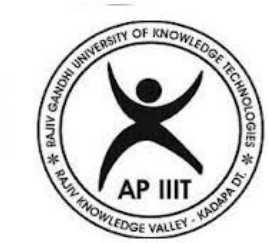
SHADOWFOX ONE MONTH VIRTUAL CYBERSECURITY INTERNSHIP

FINAL REPORT TILL ADVANCED LEVEL TASKS

Submitted by

Kamsala Vandhana

kamsalavandhana07@gmail.com



Department of Computer Science

RGUKT RKV IIIT

VEMPALLI

KADAPA, ANDHRAPRADESH

June 2025 Batch

Assistant Mentor
Mr. Pranshu
ShadowFox

Mentor
Mr. Surendharan
ShadowFox

TABLE OF CONTENTS

Introduction

With the rapid expansion of digital infrastructure, cybersecurity has become a critical concern for organizations of all sizes. Modern network environments, web applications, and data centres are frequently targeted by attackers leveraging sophisticated exploitation techniques. Despite the availability of advanced security measures, vulnerabilities persist due to outdated software, poor configurations, and insufficient security assessments. This has led to increased cyber threats, data breaches, and significant financial and reputational losses.

To address these challenges, penetration testing serves as a proactive measure to identify and exploit vulnerabilities before malicious actors can. It involves a structured approach to reconnaissance, vulnerability analysis, exploitation, and post-exploitation to simulate real-world attacks in a controlled environment.

Information about the report

This report is on the various attacks which I performed during my internship at ShadowFox. It includes *Port Scanning, Directory Busting, and intercepting network traffic via wireshark* on the test website http://testphp.vulnweb.com as the **Beginner Level Tasks**.

It also includes *decrypting a hashed password* and using it to get a secret code from an encrypted *VeraCrypt* Disk, finding the address of the *entry point* of executable file using the *PE tool* and exploitation of Windows 10 machine using *Metasploit* by getting *reverse shell* access as the **Intermediate Level Tasks**.

This report also explains the severity, impact, steps to reproduce and mitigation steps of each attack performed.

Machines & Tools Used

1. VM Ware Workstation Pro
2. Kali Linux
3. Nmap
4. Dirbuster
5. Wireshark
6. VeraCrypt
7. PE Explorer
8. Windows 10
9. Metasploit
10. Enum4linux

I thereby assure that every attack was performed in a secure and virtual environments, abiding by the ethics of Cybersecurity.

- Swastik Gondhi

BEGINNER LEVEL - TASK 1

Find all the ports that are open on the website

http://testphp.vulnweb.com/

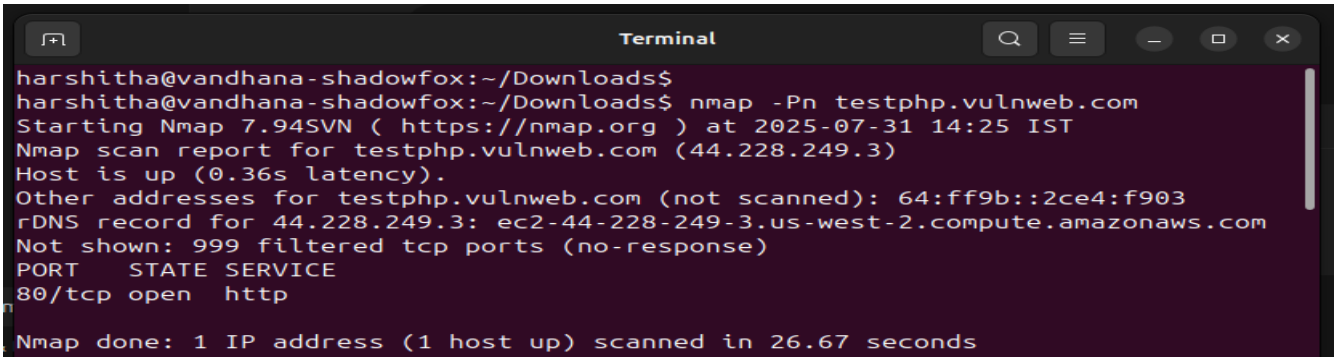
- **Attack Name**
  - *Port Scanning and Fingerprinting of* http://testphp.vulnweb.com/
- **Severity**
  - *CVSS Score: 5.3*
  - *Level: Medium*
- **Impact**

  Port scanning and fingerprinting reveal open ports, running services, and versions, which can be further exploited if vulnerabilities exist. In this case, Nmap detected:
  - ***Port 80 (HTTP)**: Running on nginx 1.19.0.*
  - *The server is powered by **PHP 5.6.40**, which is outdated and potentially vulnerable.*

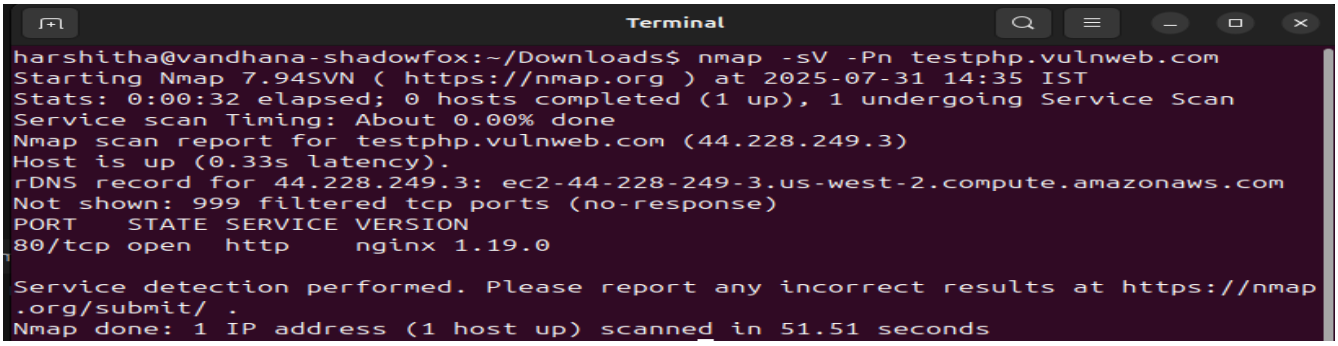
  This exposure could allow attackers to probe for vulnerabilities in outdated versions of Nginx and PHP, increasing the risk of remote code execution (RCE), information leakage, and denial-of-service (DoS) attacks.

- **Steps to Reproduce**

1. Website Fingerprinting:
   - The scan revealed that the server is running `nginx 1.19.0` on `PHP 5.6.40`.
   - Additional information such as ActiveX, Adobe Flash, and server IP `44.228.249.3` was disclosed.



2. Port Scanning with Nmap:



- Discovered open **Port 80 (HTTP)** running on nginx 1.19.0.

• Mitigations Steps

1. **Update Software Versions:**
   - Upgrade `nginx` to the latest stable version.
   - Upgrade `PHP` to a more secure version (preferably 8.x) to patch known vulnerabilities.
2. **Restrict Information Disclosure:**
   - Hide version information in server headers (use `server_tokens off;` in nginx configuration).
   - Remove ActiveX and outdated Adobe Flash if not necessary.
3. **Firewall Configurations:**
   - Apply firewall rules to limit port exposure to only necessary ones.
4. **Run Regular Vulnerability Scans:**
   - Perform regular scans to identify outdated services and vulnerabilities.

BEGINNER LEVEL – TASK 2

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

- **Attack Name**
  - ➢ *Directory Enumeration via Brute Force on* http://testphp.vulnweb.com/
- **Severity**
  - ➢ *CVSS Score: 7.5*
  - ➢ *Level: High*
- **Impact**

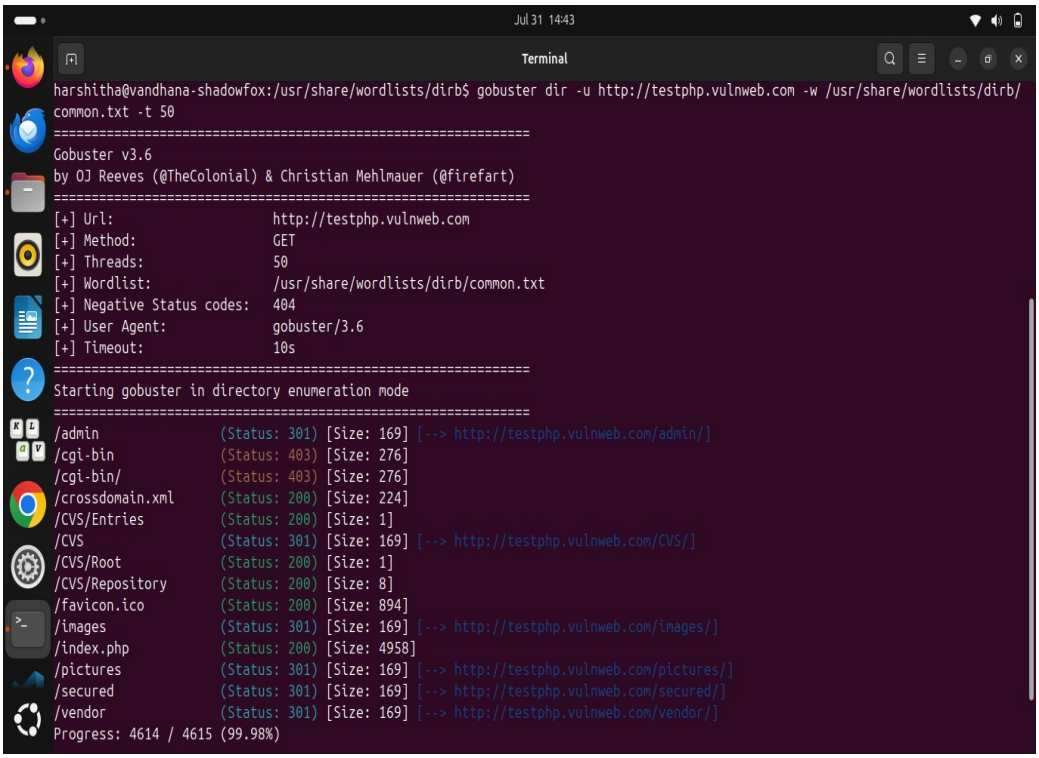
  The directory brute force attack exposed sensitive directories that may contain configuration files, admin panels, and source code repositories. This increases the attack surface, allowing potential access to:

  - `/admin/` - Possible admin panel (403 Forbidden, but visible)
  - `/cgi-bin/` - Common directory for executing scripts (403 Forbidden, but visible)
  - `/CVS/` - Version control repository exposing `Entries`, `Repository`, and `Root`
  - `/crossdomain.xml` - Cross-domain policy file, potentially exposing configurations
  - `/images/` and `/pictures/` - Image directories that may leak sensitive content
  - `/check.php` - A script that is directly accessible (Status 200, Size: 4958)

  Exposed CVS directories may allow attackers to access historical changes, configurations, and even source code, increasing the risk of source code disclosure and configuration weaknesses.

- **Steps to Reproduce**

1. Directory Brute Forcing using Dirb:



2. Analysis of Results:

Accessible Directories:

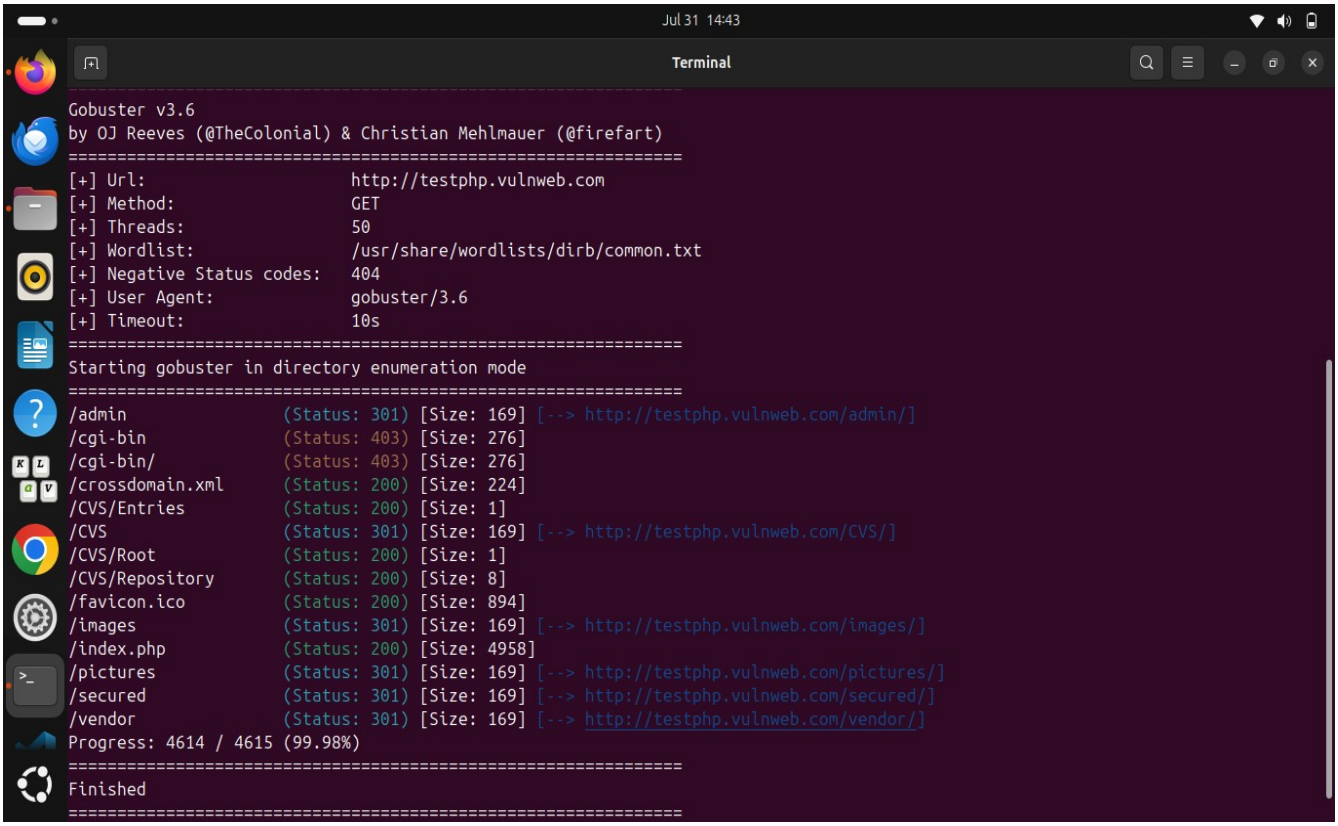
- `/admin`
- `/CVS`
- `/images`
- `/pictures`

- **Mitigation Steps:**

1. **Restrict Directory Access:**
   - Apply proper `.htaccess` rules to deny directory listing and access to sensitive directories like `/admin/`, `/cgi-bin/`, and `/CVS/`.
2. **Disable Unused Services:**
   - If `cgi-bin` is not in use, disable it from the web server configuration.
3. **Secure Version Control Paths:**
   - Ensure version control paths (`/CVS/`) are not publicly accessible.
4. **Validate Cross-Domain Policies:**

- Ensure `crossdomain.xml` is securely configured to allow only trusted domains.

5. **Regular Security Audits:**
   - Perform regular scans and audits to identify exposed paths and sensitive directories.



BEGINNER LEVEL – TASK 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.

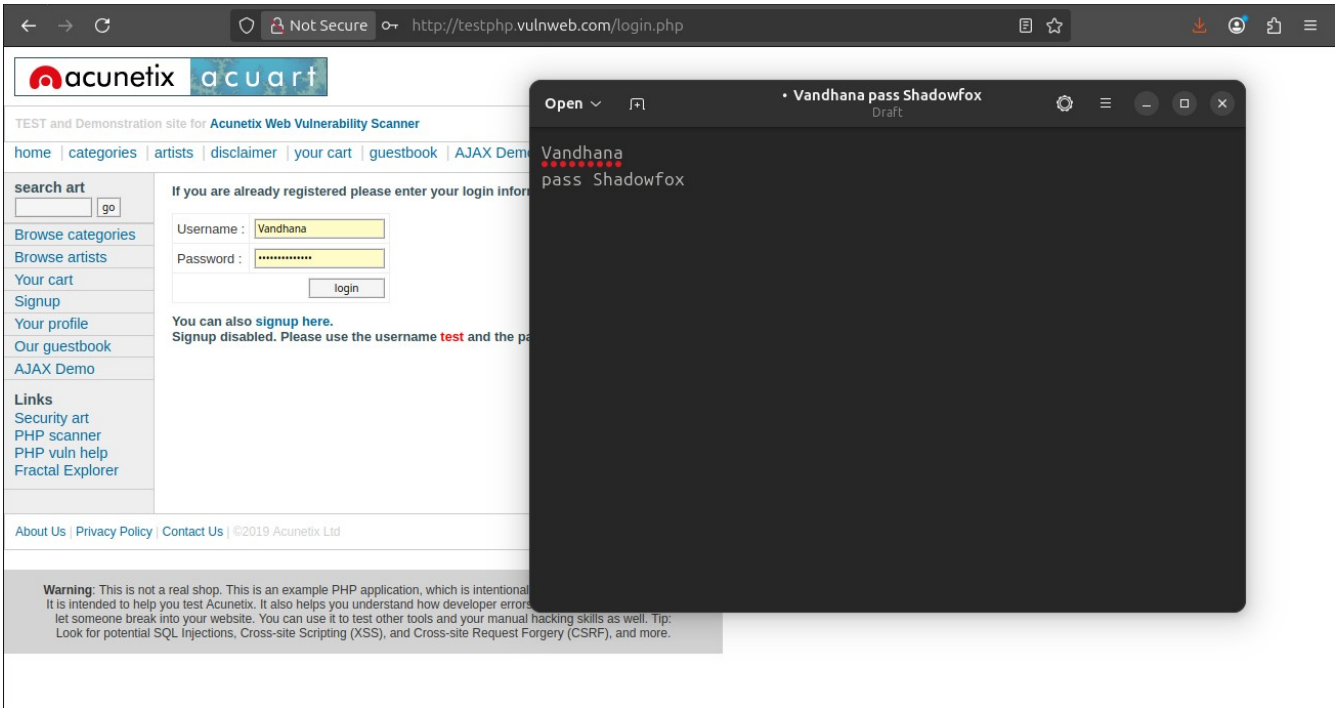
- ## Attack Name
  - ➢ *Intercepting Login Credentials with Wireshark*
- ## Severity
  - ➢ *CVSS Score: 7.5*
  - ➢ *Level: High*
- ## Impact

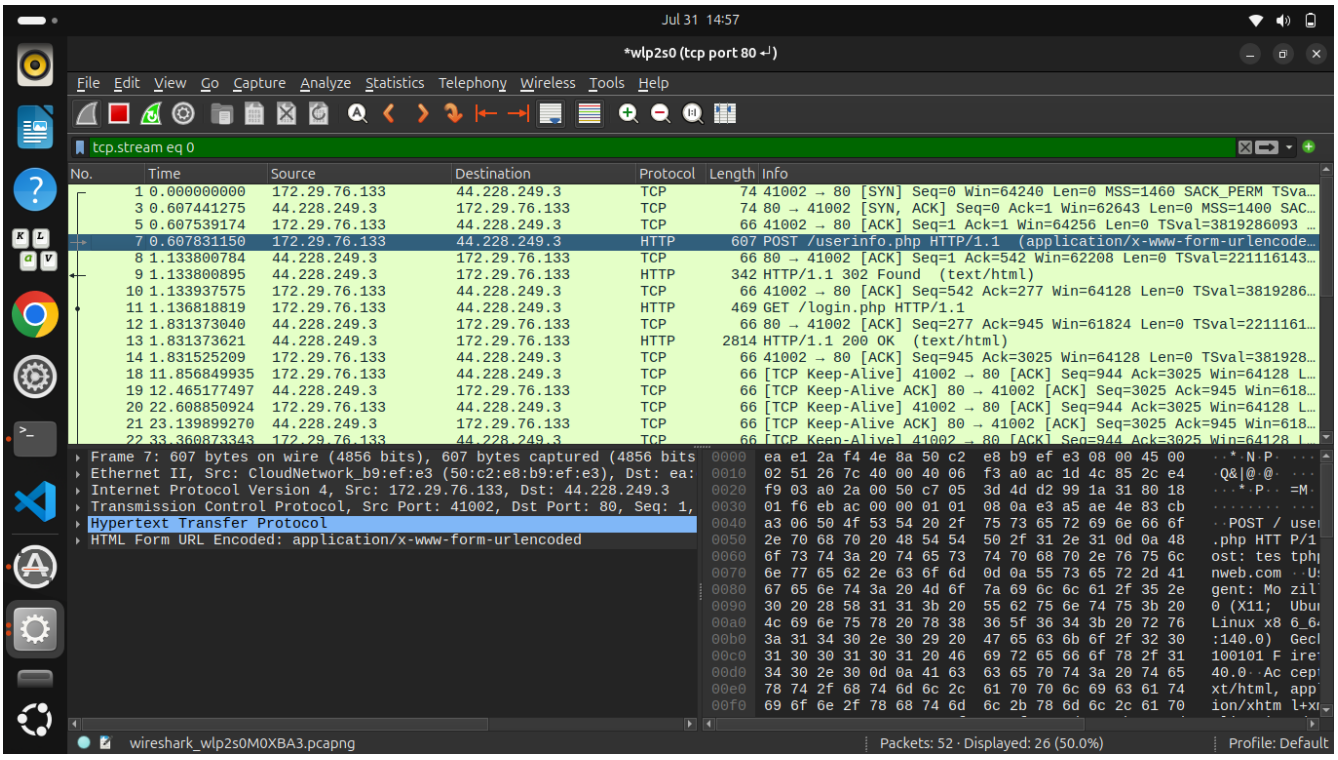
  The attack allows interception of plain-text credentials (username and password) transmitted over an unencrypted HTTP connection. An attacker positioned within the same network (Man-in-the-Middle) can easily capture sensitive information, leading to unauthorized access and potential data breaches.

- ## Steps to Reproduce

1. **Navigate to the Target Website:**
   - Open a browser and go to `http://testphp.vulnweb.com/`.
2. **Initiate a Login Attempt:**
   - Fill in the username and password fields with sample credentials and submit the form.
3. **Launch Wireshark:**
   - Open Wireshark and start capturing traffic on the active network interface.
4. **Filter the Traffic:**
   - Use the display filter `http` to isolate HTTP traffic.
5. **Locate the Login Request:**
   - Search for a POST request to `/login.php` or similar endpoint.
   - Inspect the packet to find the `username` and `password` parameters in plain text.

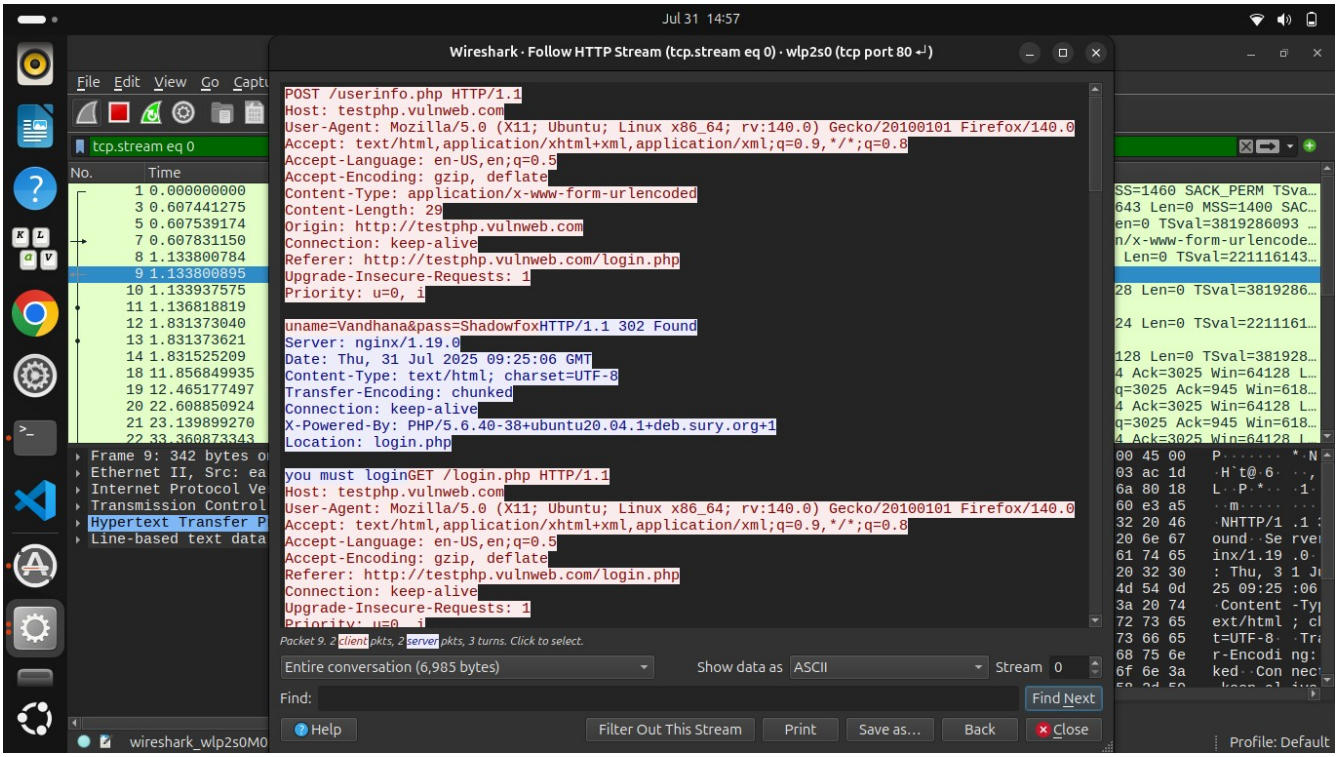
6. **Capture the Credentials:**
    - Right-click the packet → Follow → HTTP Stream.
    - The credentials should be visible in the stream.





- # Mitigation Steps

1. **Enforce HTTPS:**
    - Use SSL/TLS to encrypt HTTP traffic and prevent credential interception.
2. **Use Secure Cookies:**
    - Mark cookies as `Secure` and `HttpOnly` to avoid exposure in unencrypted channels.
3. **Implement Strong Authentication Mechanisms:**
    - Utilize multi-factor authentication (MFA) to add a second layer of protection.
4. **Network Segmentation:**
    - Isolate critical applications from public networks to reduce exposure.
5. **Regular Monitoring:**
    - Continuously monitor network traffic for signs of interception or anomalies.



INTERMEDIATE LEVEL – TASK 1

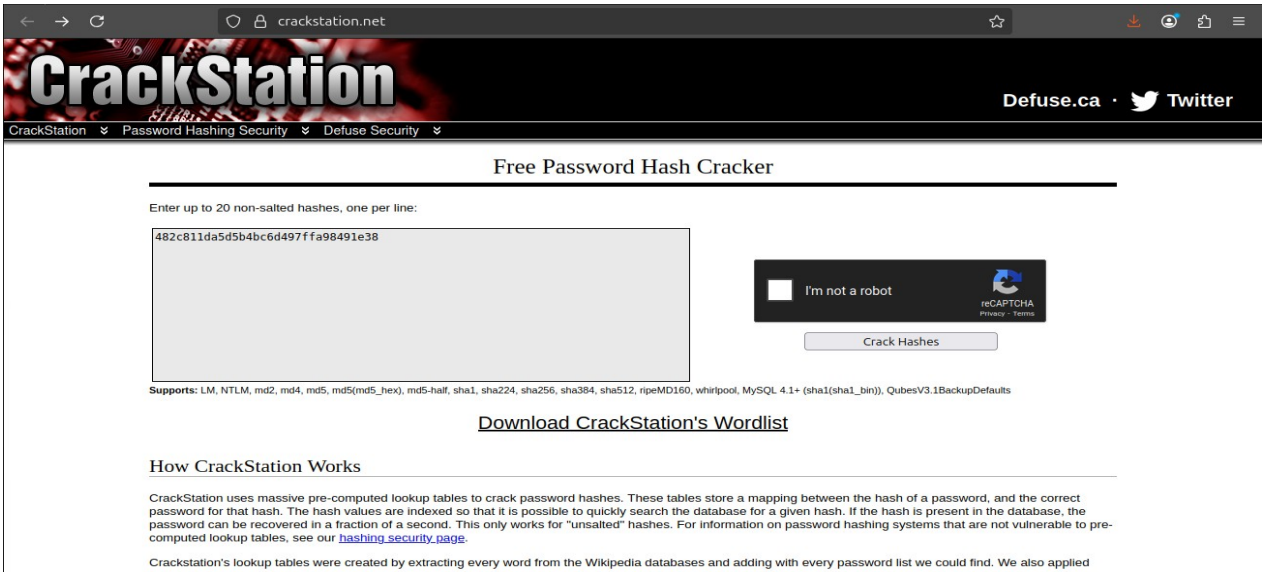
A file is encrypted using VeraCrypt (A disk encryption tool). The password to access the file is encrypted in a hash format and provided to you in the drive with the name encoded.txt. Decode the password and enter in the vera crypt to unlock the file and find the secret code in it.

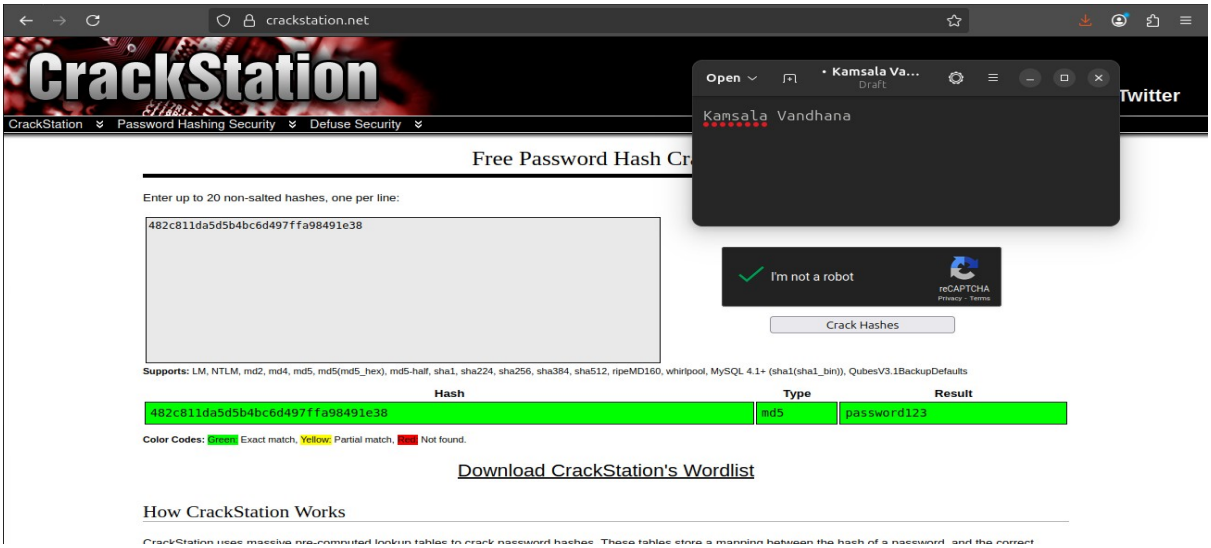
- ## Attack Name
  - ➢ *VeraCrypt Encrypted File Decryption*
- ## Severity
  - ➢ *CVSS Score: 6.8*
  - ➢ *Level: Medium*
- ## Impact

  The attack demonstrates the ability to decrypt a password-protected file container using VeraCrypt if the hash of the password is known and can be cracked. This exposes sensitive information if password hashes are not properly secured.

- ## Steps to Reproduce

1. **Locate the Encoded Password File:**
   - Access the drive and locate `encoded.txt` which contains the hashed password.
2. **Decode the Hash:**
   - Use the online tool `md5hashing.net` to decode the hash value.
   - The password was successfully decoded as `password123`.





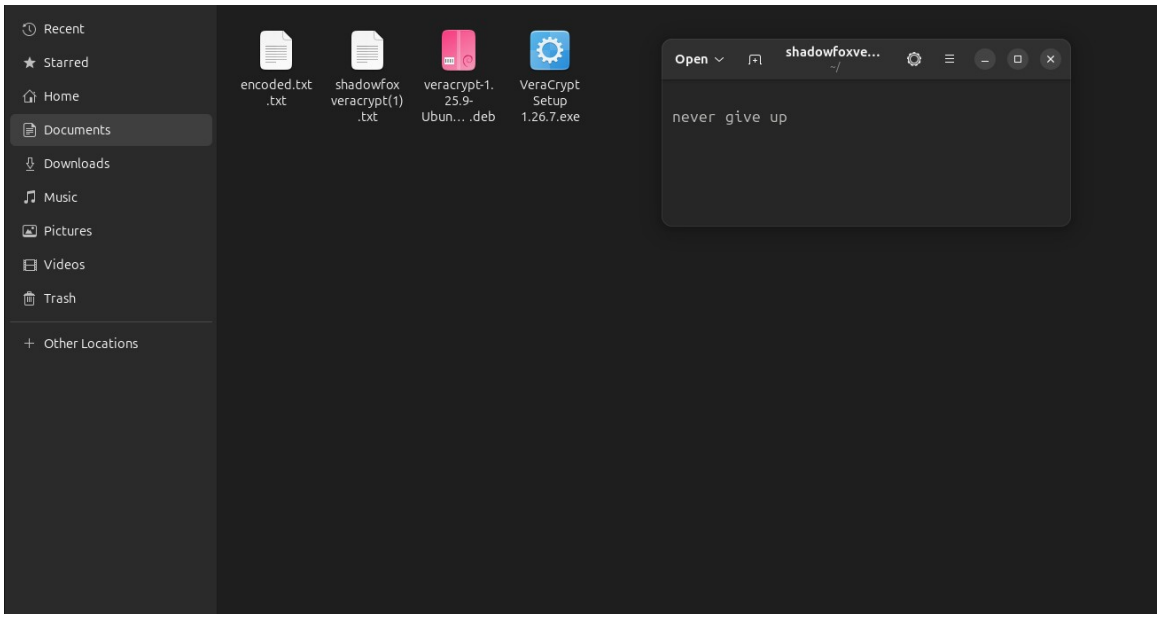
3. **Download and Install VeraCrypt:**
   - Install VeraCrypt and mount the encrypted container.

4. **Enter the Decoded Password:**

   Input password is password123,entered the password

5. **Access the Secret Code:**

   Upon successful decryption, open the file and retrieve the secret code: never give up



- ## Mitigation Steps

1. **Use Strong Passwords:**
    - Avoid using simple, easily crackable passwords. Implement password complexity policies.

2. **Hash with Salt:**
    - Always hash passwords with a unique salt value to prevent hash-based attacks.
3. **Limit Hash Exposure:**
    - Store hashed passwords securely and avoid exposing them unnecessarily.
4. **Multi-Factor Authentication:**
    - Implement MFA to add an additional layer of security to encrypted files.
5. **Monitor Access Logs:**
    - Regularly review access logs to detect unauthorized access attempts.

INTERMEDIATE LEVEL – TASK 2

An executable file of VeraCrypt will be provided to you. Find the address of the entry point of the executable using PE explorer tool and provide the value as the answer as a screenshot

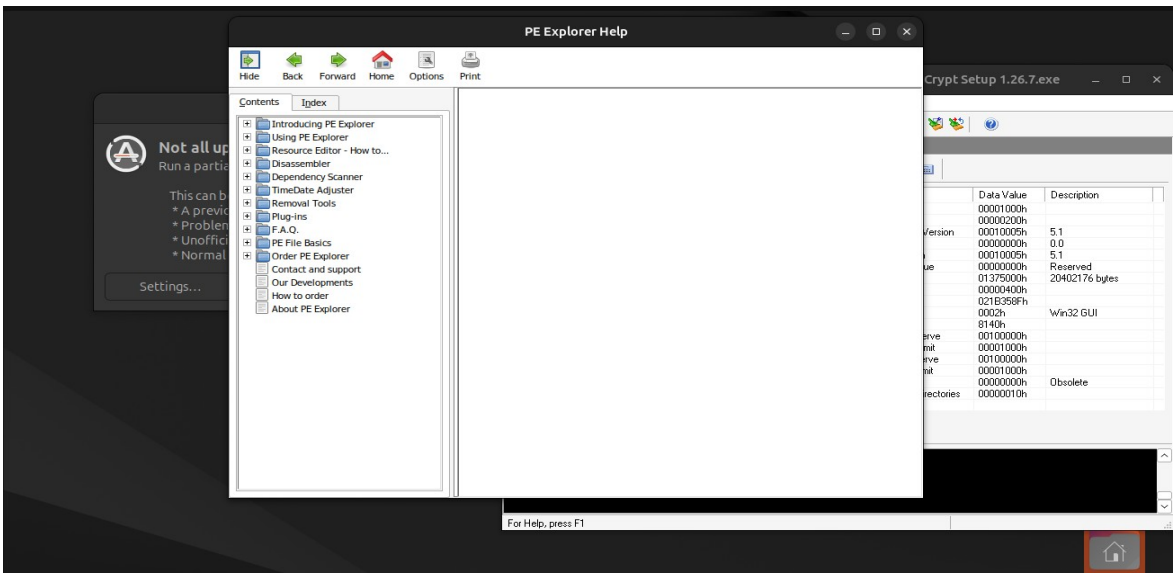
---

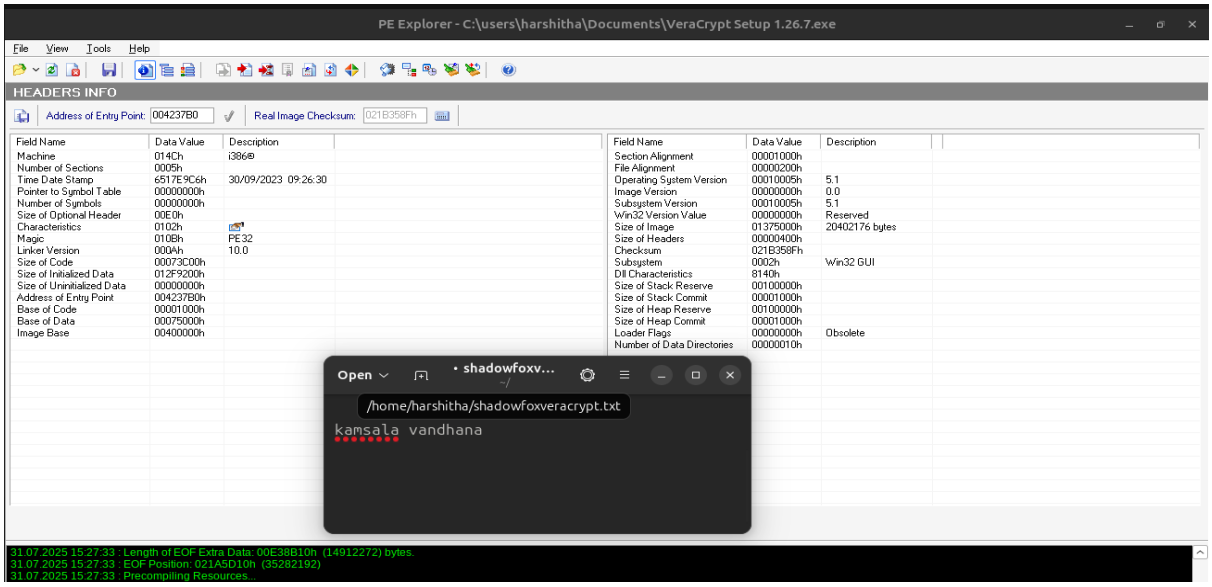
- ## Attack Name
  - ➢ *Finding the Entry Point of VeraCrypt Executable*
- ## Severity
  - ➢ *CVSS Score: 6.8*
  - ➢ *Level: Medium*
- ## Impact

  Identifying the entry point of an executable is crucial for reverse engineering and vulnerability analysis. Gaining this information helps in understanding the program's control flow and potential attack vectors for exploitation.

- ## Steps to reproduce

1. **Obtain VeraCrypt Executable:**
   - Download or access the VeraCrypt executable file.
2. **Open PE Explorer:**
   - Launch the PE Explorer tool.
3. **Upload the VeraCrypt Executable:**
   - Load the executable into PE Explorer.





4. **Locate the Entry Point:**
    - Navigate to the headers section and identify the **Address of Entry Point (AEP)**.
5. **Record the Address:**
    - Note the value displayed as the entry point address.

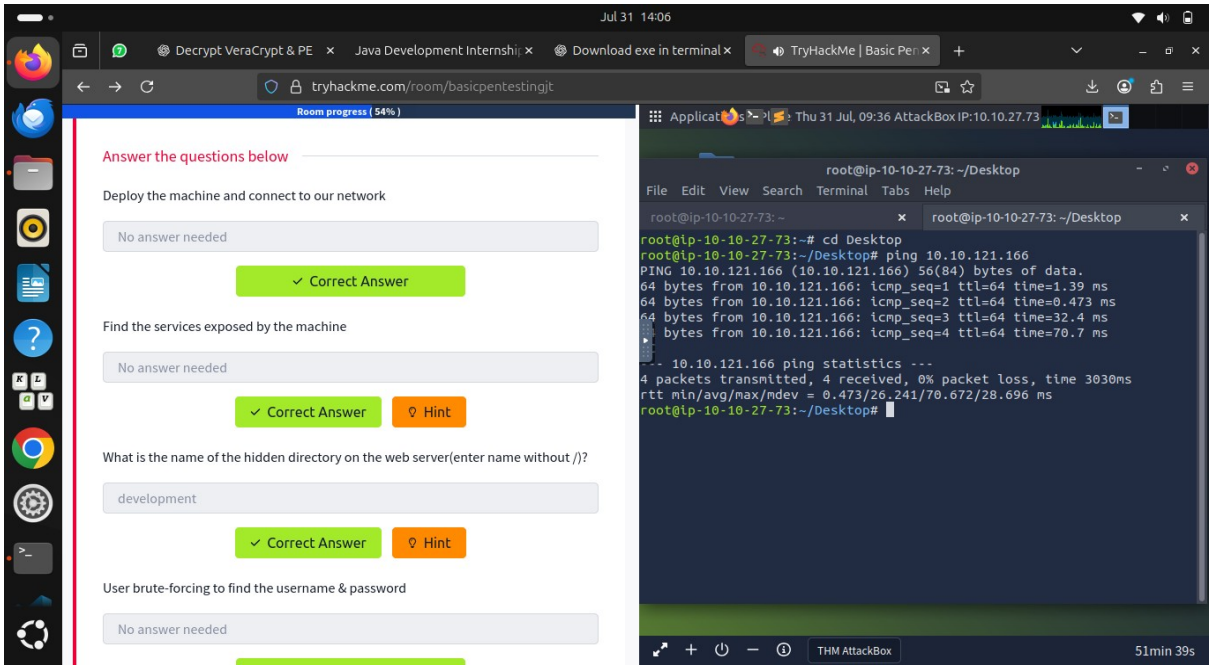
• Mitigation Steps

1. **Binary Obfuscation:**
    - Use obfuscation techniques to make it harder to identify entry points.
2. **Packers and Encryptors:**
    - Apply binary packers to complicate reverse engineering efforts.
3. **Runtime Checks:**
    - Implement runtime verification to detect tampering or unauthorized access.
4. **Regular Updates:**
    - Keep VeraCrypt and PE Explorer tools updated to prevent exploitation through known vulnerabilities.
5. **Limit Executable Access:**
    - Restrict access to executable files and monitor any modifications.

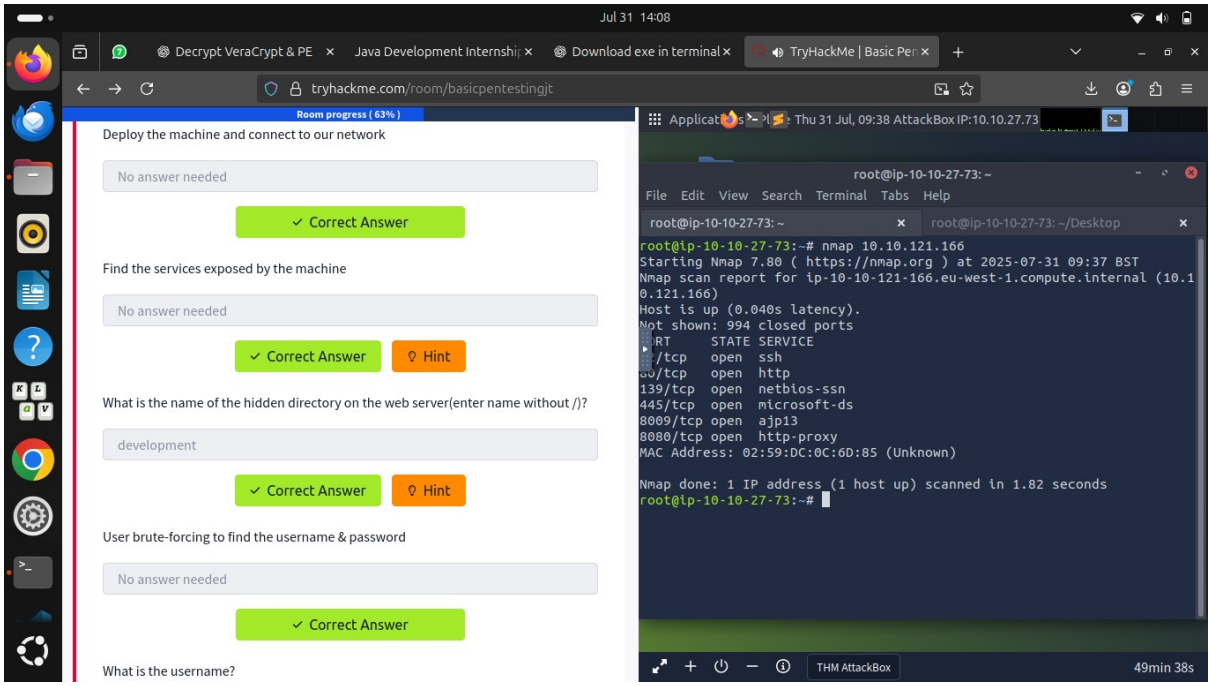
ADVANCED LEVEL TASK

TryHackMe's Basic Pentesting Room

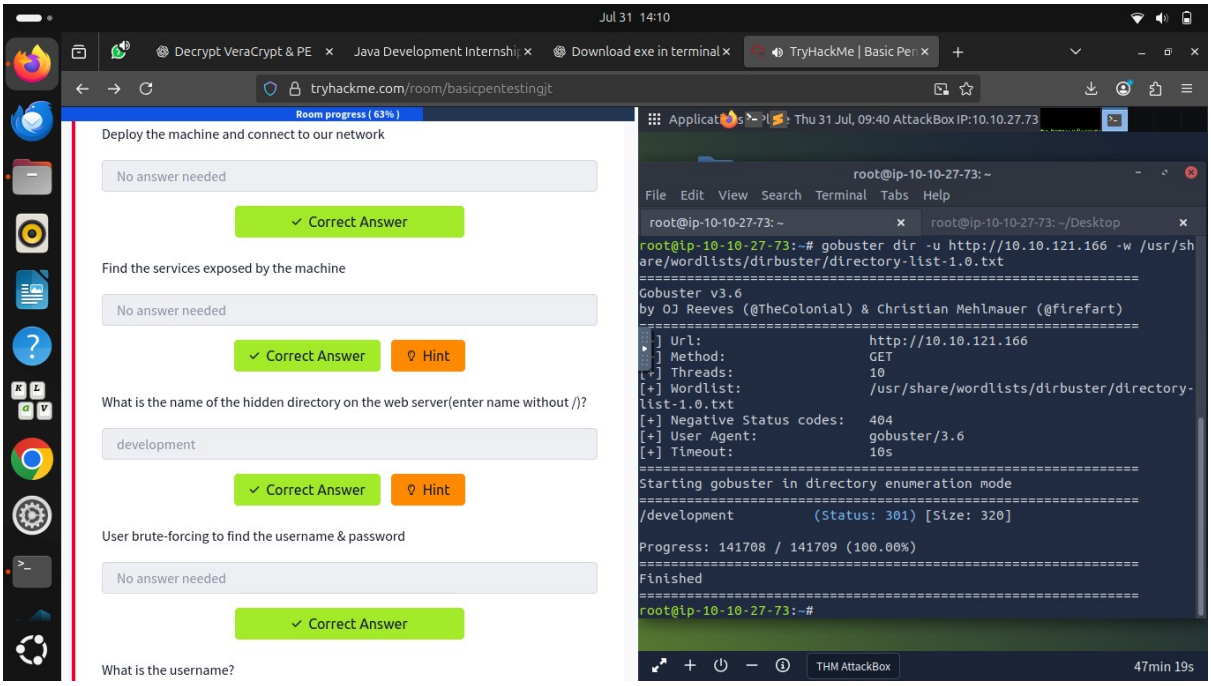
- **Attack Name**
  - ➢ *Penetration Test on the Basic Pentesting Room*
- **Severity**
  - ➢ *Level: High*
- **Impact**
  1. Unauthorized remote shell access to the target machine.
  2. Disclosure of sensitive user information via SMB shares.
  3. Exposure of login portal and internal files.
  4. Full root access gained.
- **Steps to reproduce**
  1. Checking whether target is alive or not



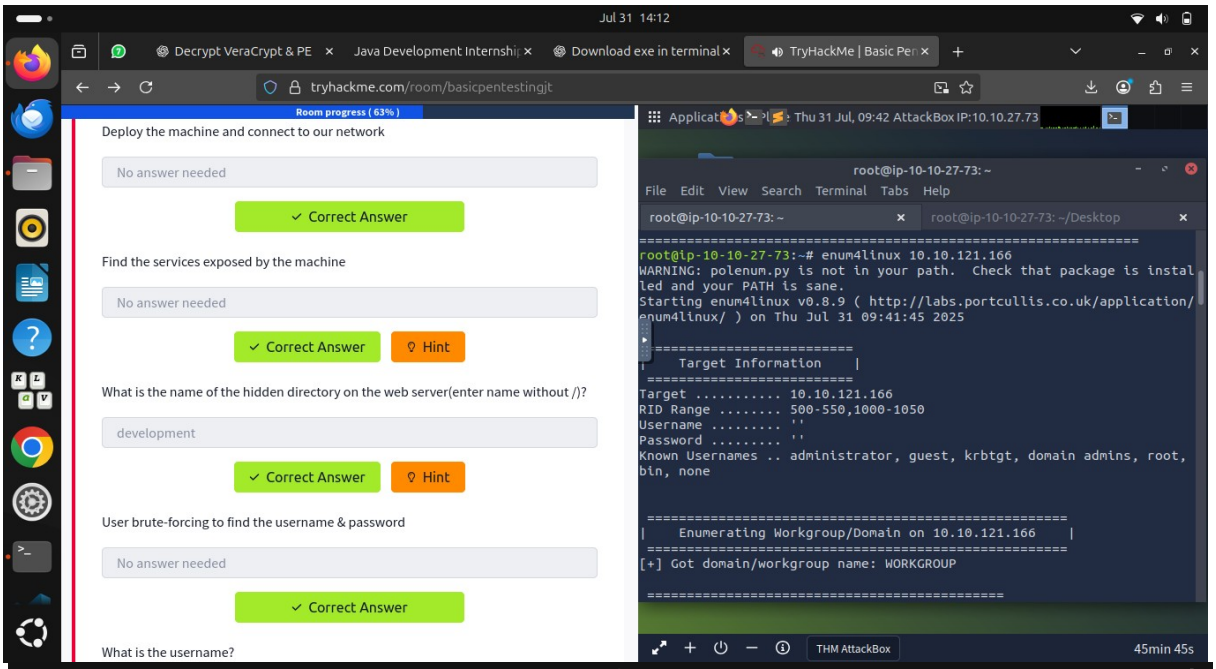
2. Scanning for open ports & services

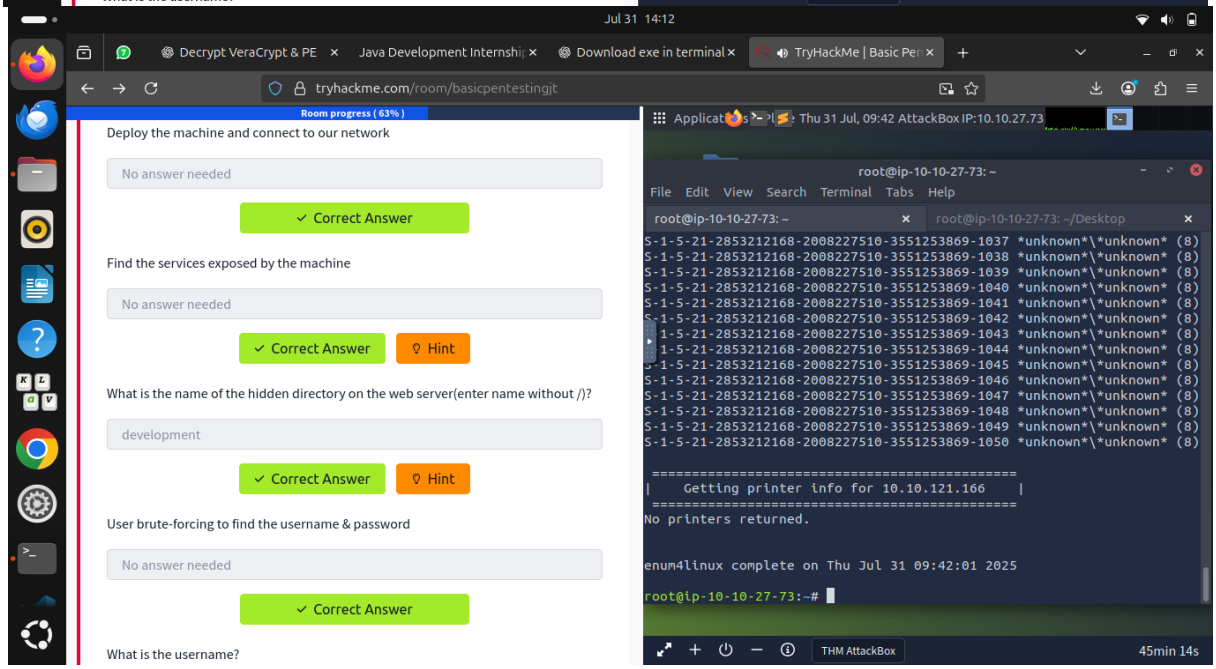


3. Directory fuzzing to find hidden directories



4. Username & password Bruteforcing using enum4linux





5. Logging in to user using SSH service

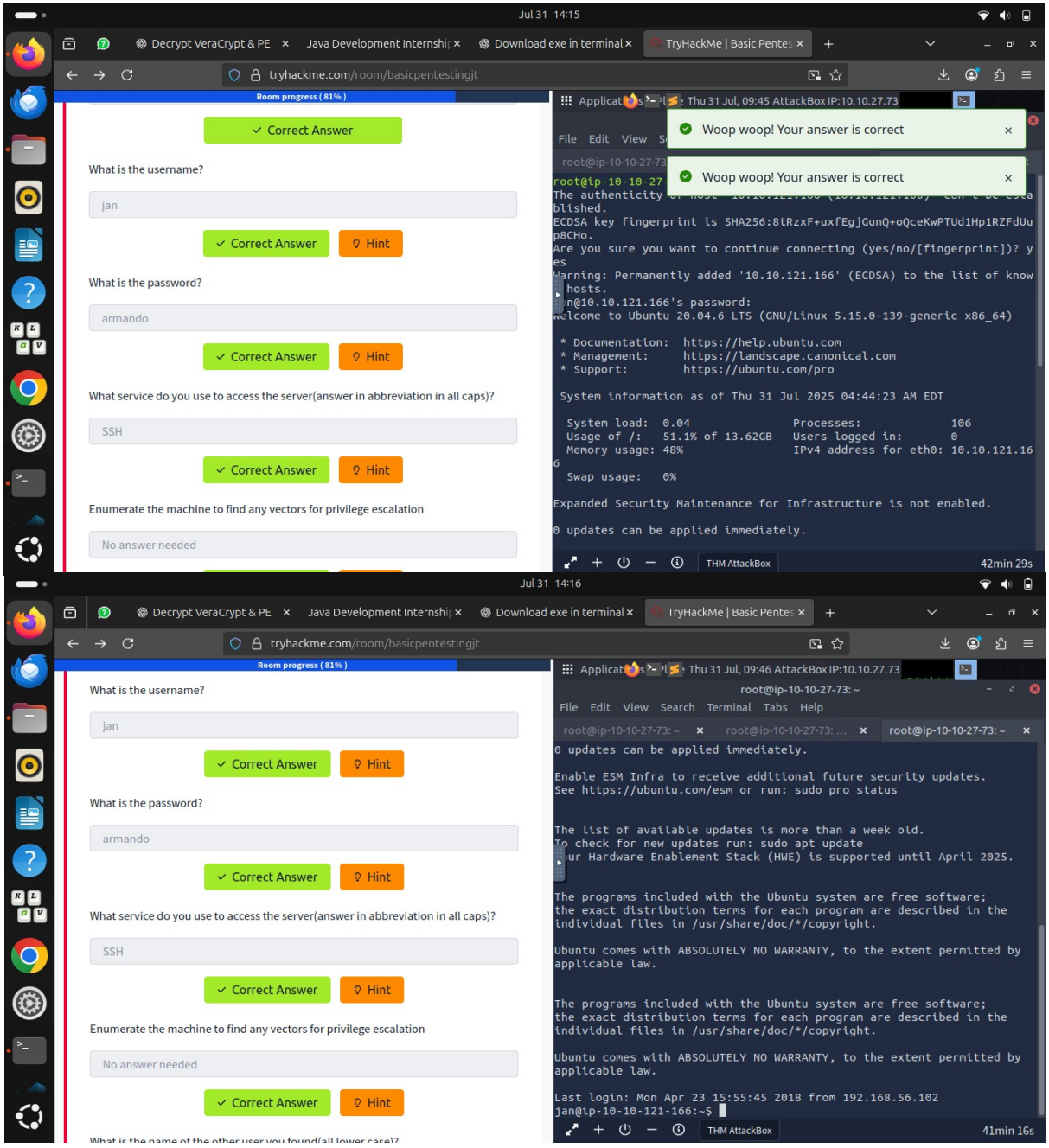
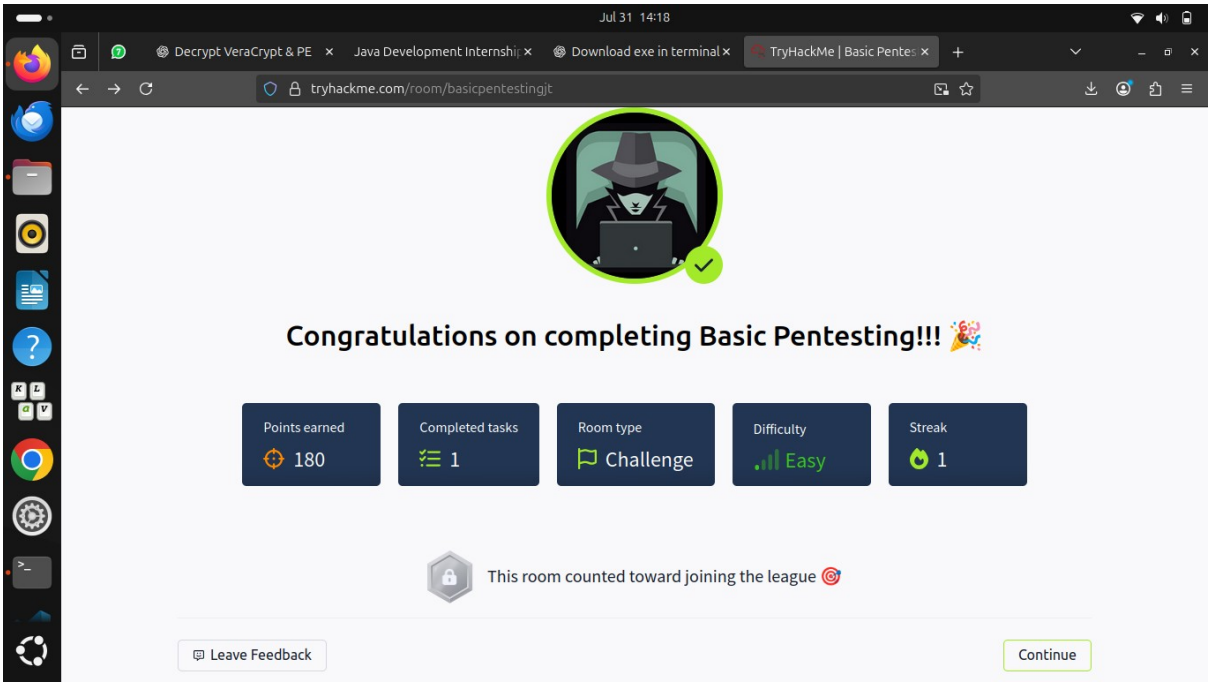


Figure 29 - SSH Login



- **Mitigation Steps**
    1. Implement account lockout policy for repeated failed login attempts.
    2. Disable SSH for default or weak accounts.
    3. Enforce strong password policies and use key-based authentication.
    4. Implement account lockout policy for repeated failed login attempts.
    5. Disable SSH for default or weak accounts.
    6. Enforce strong password policies and use key-based authentication.

END OF THE REPORT