Internship Final Report

Student Name: Emmanuel Ateji
University: Sheffield Hallam University (Alumni)
Major: Cyber Security
Internship Duration: April 10th, 2025 - May 5th, 2025
Company: Hack Secure
Domain: Cyber Security
Mentor: Mr.Nishant Prajapati
Assistant Mentor: Mr. Aman Pandey
Coordinator: Mr. Shivam Kapoor

Objectives

My primary objectives for this internship were to:

1. Develop a deep understanding of cybersecurity principles and practices.
2. Gain hands-on experience in identifying, analyzing, and mitigating security threats.
3. Enhance my skills in using cybersecurity tools and techniques in real-world scenarios.

Tasks and Responsibilities

During my internship, I was involved in the following key tasks:

- **Vulnerability Assessment:** Conducted a thorough scan of a target website to identify open ports and potential vulnerabilities.
- **Penetration Testing:** Performed brute-force attacks and directory enumeration on the website to uncover hidden directories and files.
- **Traffic Analysis:** Intercepted network traffic using Wireshark during a simulated login attempt, successfully capturing and analyzing transmitted credentials.
- **Decryption and Cryptanalysis:** Decrypted password-protected files using cryptographic tools and analyzed encoded hash values to recover plaintext passwords.
- **Reverse Engineering:** Used PE Explorer to analyze an executable file, identifying the entry point and other critical information.
- **Network Security:** Executed a de-authentication attack on a controlled network environment, capturing the handshake and subsequently cracking the Wi-Fi password using a custom wordlist.
- **Payload Creation:** Developed and deployed a Metasploit payload to establish a reverse shell connection on a virtual machine.

Intermediate Task

1. **Find all the ports that are open on the website** http://testphp.vulnweb.com/

   Command used: *nmap -sV -A testphp.vulnweb.com*

   - **sV** → Service and version detection
   - **A** → Aggressive scan (OS detection + script scanning + traceroute)

   Result:

   - Open Port Found: 80/tcp (http - nginx 1.19.0)
   - Closed Ports: 25/tcp (smtp), 139/tcp (netbios-ssn)
   - IP Address Resolved: 44.228.249.3
   - HTTP Title: Home of Acunetix Art
   - OS Guess: Oracle VirtualBox (94%), Slirp (94%)
   - Network Distance: 1 hop
   - Scan Duration: 44.41 seconds

Fig 1: Nmap Port Scan

2. **Brute force the website http://testphp.vulnweb.com/ and find the directories that are present in the website.**

   Command used: *gobuster dir -u http://testphp.vulnweb.com/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt*

   - **OWASP DirBuster 1.0-RC1 tool was used**
   - **Target: http://testphp.vulnweb.com:80/**
   - **Results show 27 directories and 25 files discovered**

   Inference:

   - The scan revealed multiple PHP files including:

     /login.php - Authentication endpoint
     /userinfo.php - Potential user data exposure
     /admin/ - Administrative panel (highlighted in blue, likely sensitive)
     /signup.php - User registration functionality

   - Several application components were found:

     /AJAX/ directory - Dynamic content functionality
     /php/ and /Flash/ directories - Different technology stacks
     /Mod_Rewrite_Shop/ - URL rewriting for an e-commerce component

   - The application appears to be an e-commerce site with user accounts, login functionality, and admin access

   - Multiple potential attack vectors are visible, particularly the admin directory and user management files

   These reconnaissance steps have uncovered the application structure, providing potential entry points for further security testing like authentication bypass, injection attacks, or privilege escalation.

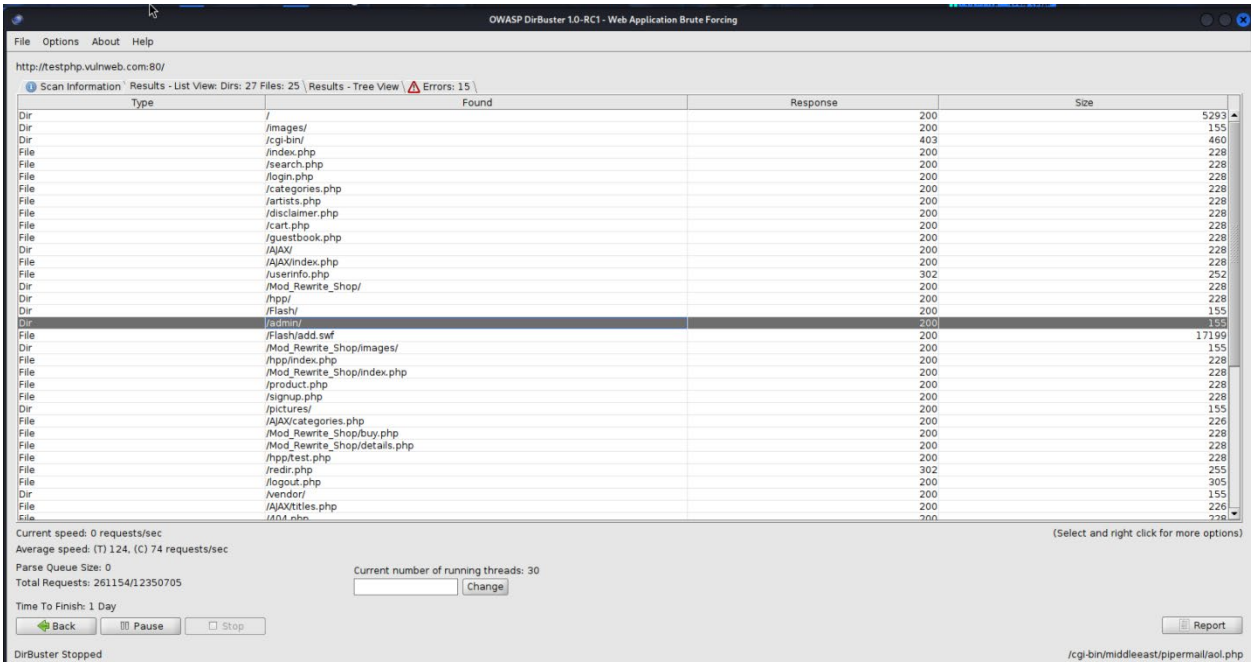


Fig 2: Brute force Directory

3. **Make a login in the website http://testphp.vulnweb.com/ and intercept the network traffic using Wireshark and find the credentials that were transferred through the network.**

Steps Performed:

- Open Wireshark and start capturing packets on the correct network interface i.e. , eth0
- Apply a capture filter for HTTP traffic:
- http
- Visit http://testphp.vulnweb.com/login.php in the browser.
- Attempt login using dummy credentials:
- Username: test
  Password: test
- Stop the Wireshark capture after the login attempt.

Observations:

- Traffic to a potentially vulnerable test website (testphp.vulnweb.com), which is commonly used for security testing and penetration testing exercises.
- A clear security issue - it shows unencrypted user credentials being transmitted in plain text, including username "Yomi" and password "kaliyomi25" as part of a sign-up form submission.
- The HTTP traffic captured in both images uses HTTP/1.1 without encryption (not HTTPS), making all communications vulnerable to interception.

- Multiple GET requests for various resources including "/categories.php", "/artists.php", "/cart.php", and "/login.php", suggesting this is a web application with e-commerce functionality.
- The server is running on nginx/1.19.0 with PHP 5.6.40 , which are older versions that may contain known vulnerabilities, especially the PHP version which is end-of-life.

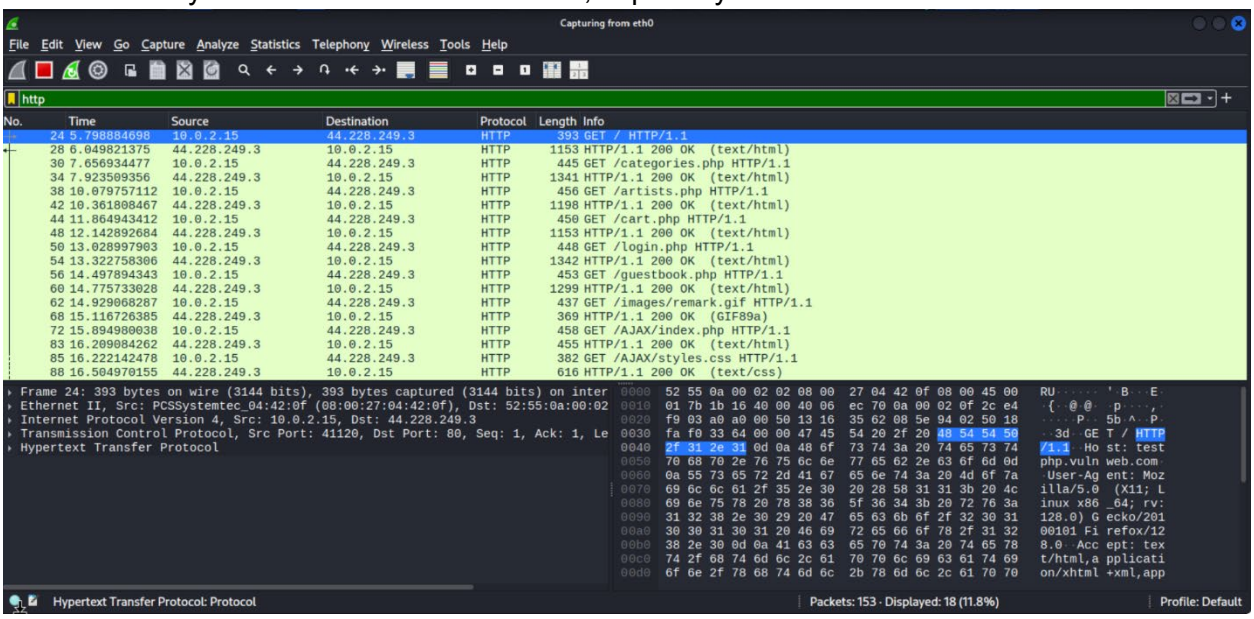


Fig 3.1: Packet Capture

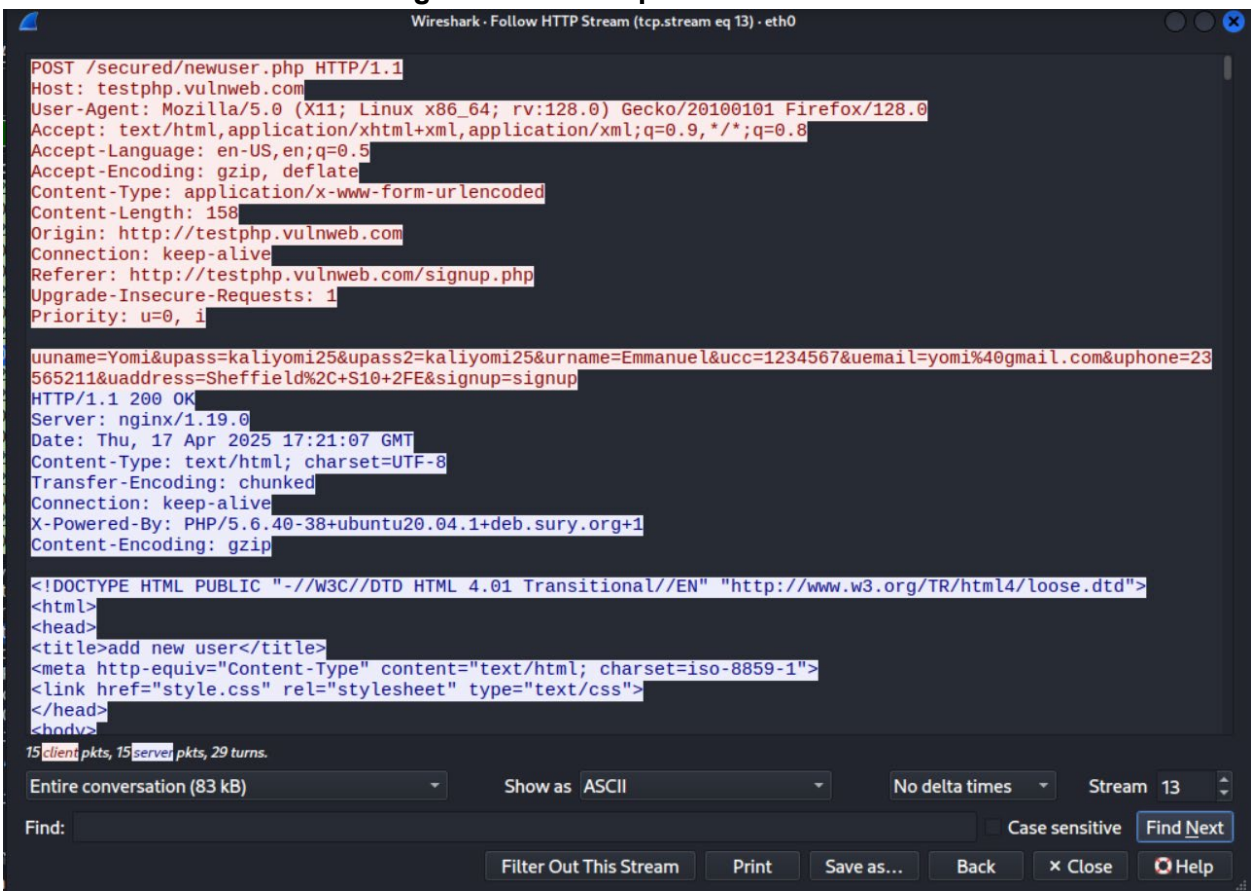


Fig 3.2 Pcap Http Stream

4. **Perform SQL injection on the login or search page of http://testphp.vulnweb.com/ and check if the website is vulnerable to SQLi by extracting database information.**

Objective: Check if the login or search field is vulnerable to SQL Injection (SQLi) and extract database info.

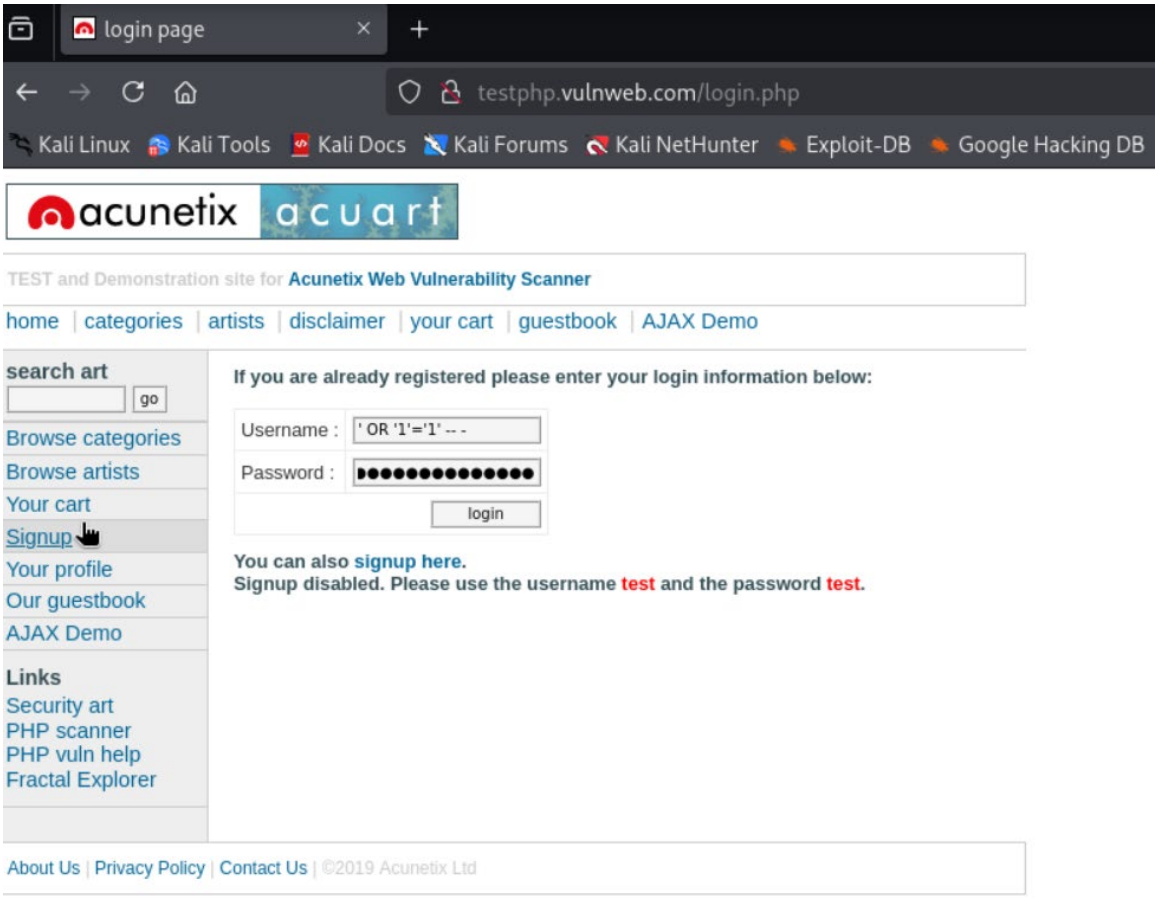


Fig 4.1: S

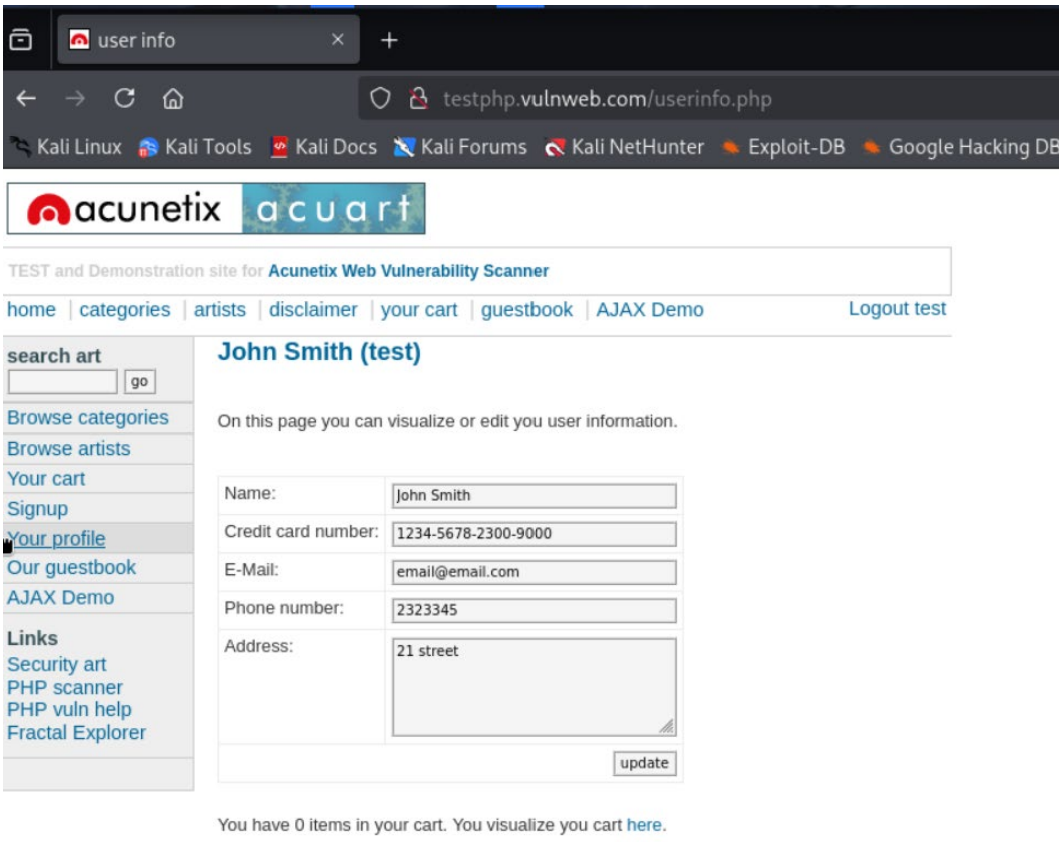


Fig 4.2: SQLi

This is a demonstration of SQL injection, where the input ' OR '1'='1' -- is a classic SQL injection attack. This input is designed to manipulate the SQL query on the server to bypass authentication by making the WHERE clause always evaluate to true and then commenting out the rest of the query with --.

5. Inject malicious JavaScript payloads in input fields (such as the comment section or search box) to see if the website is vulnerable to stored or reflected XSS attacks

   Objective: Test if input fields accept JavaScript code and execute it (indicating XSS vulnerability).

   Payload:

   - **<script>alert(1)</script>**
   - After inserting it on search box , its gives pop up

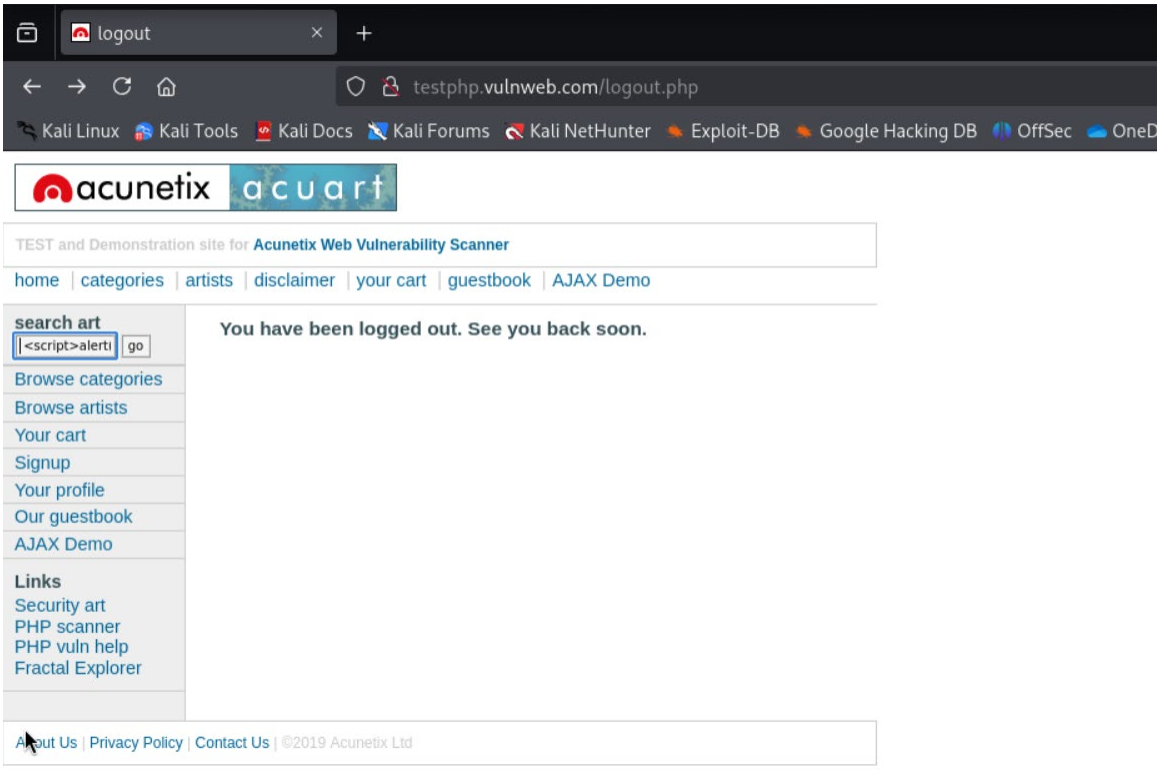


Fig 5.1: XSS Script

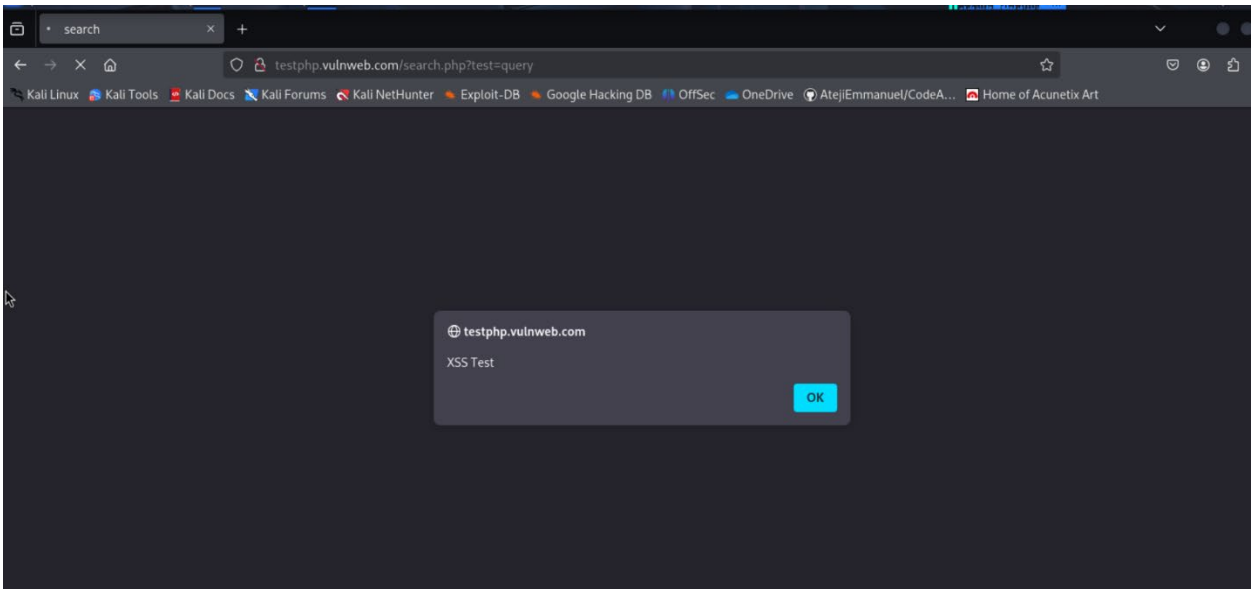


Fig 5.2: pops an alert box, the site is vulnerable to XSS.

TASK (CTF)

Red Team Fundamentals: Core Concepts and Practices

Objective

Understand the essential elements of red team engagements, distinguish them from other security assessments, and comprehend the methodologies and frameworks used in professional red teaming.

Task 1: Introduction to Red Teaming

Red teaming in cybersecurity involves simulating authentic threat actor behaviors to evaluate an organization's security posture. Unlike traditional testing approaches, red teaming specifically focuses on mimicking real-world attacks to assess detection and response capabilities. The primary objective is to emulate adversarial tactics to evaluate and strengthen defensive measures.

Task 2: Vulnerability Assessment and Penetration Tests Limitations

Vulnerability Assessments:

- Aim to discover as many security weaknesses as possible without exploitation
- Provide a comprehensive overview of potential vulnerabilities

Penetration Tests:

- Extend beyond identification to actually exploit discovered vulnerabilities
- Assess the potential impact of successful exploits
- May not accurately reflect real attacker behavior, particularly regarding stealth techniques

Key Insights:

- Standard vulnerability assessments often fail to prepare organizations for detecting actual attackers
- Penetration testers typically aren't concerned about detection, unlike real adversaries who prioritize stealth
- Sophisticated and organized threat actors are classified as Advanced Persistent Threats (APTs)

Task 3: Red Team Engagements

Red team engagements simulate sophisticated attacks to evaluate an organization's detection and response capabilities, focusing on stealth, persistence, and achieving specific objectives without triggering alerts.

Core Concepts:

- **Crown Jewels:** High-value assets or objectives that red teams target
- **TTPs (Tactics, Techniques, and Procedures):** Behavioral patterns of adversaries that red teams replicate
- The primary goal isn't discovering vulnerabilities but assessing the organization's ability to detect and counter sophisticated attacks

Task 4: Teams and Functions of an Engagement

Primary Teams:

- **Red Cell:** Offensive team conducting the simulated attacks
- **Blue Cell:** Defensive team responsible for detection and incident response
- **White Cell:** Supervisory team that oversees the engagement and ensures compliance with rules

Red Team Positions:

- **Red Team Lead:** Plans and manages engagement activities
- **Red Team Assistant Lead:** Supports operations oversight and documentation
- **Red Team Operator:** Executes specific tasks during the engagement

Task 5: Engagement Structure

The Lockheed Martin Cyber Kill Chain framework outlines the progression of a cyberattack:

1. **Reconnaissance:** Gathering target intelligence
2. **Weaponization:** Developing a malicious payload
3. **Delivery:** Transmitting the payload to the target
4. **Exploitation:** Executing the exploit to gain initial access
5. **Installation:** Deploying malware or tools on the compromised system
6. **Command & Control (C2):** Establishing persistent communication
7. **Actions on Objectives:** Executing mission goals (e.g., data exfiltration)

Examples:

- Deploying Mimikatz is part of the Installation phase
- Exploiting vulnerabilities to execute code falls under the Exploitation phase

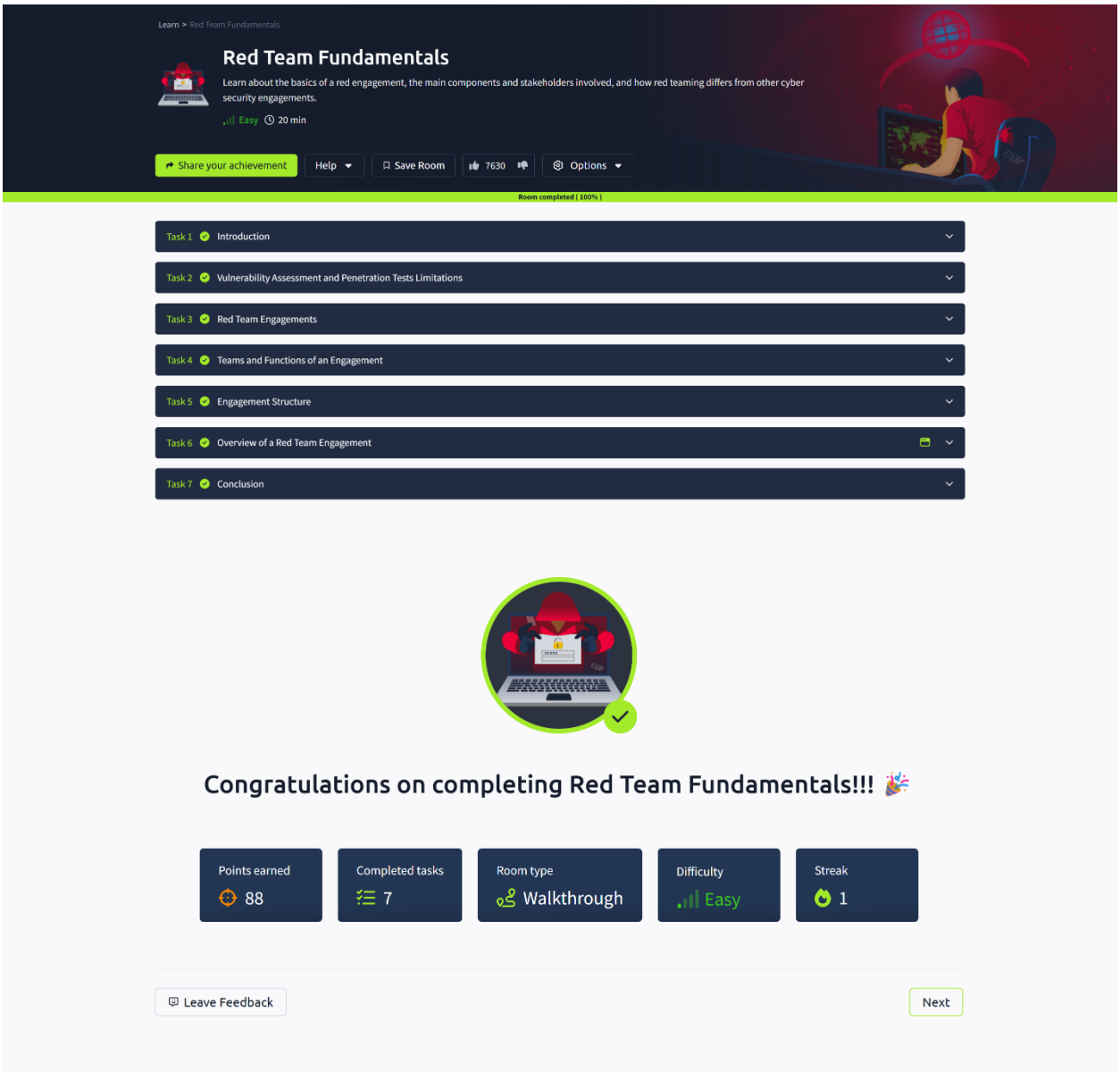
Task 6: Overview of a Red Team Engagement

This section provides a practical demonstration of a red team engagement, applying the concepts covered throughout the previous tasks. The demonstration culminates with:

Flag: THM{RED_TEAM_ROCKS}

Conclusion

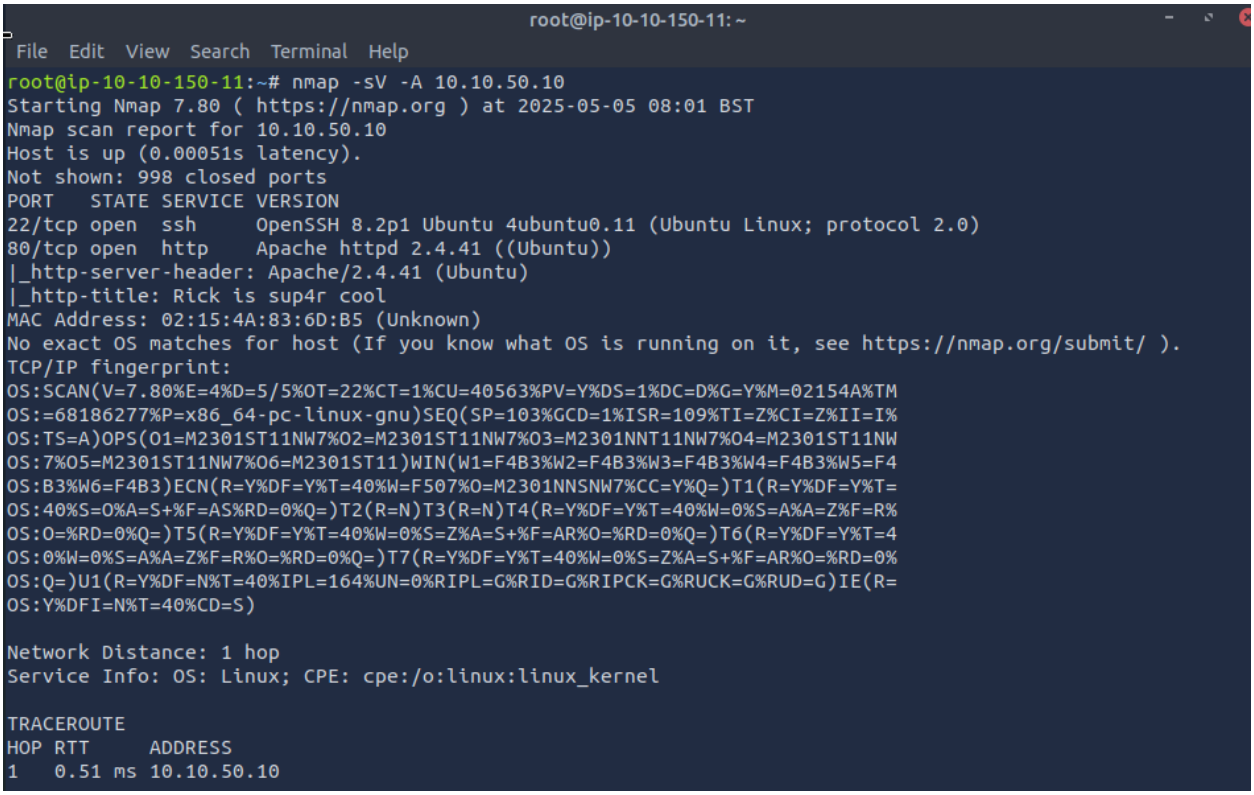
Red team engagements are crucial for evaluating an organization's resilience against sophisticated attacks. These exercises highlight the importance of continuous improvement in detection capabilities and response strategies to counter evolving threats.



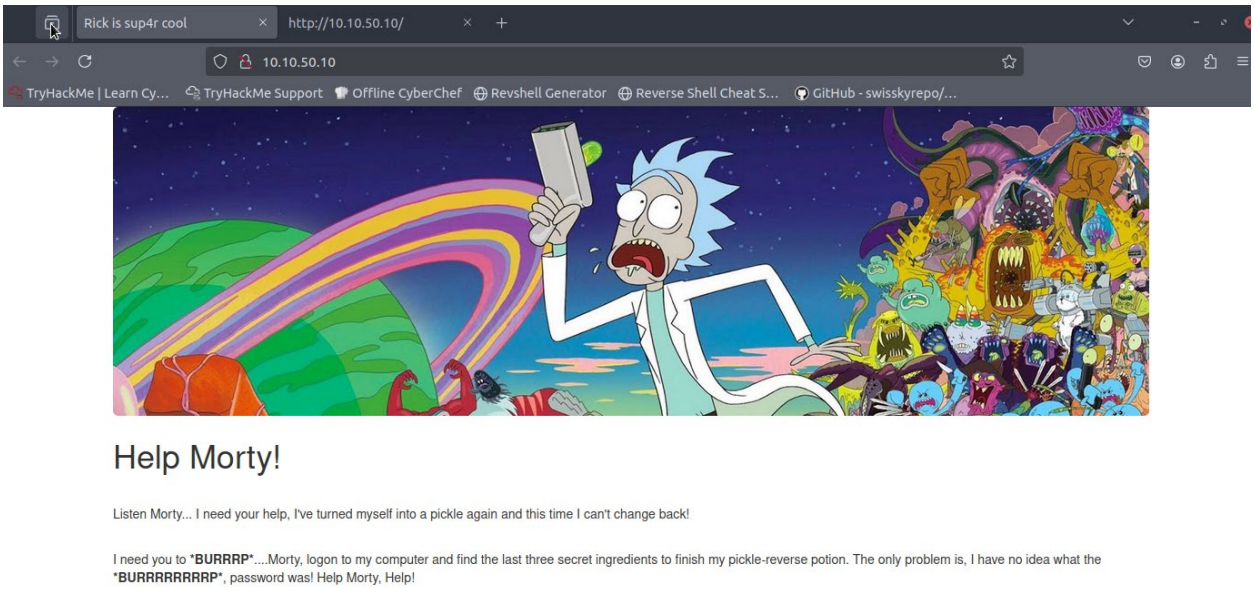
Pickle Rick

Nmap Scan (Network Scanning)

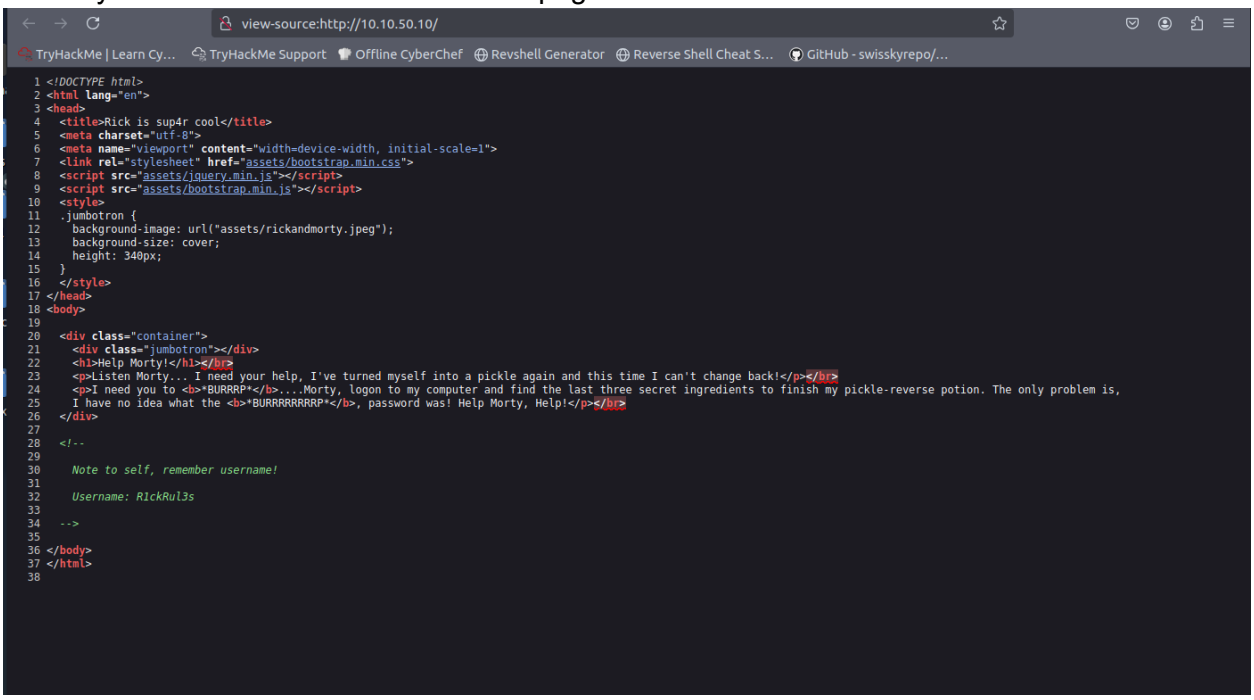
IP Address: 10.10.50.10
Command used : nmap -sV -A 10.10.50.10
Result : Nmap was able to identify 2 services running on the target machine-
SSH (22)
HTTP (80).



We may disregard port 22 for the time being as we don't have an SSH account and password, leaving us only port 80. We visit http:// 10.10.50.10/, where the following page is hosted:



We may view the source code of the webpage:



Something interesting was found below:

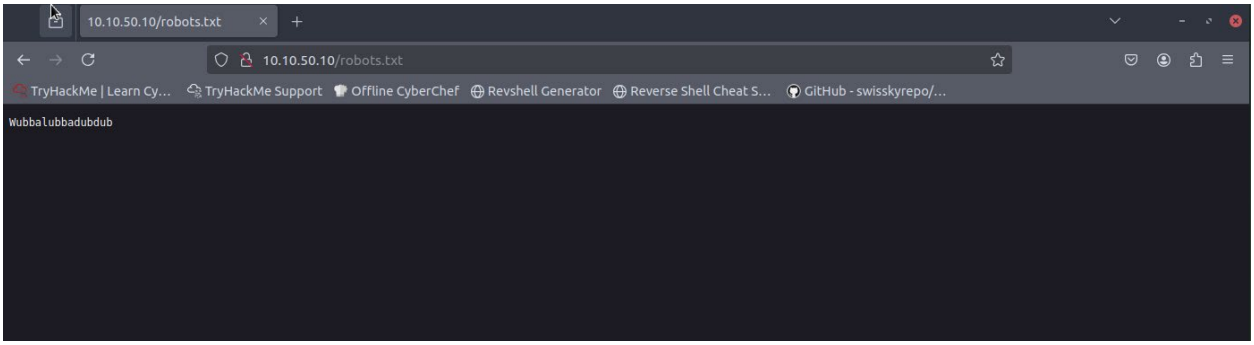
```
28  <!--
29
30    Note to self, remember username!
31
32    Username: R1ckRul3s
33
34  -->
35
36 </body>
37 </html>
38
```

Therefore, R1ckRul3s is our username. Is it for SSH? What is the password, if any? More recon is in order!

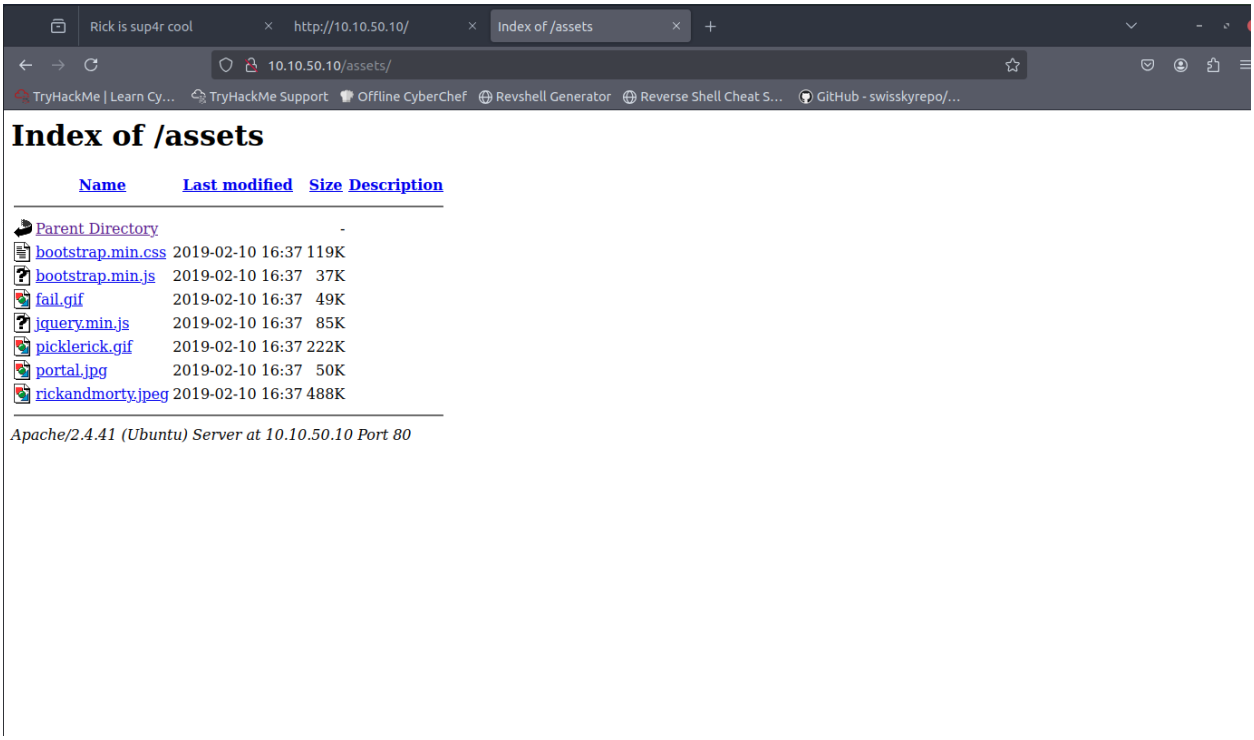
I found it intriguing that there was a folder called "/assets" that included a lot of useful information.

```
4   <title>Rick is sup4r cool</title>
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <link rel="stylesheet" href="assets/bootstrap.min.css">
8   <script src="assets/jquery.min.js"></script>
9   <script src="assets/bootstrap.min.js"></script>
0   <style>
1   .jumbotron {
2     background-image: url("assets/rickandmorty.jpeg");
3     background-size: cover;
4     height: 340px;
```

I therefore get something truly fascinating when I send a curl request to http://10.10.50.10/robots.txt (10.10.50.10 being the server's IP address).



Is a password, then? In the past, we had a username. Is this the password for SSH, then? Nevertheless, I am unable to SSH onto the system when I attempt to do so. This is obviously not the best course of action. Let's return our attention to /assets.



A few pictures and gifs are included here, along with a tonne of CSS and JS code. Using the password we found, I searched for embedded messages in the pictures but was unable to locate anything. I thought I had reached a dead end.

Let's now enumerate our knowledge:

A user using the username ***R1ckRul3s***

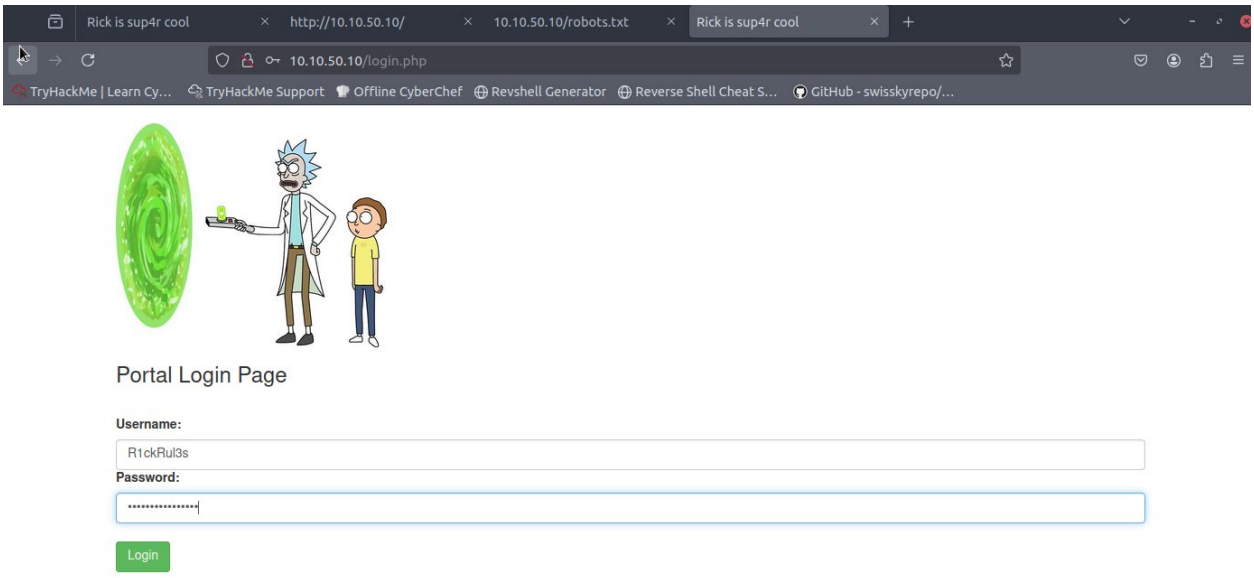
This is the password: ***Wubbalubbadubdub***

There must be a login page someplace because these aren't designed for SSH.

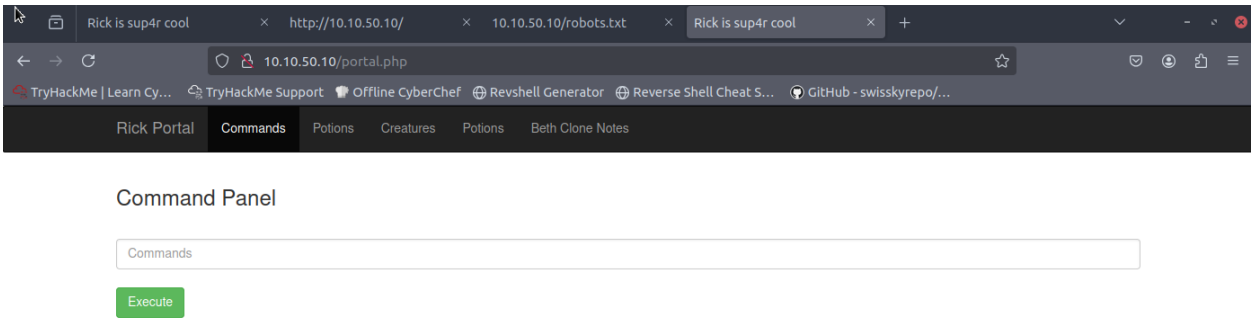
A portal.jpg is now visible if we examine the files under /assets once more.

So, is a /portal possible? Let's investigate. (This is really a rough guess; we could always use gobuster or dirbuster.)

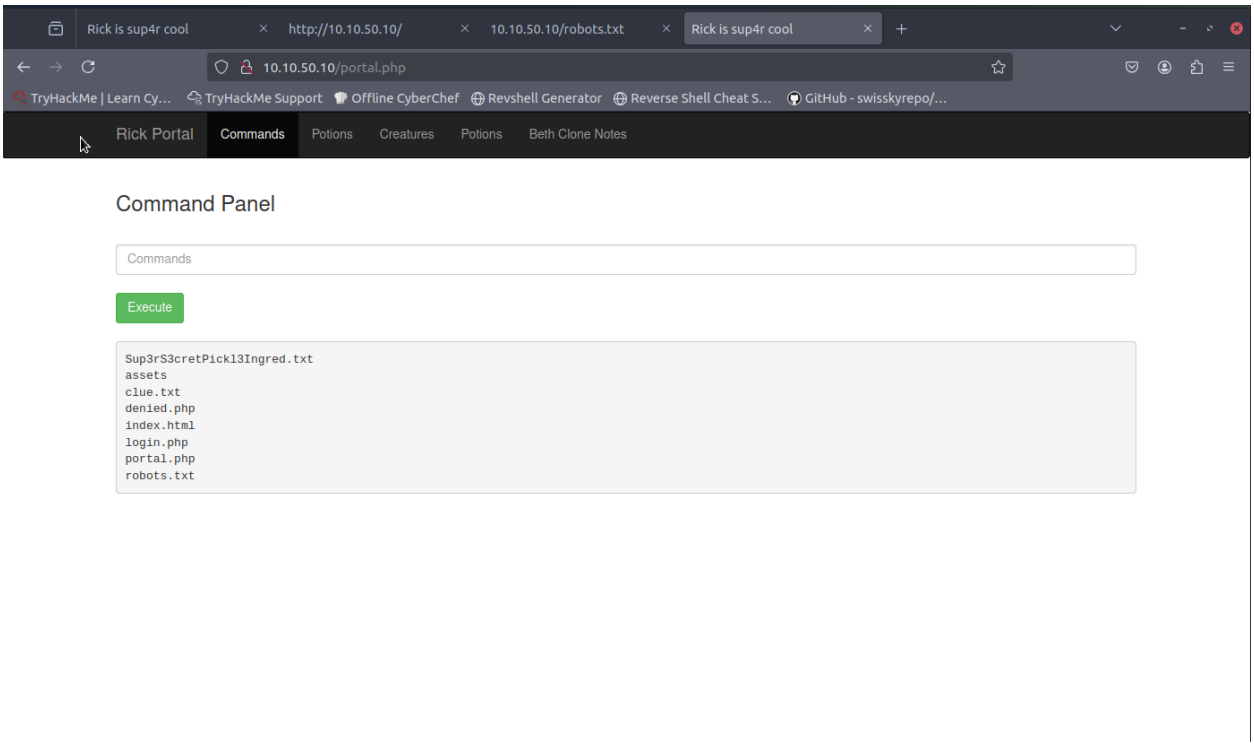
After some stumbling, I discovered this on /login.php or /portal.php:



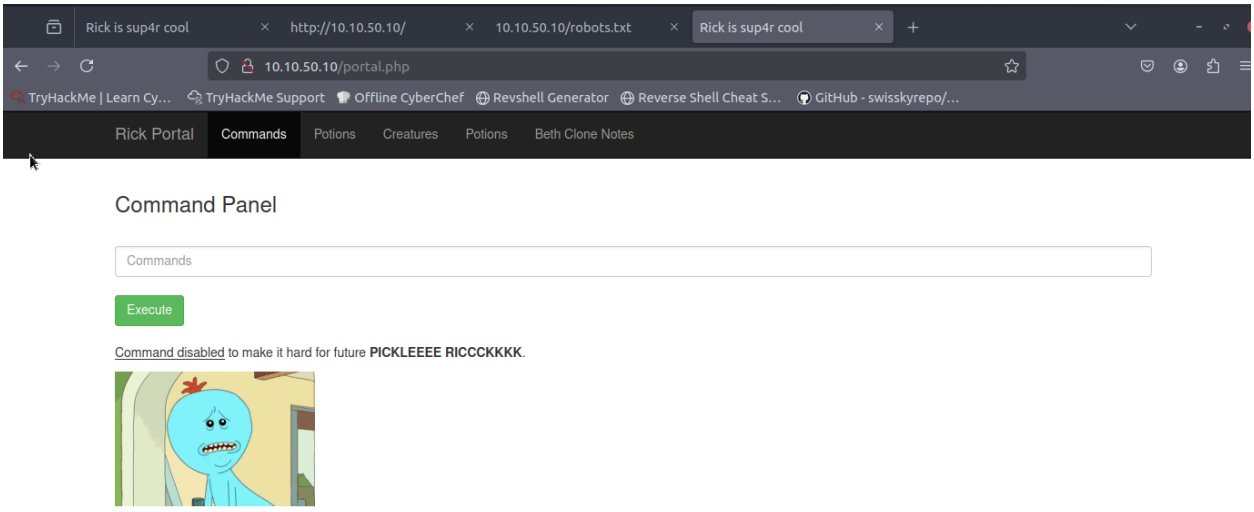
Putting the username and password which we previously got, we are in!

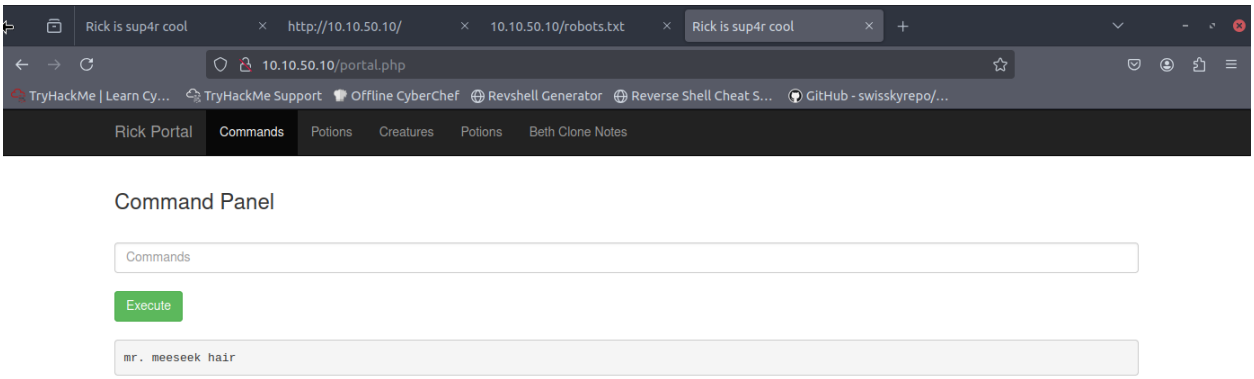


In this case, we can run *some* code and obtain the appropriate result. I entered ‘ls’, and the contents of the folder are displayed. Sup3rS3cretPickl3Ingred.txt is our first secret ingredient!

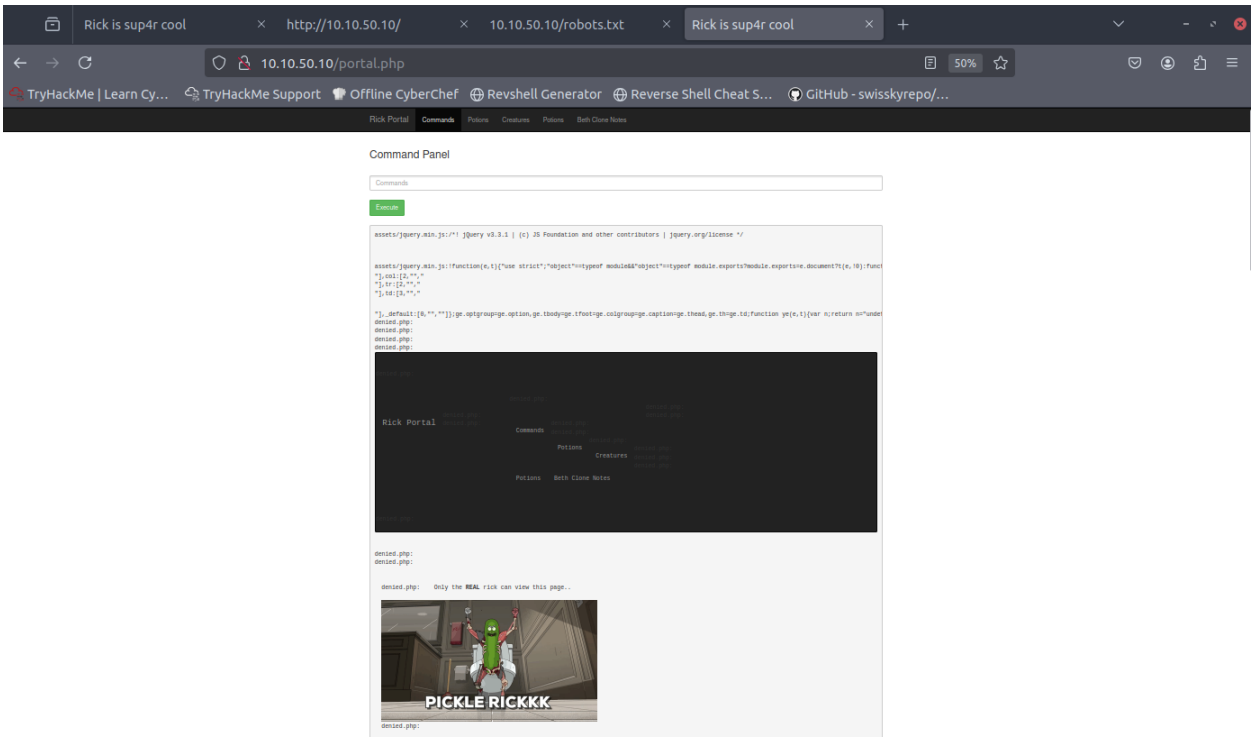


- We used the cat command, but Mr. Meeseek stopped us and claimed that it was limited.
- Let's look for a substitute for the ls, less command.

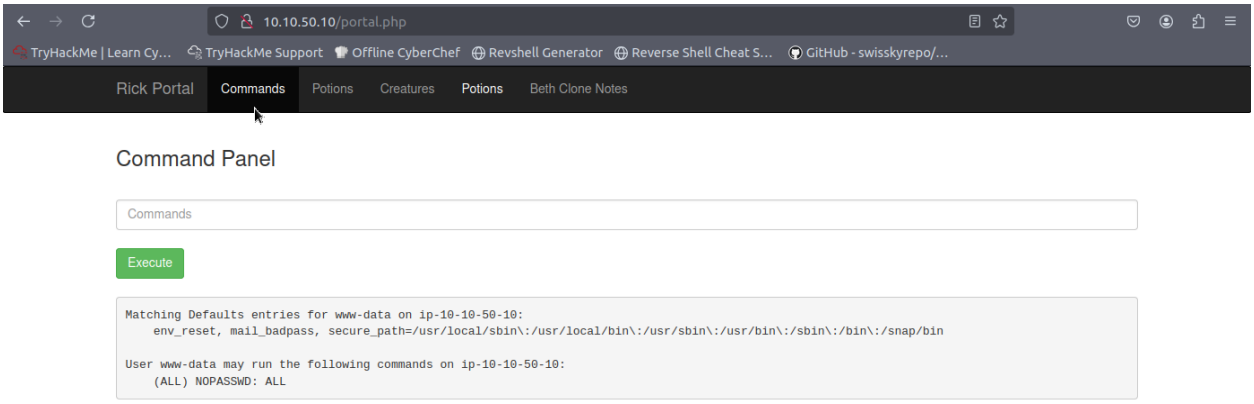




In order to obtain a list of every command that has been blacklisted, we now attempt to obtain the source PHP code. The output that appears when we enter grep -R "" in the command box is as follows:

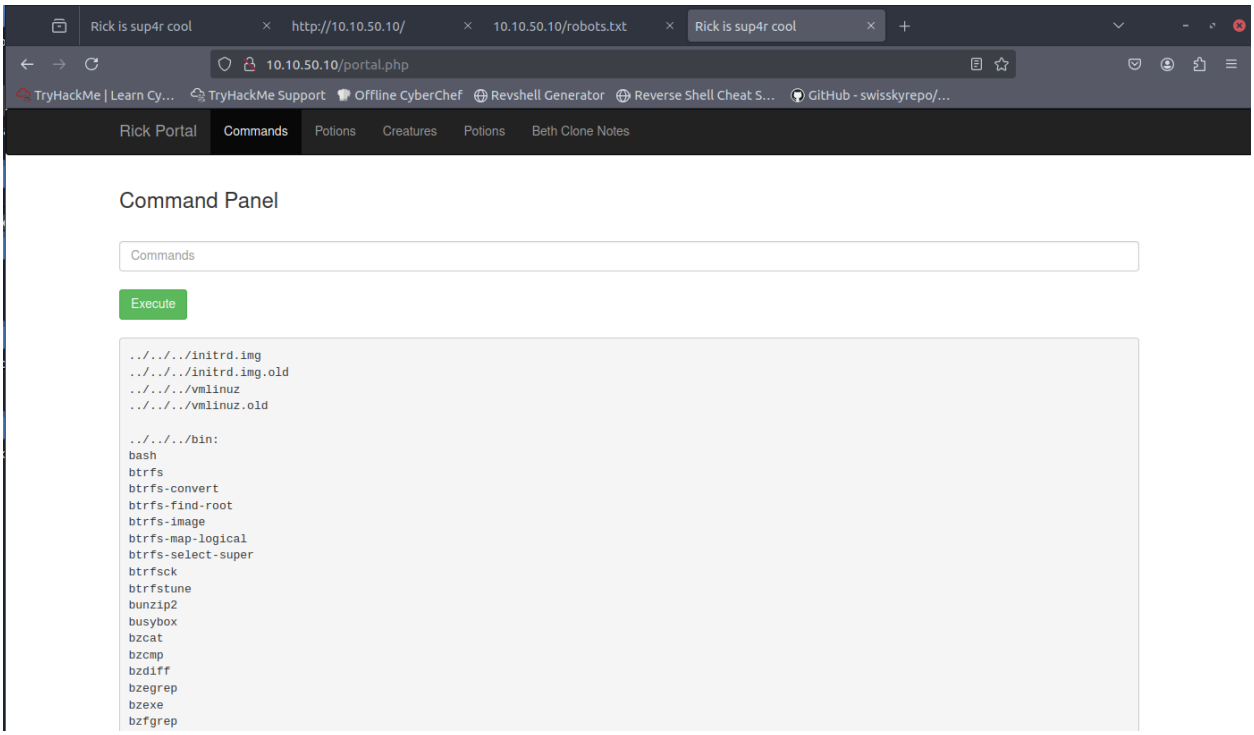


Examining the source, we obtain the following set of commands that are blocked:
Fortunately for us, the ls command is not prohibited. We now need to use sudo -l to verify for sudo rights.



I'm thrilled! We don't need a password to execute sudo instructions!

To locate the last two flags, enter sudo ls../../../*.

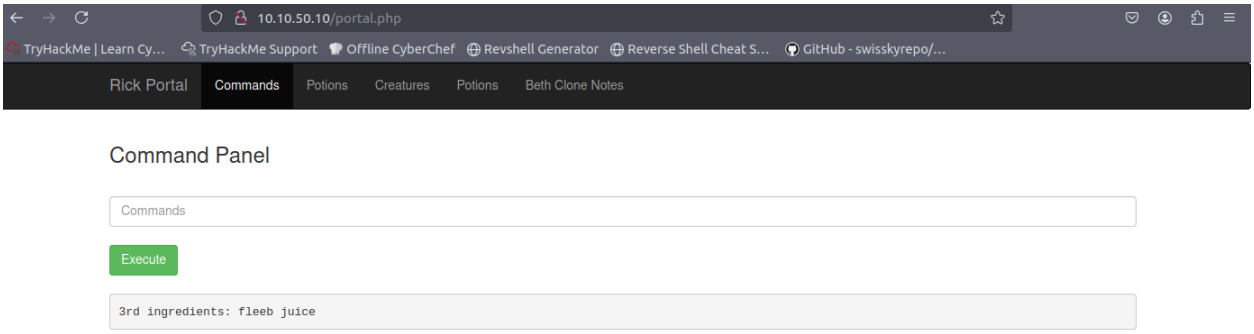


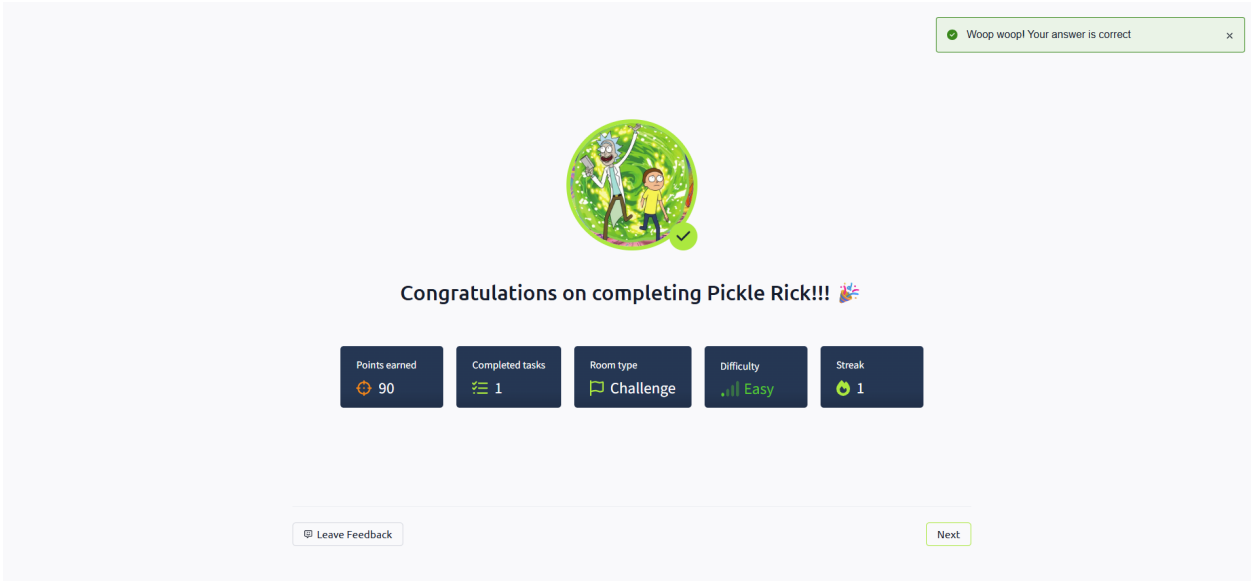
This is a list of every folder and subdirectory in the root. A careful look reveals a flag: /root/3rd.txt.

We obtain the flag as follows by using less (sudo less in the case of the third flag):

```
../../../root:
3rd.txt
snap
```

Fleeb juice, the challenge's last ingredient or flag, appeared.





ETHICAL HACKING PROJECT

1. **Password Strength Checker**

- Create a Python program to evaluate password strength based on length, uppercase/lowercase letters, numbers, and special characters.
- Provide feedback like "Weak," "Moderate," or "Strong."

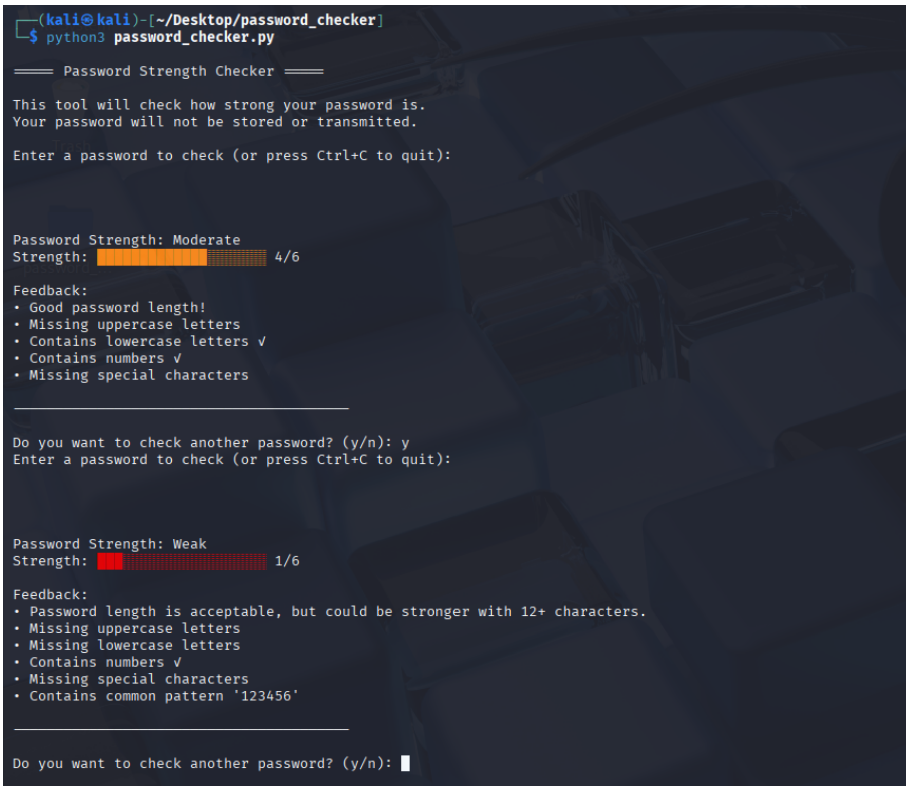
```
#!/usr/bin/env python3

import re
import getpass
import sys
import string

def check_password_strength(password):
    """
    Check the strength of a password based on various criteria.
    Returns a score and feedback.
    """
    score = 0
    feedback = []

    # Check length
    if len(password) < 8:
        feedback.append("Password is too short. Minimum 8 characters
recommended.")
    elif len(password) >= 12:
        score += 2
        feedback.append("Good password length!")
    else:
        score += 1
        feedback.append("Password length is acceptable, but could be
stronger with 12+ characters.")

    # Check for uppercase letters
    if re.search(r'[A-Z]', password):
        score += 1
        feedback.append("Contains uppercase letters ✓")
    else:
        feedback.append("Missing uppercase letters")

    # Check for lowercase letters
```

```
    if re.search(r'[a-z]', password):
        score += 1
        feedback.append("Contains lowercase letters ✓")
    else:
        feedback.append("Missing lowercase letters")

    # Check for numbers
    if re.search(r'[0-9]', password):
        score += 1
        feedback.append("Contains numbers ✓")
    else:
        feedback.append("Missing numbers")

    # Check for special characters
    if re.search(r'[!@#$%^&*(),.?":{}|<>]', password):
        score += 1
        feedback.append("Contains special characters ✓")
    else:
        feedback.append("Missing special characters")

    # NEW: Check for repetitive characters (more than 2 in a row)
    if re.search(r'(.)\1{2,}', password):
        score -= 1
        feedback.append("Contains repetitive characters (e.g., 'aaa',
'111')")

    # NEW: Check for sequential characters
    # Check for sequential letters
    for i in range(len(password) - 2):
        if (ord(password[i].lower()) + 1 == ord(password[i+1].lower())
and
            ord(password[i+1].lower()) + 1 ==
ord(password[i+2].lower())):
            score -= 1
            feedback.append("Contains sequential letters (e.g., 'abc',
'xyz')")
            break

    # Check for sequential numbers
    for i in range(len(password) - 2):
        if (password[i].isdigit() and password[i+1].isdigit() and
password[i+2].isdigit()):
            if (int(password[i]) + 1 == int(password[i+1]) and
                int(password[i+1]) + 1 == int(password[i+2])):
```

```
                score -= 1
                feedback.append("Contains sequential numbers (e.g.,
'123', '789')")
                break

    # NEW: Check for keyboard patterns
    keyboard_rows = [
        "qwertyuiop",
        "asdfghjkl",
        "zxcvbnm"
    ]

    for row in keyboard_rows:
        for i in range(len(row) - 2):
            pattern = row[i:i+3].lower()
            if pattern in password.lower():
                score -= 1
                feedback.append(f"Contains keyboard pattern
'{pattern}'")
                break
        else:
            continue
        break

    # Check for common patterns
    common_patterns = [
        "password", "123456", "qwerty", "admin", "welcome",
        "abc123", "letmein", "monkey", "1234", "12345",
        "football", "baseball", "dragon", "master", "sunshine",
        "ashley", "bailey", "shadow", "superman", "trustno1"
    ]

    for pattern in common_patterns:
        if pattern in password.lower():
            score -= 1
            feedback.append(f"Contains common pattern '{pattern}'")
            break

    # NEW: Check for character diversity across the password
    char_groups = [
        sum(1 for c in password if c.islower()),  # lowercase
        sum(1 for c in password if c.isupper()),  # uppercase
        sum(1 for c in password if c.isdigit()),  # digits
        sum(1 for c in password if c in string.punctuation)  # special
```

```
    ]

    # Calculate character diversity
    non_zero_groups = sum(1 for g in char_groups if g > 0)
    diversity_ratio = sum(g for g in char_groups if g > 0) /
len(password) if len(password) > 0 else 0

    if non_zero_groups >= 3 and diversity_ratio > 0.7:
        score += 1
        feedback.append("Good character diversity throughout password
✓")

    # Determine strength based on score
    strength = ""
    if score <= 2:
        strength = "Weak"
    elif score <= 4:
        strength = "Moderate"
    elif score <= 6:
        strength = "Strong"
    else:
        strength = "Very Strong"

    return score, strength, feedback

def display_strength_bar(score):
    """Display a visual representation of password strength"""
    max_score = 7  # Updated max score with new checks
    bar_length = 20
    filled_length = int(round(bar_length * max(0, score) / max_score))

    # Choose color based on score
    if score <= 2:
        color = '\033[91m'  # Red
    elif score <= 4:
        color = '\033[93m'  # Yellow
    elif score <= 6:
        color = '\033[92m'  # Green
    else:
        color = '\033[94m'  # Blue for very strong

    reset = '\033[0m'  # Reset color
    bar = color + '' * filled_length + '' * (bar_length -
filled_length) + reset
```

```
    print(f"Strength: {bar} {score}/{max_score}")

def main():
    print("\n===== Password Strength Checker =====\n")
    print("This tool will check how strong your password is.")
    print("Your password will not be stored or transmitted.\n")

    while True:
        try:
            # Use getpass to hide the password input
            password = getpass.getpass("Enter a password to check (or
press Ctrl+C to quit): ")

            # Check password strength
            score, strength, feedback =
check_password_strength(password)

            # Clear the previous results (for better UI)
            print("\n" * 2)

            # Display results
            print(f"\nPassword Strength: {strength}")
            display_strength_bar(score)

            print("\nFeedback:")
            for item in feedback:
                print(f"• {item}")

            print("\n" + "-" * 40)

            # Ask if user wants to check another password
            again = input("\nDo you want to check another password?
(y/n): ")
            if again.lower() != 'y':
                print("Thank you for using Password Strength Checker!")
                break

        except KeyboardInterrupt:
            print("\n\nExiting Password Strength Checker. Goodbye!")
            sys.exit(0)

if __name__ == "__main__":
    main()
```



2. Basic Port Scanner

- Basic Port Scanner Write a Python script to scan open ports on a given IP address and port range.
- Handle invalid inputs and display open ports.

```
#!/usr/bin/env python3
import socket
import sys
import re
import argparse
from concurrent.futures import ThreadPoolExecutor
import time

def is_valid_ip(ip):
    """Validate if the given string is a valid IPv4 address."""
    pattern = r'^(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})$'
    match = re.match(pattern, ip)
    if not match:
        return False
```

```
    # Check each octet is between 0 and 255
    for octet in match.groups():
        if int(octet) > 255:
            return False
    return True

def scan_port(ip, port, timeout=1):
    """Scan a specific port on the given IP address."""
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(timeout)
        result = sock.connect_ex((ip, port))
        if result == 0:
            try:
                service = socket.getservbyport(port)
            except:
                service = "unknown"
            sock.close()
            return port, True, service
        sock.close()
        return port, False, None
    except socket.error:
        return port, False, None

def print_progress(current, total):
    """Print a simple progress bar."""
    bar_length = 30
    percent = current / total
    arrow = '=' * int(bar_length * percent)
    spaces = ' ' * (bar_length - len(arrow))
    sys.stdout.write(f"\r[{arrow}{spaces}] {int(percent * 100)}%
({current}/{total} ports)")
    sys.stdout.flush()

def main():
    parser = argparse.ArgumentParser(description="Basic Port Scanner")
    parser.add_argument("ip", help="IP address to scan")
    parser.add_argument("-p", "--ports", help="Port range to scan (e.g.,
1-1024)", default="1-1024")
    parser.add_argument("-t", "--threads", type=int, help="Number of
threads to use", default=50)
    parser.add_argument("--timeout", type=float, help="Socket timeout in
seconds", default=1.0)
```

```
    args = parser.parse_args()

    # Validate IP address
    if not is_valid_ip(args.ip):
        print(f"Error: '{args.ip}' is not a valid IPv4 address.")
        sys.exit(1)

    # Parse port range
    try:
        if "-" in args.ports:
            start_port, end_port = map(int, args.ports.split("-"))
        else:
            start_port = end_port = int(args.ports)

        if not (1 <= start_port <= 65535 and 1 <= end_port <= 65535):
            raise ValueError("Port numbers must be between 1 and 65535")
        if start_port > end_port:
            start_port, end_port = end_port, start_port
    except ValueError as e:
        print(f"Error: Invalid port range '{args.ports}'. {e}")
        sys.exit(1)

    # Begin scanning
    ports_to_scan = list(range(start_port, end_port + 1))
    total_ports = len(ports_to_scan)

    print(f"\nStarting scan on host {args.ip} for ports {start_port}-
{end_port}")
    print(f"Using {args.threads} threads with {args.timeout}s
timeout\n")

    start_time = time.time()
    open_ports = []

    # Use ThreadPoolExecutor to scan ports concurrently
    with ThreadPoolExecutor(max_workers=args.threads) as executor:
        futures = []
        for port in ports_to_scan:
            futures.append(executor.submit(scan_port, args.ip, port,
args.timeout))

        # Process results as they complete
        for i, future in enumerate(futures, 1):
            port, is_open, service = future.result()
```

```
            if is_open:
                open_ports.append((port, service))
            print_progress(i, total_ports)

    # Calculate scan duration
    duration = time.time() - start_time

    # Display results
    print("\n\nScan completed in {:.2f} seconds".format(duration))

    if open_ports:
        print("\n{:<10} {:<10}".format("PORT", "SERVICE"))
        print("-" * 25)
        for port, service in sorted(open_ports):
            print("{:<10} {:<10}".format(port, service))
        print(f"\nFound {len(open_ports)} open port(s) out of
{total_ports} scanned.")
    else:
        print("\nNo open ports found in the specified range.")

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print("\n\nScan terminated by user.")
        sys.exit(0)
```

```
┌──(kali㉿kali)-[~/Documents/Hack Secure Internship]
└─$ python port_scanner.py 45.33.32.156

Starting scan on host 45.33.32.156 for ports 1-1024
Using 50 threads with 1.0s timeout

[══════════════════════════════] 100% (1024/1024 ports)

Scan completed in 21.24 seconds

PORT       SERVICE
─────────────────────
22         ssh
80         http

Found 2 open port(s) out of 1024 scanned.
```

3. File Encryption/Decryption Tool

- Create a Python program to encrypt and decrypt text files using a secret key with the cryptography library.
- Include options to save the output as a new file.

```
import os
import base64
import argparse
from getpass import getpass
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC


def generate_key(password, salt=None):
    """Generate a Fernet key from a password and salt."""
    if salt is None:
        salt = os.urandom(16)

    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,
        salt=salt,
        iterations=100000,
    )

    key = base64.urlsafe_b64encode(kdf.derive(password.encode()))
    return key, salt


def encrypt_file(file_path, password, output_file=None):
    """Encrypt a text file using a password."""
    try:
        # Generate a key from the password
        key, salt = generate_key(password)

        # Create a Fernet cipher with the key
        cipher = Fernet(key)

        # Read the file
        with open(file_path, 'rb') as file:
            file_data = file.read()

        # Encrypt the data
        encrypted_data = cipher.encrypt(file_data)

        # Prepend the salt to the encrypted data
        final_data = salt + encrypted_data
```

```
        # Determine output file path
        if output_file is None:
            output_file = file_path + '.encrypted'

        # Write the encrypted data to the output file
        with open(output_file, 'wb') as file:
            file.write(final_data)

        print(f"File encrypted successfully. Output: {output_file}")
        return True

    except Exception as e:
        print(f"Encryption failed: {str(e)}")
        return False


def decrypt_file(file_path, password, output_file=None):
    """Decrypt an encrypted text file using a password."""
    try:
        # Read the encrypted file
        with open(file_path, 'rb') as file:
            file_data = file.read()

        # Extract the salt (first 16 bytes)
        salt = file_data[:16]
        encrypted_data = file_data[16:]

        # Generate the key from the password and salt
        key, _ = generate_key(password, salt)

        # Create a Fernet cipher with the key
        cipher = Fernet(key)

        # Decrypt the data
        decrypted_data = cipher.decrypt(encrypted_data)

        # Determine output file path
        if output_file is None:
            if file_path.endswith('.encrypted'):
                output_file = file_path[:-10] + '.decrypted'
            else:
                output_file = file_path + '.decrypted'

        # Write the decrypted data to the output file
```

```
        with open(output_file, 'wb') as file:
            file.write(decrypted_data)

        print(f"File decrypted successfully. Output: {output_file}")
        return True

    except Exception as e:
        print(f"Decryption failed: {str(e)}")
        return False


def main():
    parser = argparse.ArgumentParser(description='Encrypt or decrypt
text files with a password.')

    parser.add_argument('-e', '--encrypt', action='store_true',
help='Encrypt the file')
    parser.add_argument('-d', '--decrypt', action='store_true',
help='Decrypt the file')
    parser.add_argument('-f', '--file', required=True, help='File to
encrypt/decrypt')
    parser.add_argument('-o', '--output', help='Output file path
(optional)')

    args = parser.parse_args()

    if args.encrypt and args.decrypt:
        print("Error: You can only encrypt or decrypt, not both at the
same time.")
        return

    if not (args.encrypt or args.decrypt):
        print("Error: You must specify either encrypt (-e) or decrypt (-
d).")
        return

    if not os.path.isfile(args.file):
        print(f"Error: File '{args.file}' does not exist.")
        return

    # Get password securely
    password = getpass("Enter password: ")

    if args.encrypt:
```

```
        encrypt_file(args.file, password, args.output)
    else:
        decrypt_file(args.file, password, args.output)


if __name__ == "__main__":
    main()
```

Learning Outcomes

- **Technical Proficiency:** I gained practical experience in various cybersecurity tools and techniques, including vulnerability scanning, penetration testing, and cryptographic analysis.
- **Understanding of Cybersecurity Lifecycle:** I developed a comprehensive understanding of the steps involved in securing systems, from initial vulnerability assessment to implementing mitigations.
- **Problem-Solving Skills:** Tackling complex security challenges enhanced my analytical thinking and my ability to devise effective solutions under pressure.
- **Professional Development:** My experience improved my ability to work in a team, communicate technical information clearly, and manage time efficiently in a fast-paced environment.

Challenges and Solutions

- **Adapting to Complex Tools:** Initially, mastering advanced cybersecurity tools like Metasploit and Wireshark was challenging. I overcame this by dedicating time to study tutorials and practicing in a simulated environment, which significantly improved my proficiency.
- **Handling Advanced Security Scenarios:** The complexity of real-world security threats required a deep understanding of underlying principles. I tackled this by engaging in continuous learning and seeking guidance from my coordinator and team members.

Conclusion

My internship at Hack Secure was an enriching experience that significantly expanded my knowledge and skills in cybersecurity. The practical exposure to real-world security challenges and the application of advanced tools have solidified my interest in pursuing a career in cybersecurity. This experience has been instrumental in preparing me for the complexities of the cybersecurity field.

Acknowledgments

I express my sincere gratitude to Hack Secure, especially my mentor, Mr. Aman Pandey, and assistant mentor Mr. Prabhat Raj, for their guidance and support throughout my internship. I also thank Amrita Vishwa Vidyapeetham for providing this internship opportunity, which has been crucial in my personal and professional development.

This report encapsulates the essence of my internship experience, highlighting the integration of academic knowledge with practical skills in a professional setting. It reflects my journey of learning, growth, and development in the field of cybersecurity.