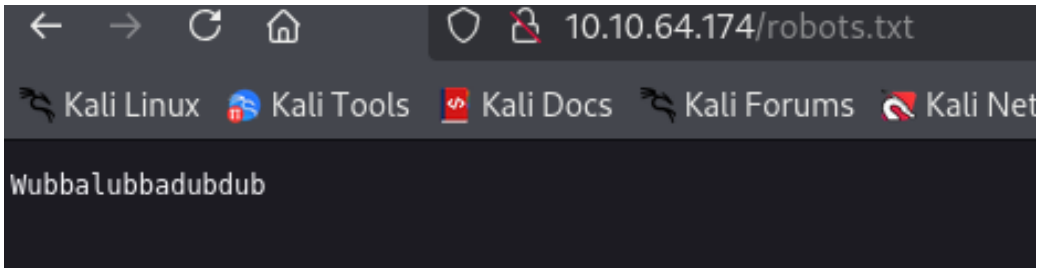
At this stage, I had access to only the **username** required for authentication. To enumerate further and potentially discover **credentials or sensitive files**, I manually inspected each discovered path.

- The `index.html` file revealed no useful information—it appeared to be either a placeholder or static content with no relevant indicators.

- I then navigated to `/robots.txt`, a file typically used to disallow web crawlers from accessing certain paths. This is a commonly overlooked source of sensitive or hidden directory references.

The contents of `robots.txt` appeared to disclose potentially restricted or sensitive endpoints, which could aid in further **privilege escalation** or **information disclosure**.

(Screenshot and analysis of `robots.txt` content follows below.)



Upon inspecting the contents of the **`robots.txt`** file, I identified a string that resembled a potential password. Based on its format and context, I hypothesized that it could be used as the login credential in combination with the previously discovered username (`R1ckRul3s`). I proceeded to test this string on the login page—and successfully authenticated, confirming its use as the valid user password.

Step 5: Accessing the Command Panel and Finding Key Files

After logging into the application, I was directed to a section labeled "Command Panel" under the “Commands” tab. This panel allowed me to enter commands—similar to how we would on a terminal—and the server responded with output.

I entered the `ls` command to list the files in the current directory. The following files and folders were displayed:

- `Sup3rS3cretPickl3Ingred.txt`
- `assets/`
- `clue.txt`
- `denied.php`
- `index.html`
- `login.php`
- `portal.php`
- `robots.txt`

This behavior suggests that the page might be running system commands in the background, which could indicate a command injection vulnerability.

Among the results, the file named `Sup3rS3cretPickl3Ingred.txt` stood out, as it seemed likely to contain the answer to the first question in the challenge.

Rick Portal | Commands | Potions | Creatures | Potions

Command Panel

```
ls
```

Execute

```
Sup3rS3cretPickl3Ingred.txt
assets
clue.txt
denied.php
index.html
login.php
portal.php
robots.txt
```

Rick Portal | Commands | Potions | Creatures | Potions

Command Panel

```
less Sup3rS3cretPickl3Ingred.txt
```

Execute

To confirm, I used the same command panel to open the file `Sup3rS3cretPickl3Ingred.txt`. Upon executing the command to read its contents, it displayed the text:

"Mr. Meeseek Hair"

This matched the format and expectation for the **answer to the first question** in the challenge.

Step 6: Retrieving the Second Ingredient from Target File

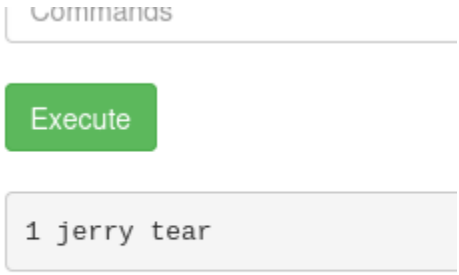
In the Command Panel of the Rick Portal interface, I entered the command:

```
less /home/rick/"second ingredients"
```

This command was used to open a file named `"second ingredients"` located in the `/home/rick/` directory. Upon execution, the content displayed was:

"1 jerry tear"

This confirmed the second required item for the CTF challenge (answer to the second question).



Step 7: Finding the Final Ingredient (Third Answer)

In the first image, I used the command:

```
sudo ls /root
```

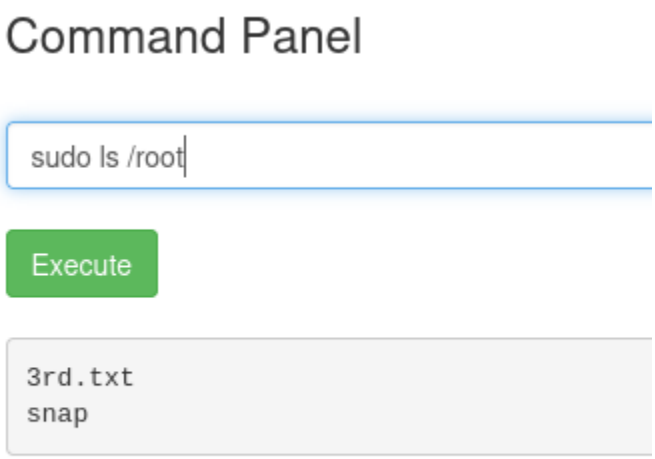
This allowed me to list the contents of the **/root** directory, which is usually restricted. Since the command worked, it showed two entries: `3rd.txt` and `snap`. Based on the naming convention,

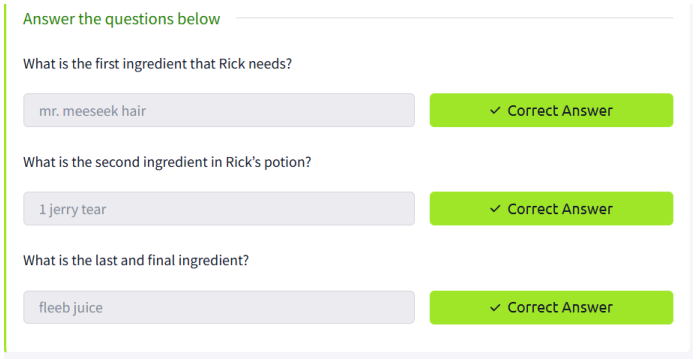
`3rd.txt` appeared to contain the **final ingredient** needed for the CTF challenge.

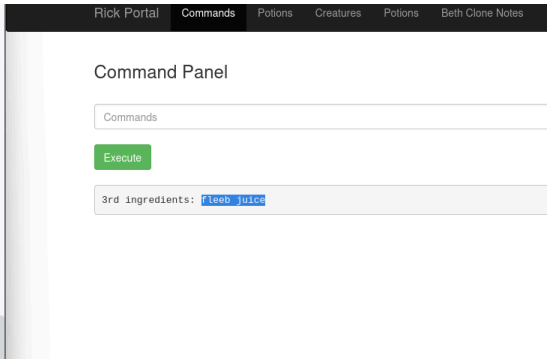
I then opened the file using the command panel, and the content revealed was:

"fleeb juice"

This confirmed the **third and final ingredient** Rick needed.







Summary

The last image shows the successful submission of all three required ingredients for the challenge:

1. First Ingredient – `mr. meeseek hair`

2. Second Ingredient – `1 jerry tear`

3. Third Ingredient – `fleeb juice`

Each entry was marked as Correct Answer, confirming the completion of the CTF objectives