Vulnerability Assessment Report

Mr.Robot

Issued on: 16/02/2025

Issues by: Mohamed Akhil

(ADCD)

TABLE OF CONTENTS

Confidentiality Notice

This report contains sensitive, privileged, and confidential information. Precautions should be taken to protect the confidentiality of the information contained in this document. Unauthorized disclosure, distribution, or reproduction of this report is strictly prohibited. Publication of this report may facilitate attacks against the Mr. Robot TryHackMe machine or expose vulnerabilities that could be exploited by malicious actors. The author shall not be held liable for any damages, including but not limited to reputational harm, financial loss, or security breaches resulting from the misuse of the information in this document.

Disclaimer

This penetration testing report is a point-in-time assessment of the Mr. Robot TryHackMe machine and may not uncover all vulnerabilities present within the system. The findings and recommendations provided are based on the system's state at the time of testing. Any changes made to the environment after the assessment may affect the validity of the results. This report is for educational and security research purposes only. The author assumes no responsibility for unauthorized use or misuse of this information.

EXECUTIVE SUMMARY

A penetration test was conducted on the Mr. Robot TryHackMe machine to evaluate its security vulnerabilities and assess potential risks. The objective was to simulate an external attack, starting from reconnaissance and exploitation to privilege escalation. The test focused on identifying misconfigurations, weak authentication mechanisms, and exploitable services that could lead to system compromise.

During the assessment, various vulnerabilities were identified, including exposed sensitive files, weak credentials, and privilege escalation via misconfigured binaries. The testing process involved discovering hidden files, exploiting a vulnerable WordPress installation, and leveraging an outdated Nmap version to achieve root access. This report provides detailed findings and methodologies used during the engagement which is broken down by severity in the table below:

CRITICAL	HIGH	MEDIUM	LOW
1	2	2	2

SCOPE

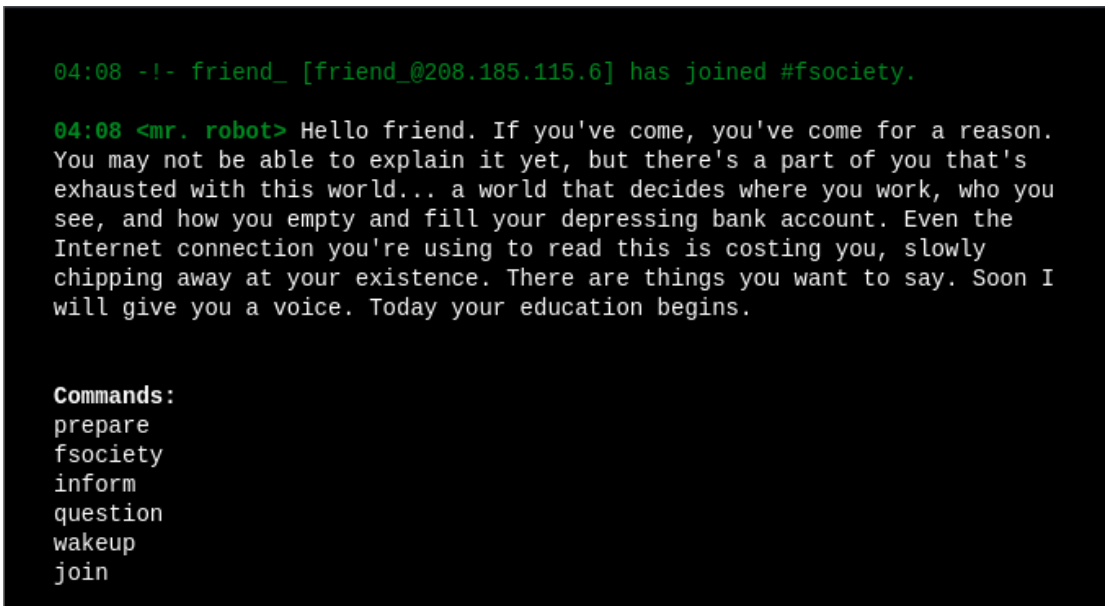
Objective:

The goal of this penetration test is to identify security vulnerabilities in the Mr. Robot TryHackMe machine, assess their impact, and demonstrate exploitation techniques leading to system compromise. The test simulates an external attacker's approach to gaining unauthorized access and escalating privileges to root.

In-Scope Components

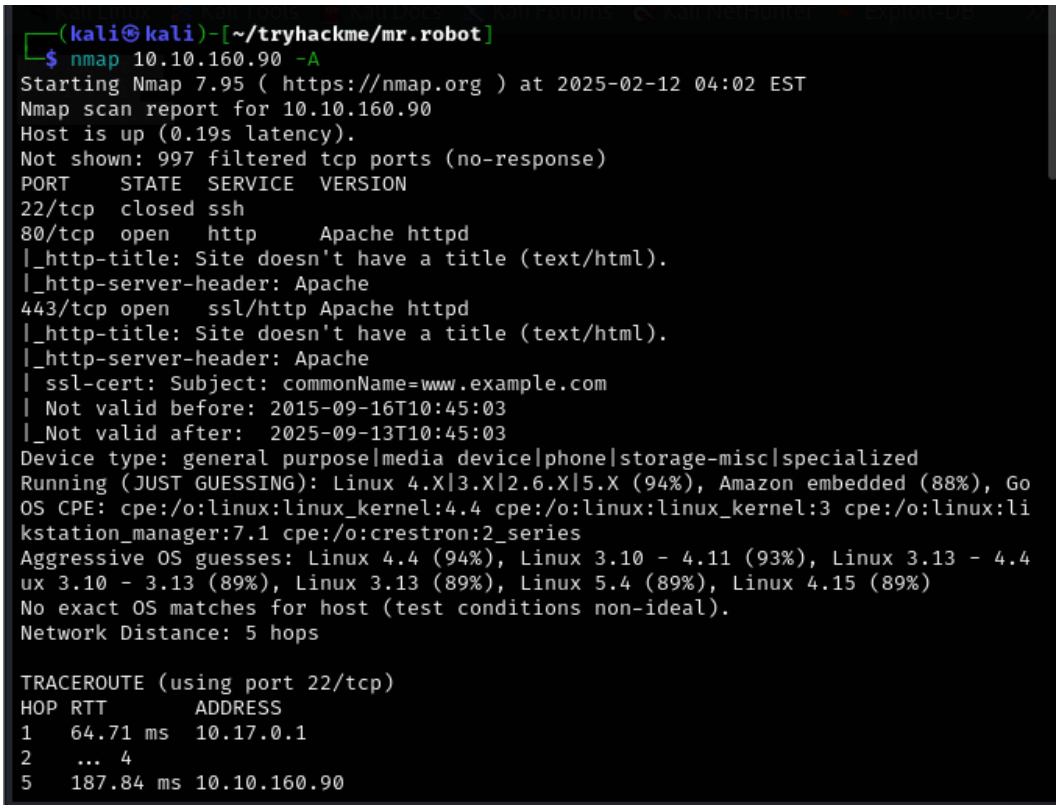
- Target System: **10.10.205.238** (Mr. Robot CTF Machine)
- Network Services:
  Web Server (Port 80, 443)
  SSH (Port 22 - Closed)
- Web Application: WordPress installation running on the target system
- User Privilege Levels Tested:
  Anonymous User (External attacker)
  Authenticated User (Robot user after credential discovery)
  Root User (Privilege escalation)

TESTING METHODOLOGY



Conducted an Nmap scan to identify open ports, running services, and potential vulnerabilities using the following command

Nmap 10.10.160.90 -A

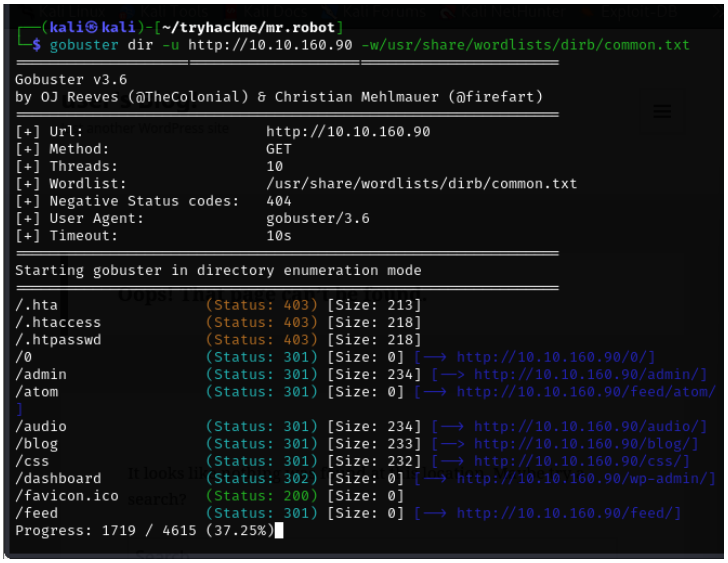


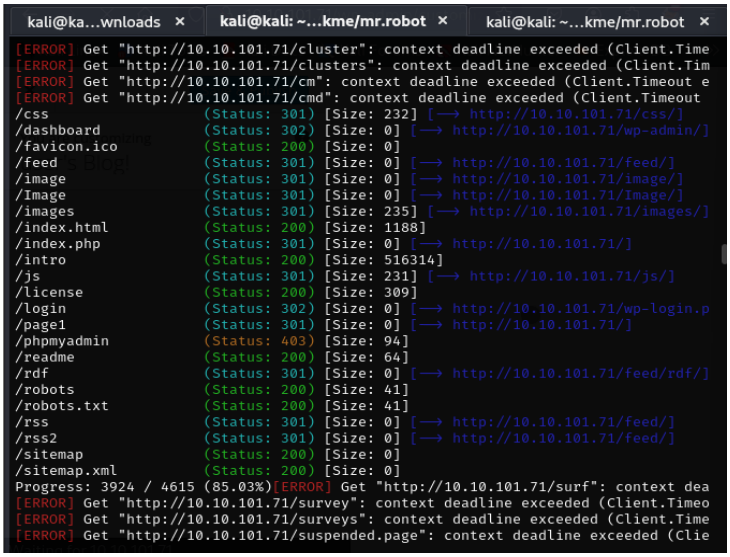
Discovered that HTTP service (port 80) was open from the Nmap scan.

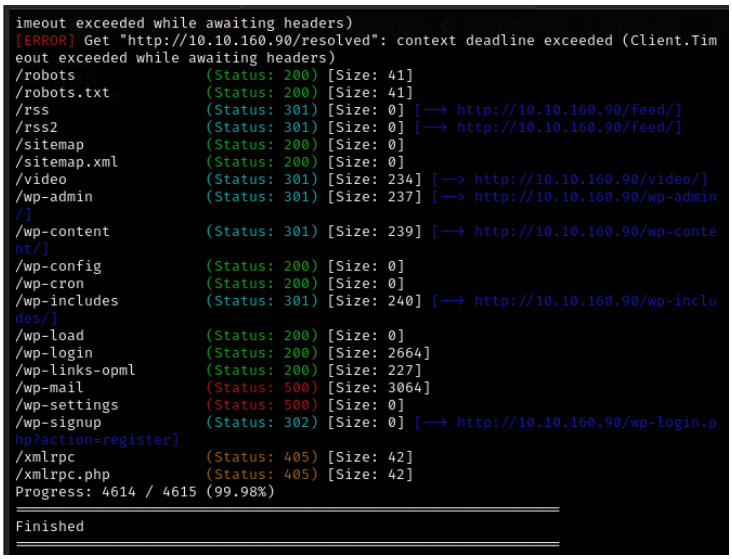
Used Gobuster to enumerate directories and hidden files using the following command:

gobuster dir -u http://10.10.160.90 -w /usr/share/wordlists/dirb/common.txt

Identified potential areas of interest for further exploitation.





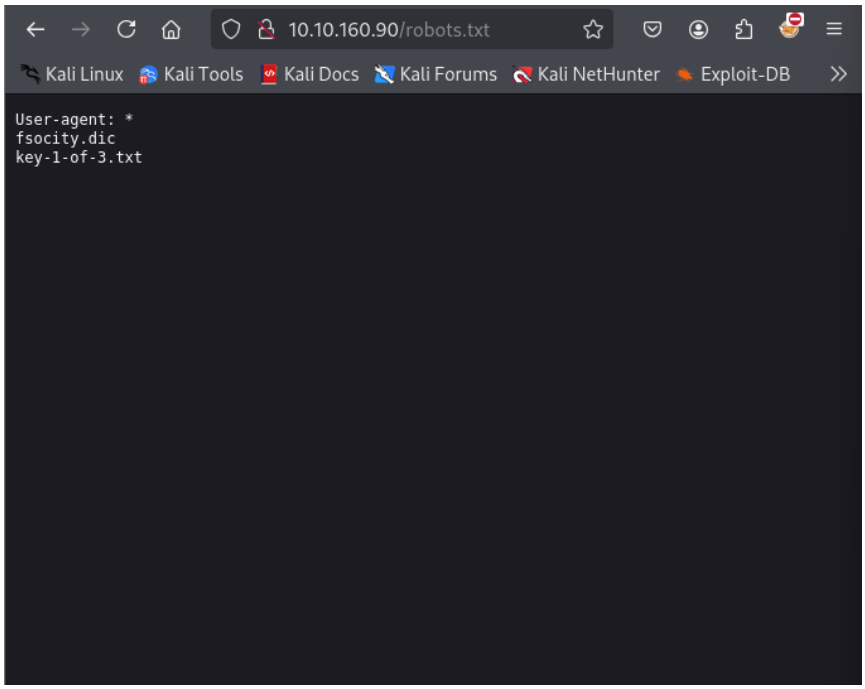


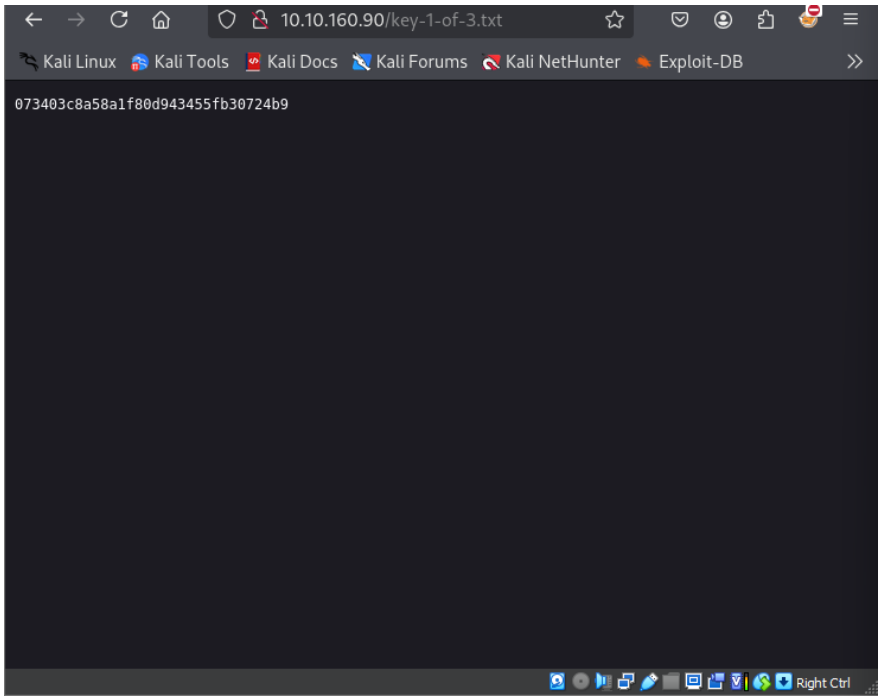
After identifying open directories, accessed robots.txt at:

http://10.10.160.90/robots.txt

Discovered two files:

- fsociety.dic – A dictionary file (potentially useful for password cracking)
- **key-1-of-3.txt** – Contained the first key for the challenge

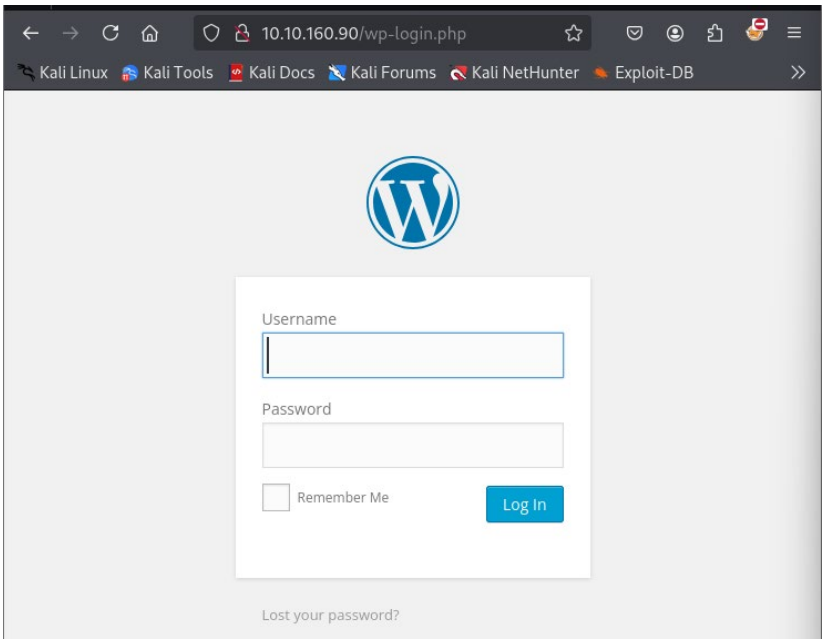




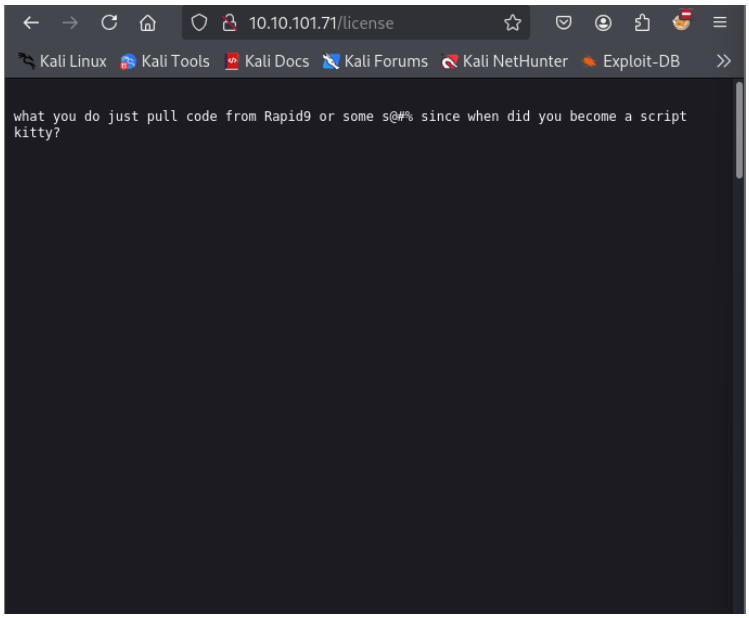
During directory enumeration, discovered a WordPress login page at:

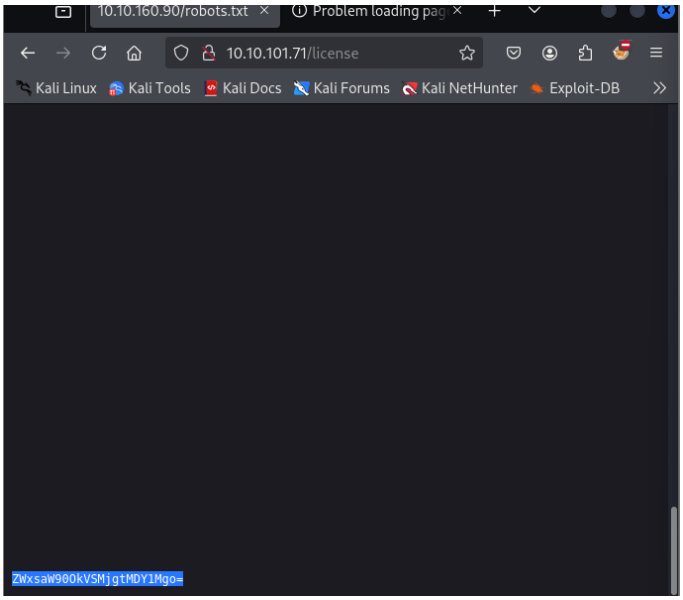
http://10.10.160.90/wp-login.php

This indicates the target is running a WordPress CMS, which may have potential vulnerabilities.



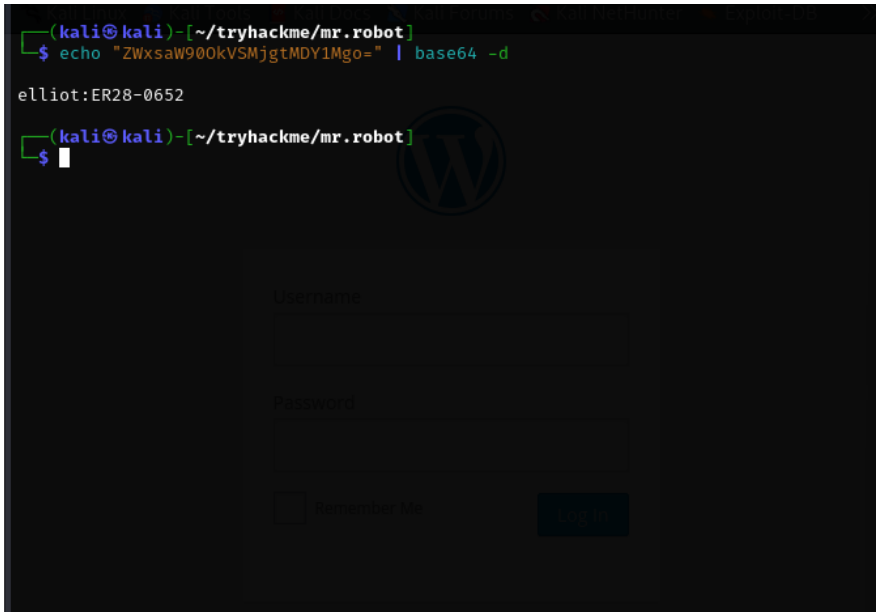
During enumeration, we discovered the **/license** directory containing an encoded string.





The string was decoded using Base64, revealing the following credentials:

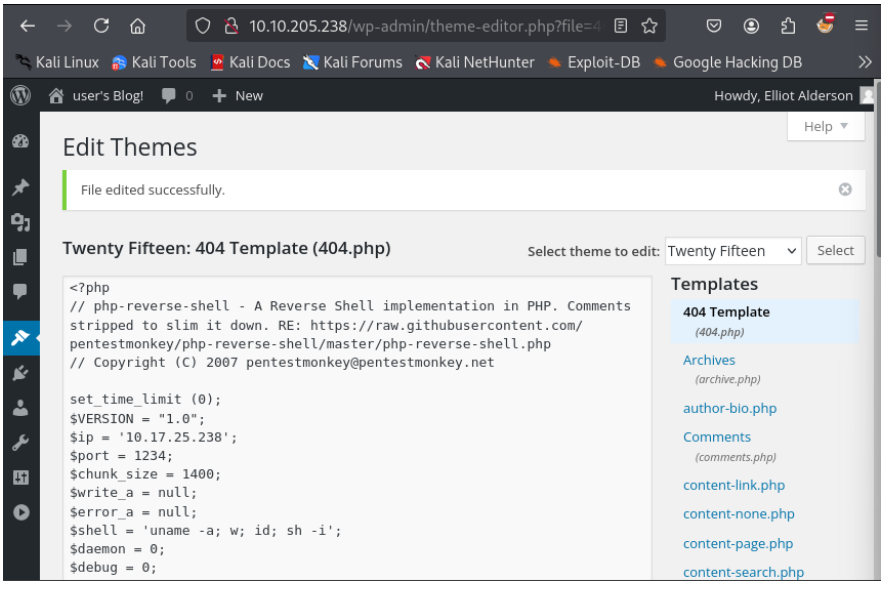
Elliot:ER28-0652



Using these credentials, we successfully logged into the WordPress admin panel.

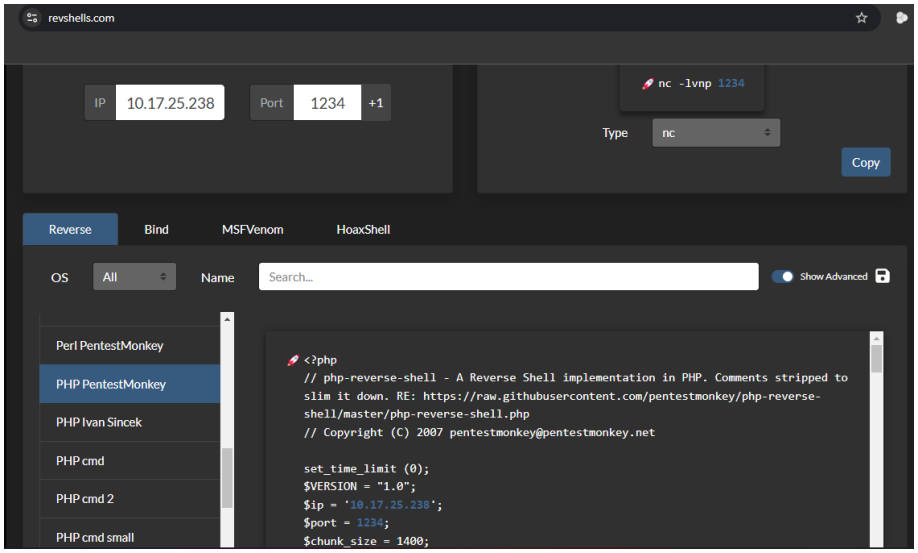
After successfully logging into the WordPress admin panel using the credentials obtained in the previous step, we proceeded with setting up a reverse shell.

1. **Navigating to the Theme Editor:**
    - Within the WordPress dashboard, we accessed the **Appearance > Theme Editor** section.
    - From the available theme files, we selected the **404.php** template file.



2. **Injecting Reverse Shell Code:**

   - We retrieved a **PHP reverse shell script** from revshells.com (specifically the PHP PentestMonkey reverse shell).

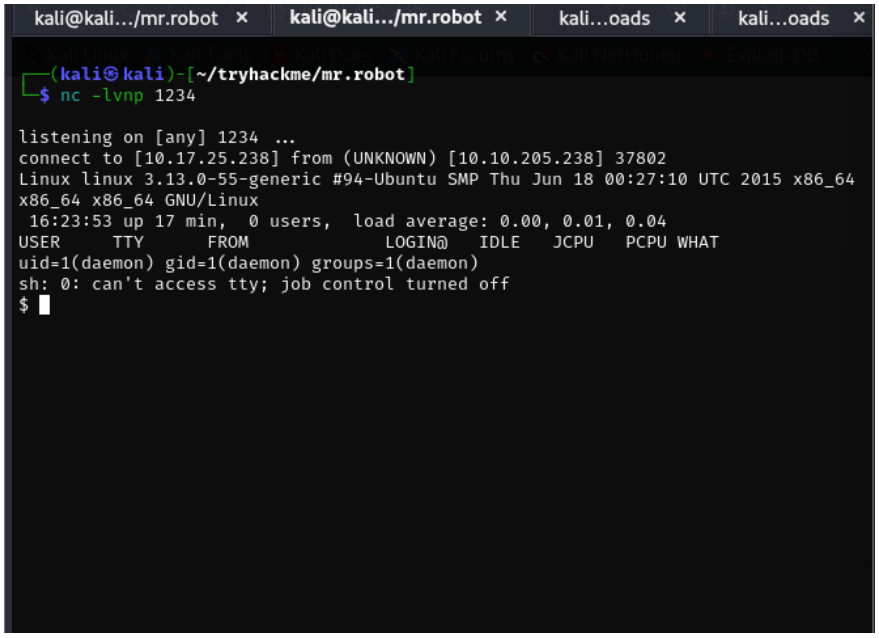


   - We modified the script by replacing the IP and port with our attacker's machine IP and a listening port.
   - The modified shell script was then inserted into the **404.php** template file.

This setup ensured that whenever the 404 error page was accessed, the PHP code would execute, establishing a connection back to our attacking machine.

To catch the reverse shell connection, we started a Netcat listener on our attack machine. We executed the following command:

nc -lvnp 1234



After successfully gaining a shell session on the target machine, we executed the following commands to gather system information and explore directories:

1. Check the current user

   whoami

Output: daemon (indicating a low-privileged user).

2. Verify the current working directory

   pwd

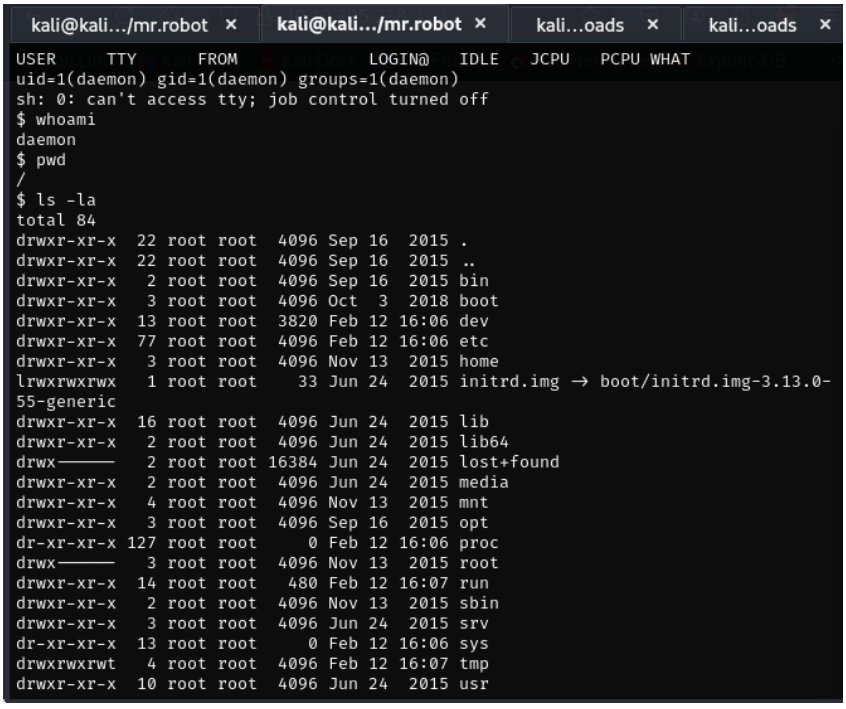
Output: Displays the present working directory.

3. List all files and directories in the root directory

   ls -la

This command provided detailed information about system directories, including permissions, ownership, and hidden files.

At this point, we identified key directories such as /home, /etc, /root, and /var, which could contain valuable information for privilege escalation.



After obtaining the initial shell as the "daemon" user, we explored the system and navigated to the /home directory. We found a user named **robot** and inside its home directory, we discovered two files:

1. password.raw-md5 - Containing an MD5 hashed password.
2. **key-2-of-3.txt** - A key file, but it was only readable by the robot user.

```
eric
$ cd home
$ ls -la
total 12
drwxr-xr-x  3 root root 4096 Nov 13  2015 .
drwxr-xr-x 22 root root 4096 Sep 16  2015 ..
drwxr-xr-x  2 root root 4096 Nov 13  2015 robot
$ cd robot
$ ls -la
total 16
drwxr-xr-x 2 root  root  4096 Nov 13  2015 .
drwxr-xr-x 3 root  root  4096 Nov 13  2015 ..
-r-------- 1 robot robot   33 Nov 13  2015 key-2-of-3.txt
-rw-r--r-- 1 robot robot   39 Nov 13  2015 password.raw-md5
$ cd password.raw-md5
sh: 8: cd: can't cd to password.raw-md5
$ cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
```

Since we needed access to the "robot" user, we extracted the MD5 hash from password.raw-md5 and cracked it using **John the Ripper** with the rockyou.txt wordlist.

The command used:

john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash2.txt

```
kali@kali.../mr.robot ×  kali@kali.../mr.robot ×  kali...oads ×  kali...oads ×

┌──(kali㉿kali)-[~/tryhackme/mr.robot]
└─$ nano hash2.txt

┌──(kali㉿kali)-[~/tryhackme/mr.robot]
└─$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash2.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
abcdefghijklmnopqrstuvwxyz (?)
1g 0:00:00:00 DONE (2025-02-12 11:49) 33.33g/s 1356Kp/s 1356Kc/s 1356KC/s bonjour
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords
Session completed.

┌──(kali㉿kali)-[~/tryhackme/mr.robot]
└─$
```

John the Ripper successfully cracked the hash and retrieved the **plaintext password**.

After successfully cracking the **MD5 password hash**, we attempted to switch to the **robot** user using:

su robot

However, this failed because our initial shell (**daemon**) was a **limited shell** and did not allow us to use the `su` command directly.

To overcome this restriction, we needed to **spawn an interactive shell** using **Python**. We executed:

python -c 'import pty; pty.spawn("/bin/bash")'

This command:

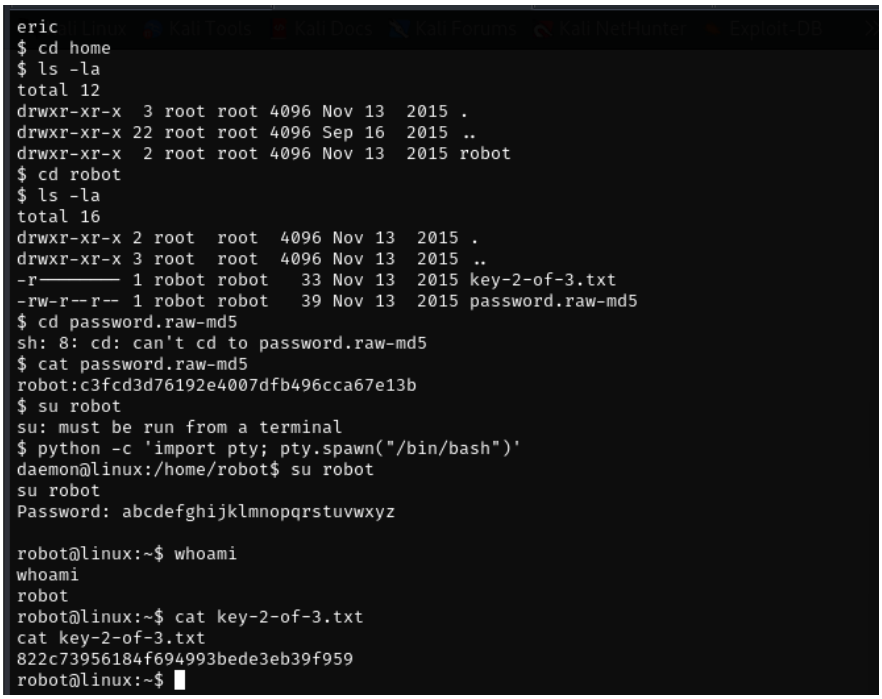
- Uses Python's pty module to spawn a pseudo-terminal (PTY).
- Upgrades our limited shell to a fully interactive shell, allowing us to use commands like su.

Once the interactive shell was spawned, we ran **su robot** again

We then entered the cracked password, successfully switching to the **robot** user.

Now that we have access to the robot user, we can read key-2-of-3.txt and continue with **privilege escalation to root**.

Key 2 : 822c73956184f694993bede3eb39f959

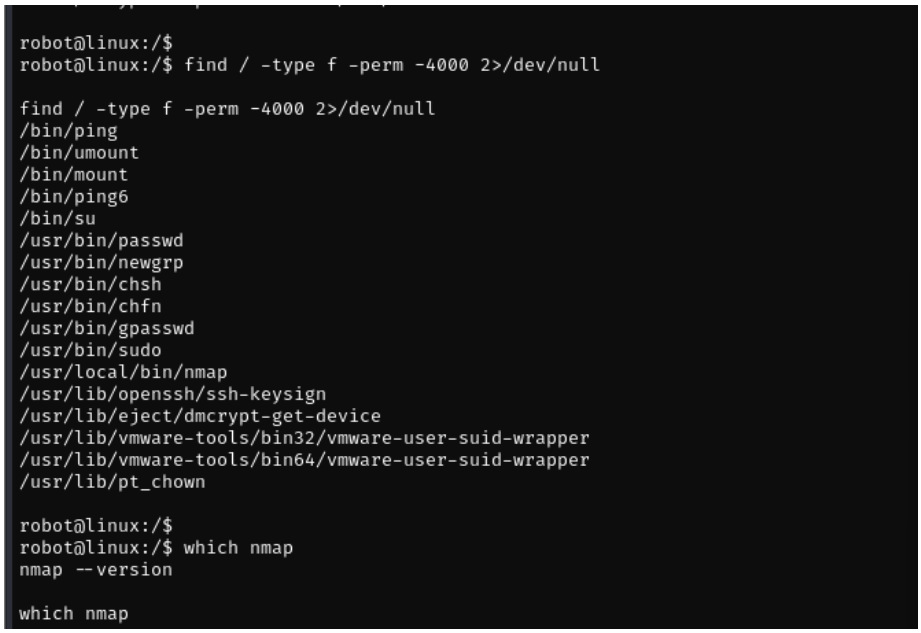


Exploiting **SUID Bit** for Privilege Escalation

Executed the following command to identify files with the SUID bit set:

find / -type f -perm -4000 2>/dev/null

This searches for files with SUID permissions, meaning they execute with the privileges of their owner (usually root).



From the output, we see that **/usr/local/bin/nmap** has the SUID bit set. This means we can potentially escalate privileges using it.

Checking the Nmap Version:

which nmap
nmap –version

The version displayed is `3.81`, which supports interactive mode, allowing us to execute shell commands

Enter Nmap Interactive Mode:

nmap –interactive

This starts an interactive Nmap shell where we can run commands.

Gain a Root Shell

In the interactive prompt, execute:

!sh

The !sh command spawns a shell, and since nmap is running with SUID root permissions, the shell will also have root privileges.

Verify Root Access by running:

Whoami

Navigate to the root directory and retrieve the flag:

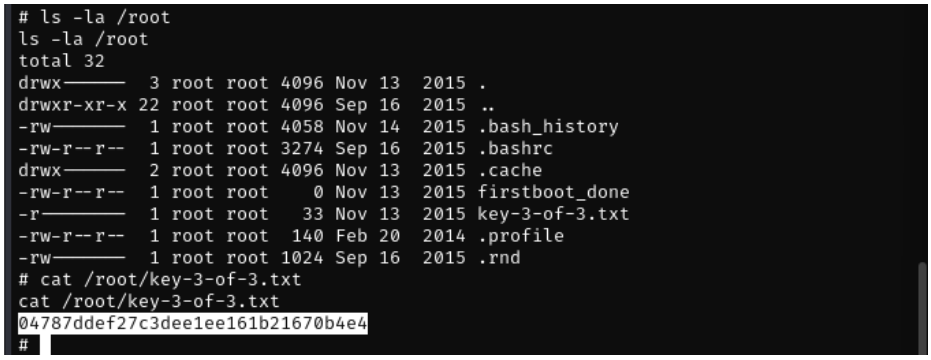
ls -la /root

In the root directory we found out the **key-3-of-3.txt** file which contained key 3

cat /root/key.txt

This should reveal the final flag, completing the privilege escalation

Key 3: 04787ddef27c3dee1ee161b21670b4e4



CRITICAL SEVERITY VULNERABILITY

Here is a simple severity table summarizing the key findings from Mr. Robot CTF penetration test:

Severity	Vulnerability	Impact
Critical	Privilege Escalation via SUID Nmap	Full system compromise (Root access)
High	WordPress Reverse Shell Exploitation	Remote command execution
High	Weak Authentication (Cracked Password)	Unauthorized access to the system
Medium	Sensitive Information in robots.txt	Leakage of internal file paths
Medium	MD5 Password Hash Storage	Weak encryption, easy to crack
Low	Outdated Software (Nmap v3.81)	Potential for exploitation, should be updated
low	Default WordPress Login Page Exposed	Can help attackers identify login targets

RECOMMENDATIONS

To secure a system against the vulnerabilities exploited in this challenge, the following measures should be implemented:

1. Protect Against Weak Credentials

- Enforce strong password policies, requiring a mix of uppercase, lowercase, numbers, and special characters.
- Implement multi-factor authentication (MFA) to reduce the risk of credential-based attacks.
- Regularly rotate passwords and avoid using default credentials.
- Use tools like fail2ban to block brute-force attacks.

2. Secure File Permissions

- Restrict file and directory permissions to follow the principle of least privilege.
- Run the following command to check for files with SUID permissions and remove unnecessary ones:

  find / -perm -4000 -type f 2>/dev/null

- Remove the SUID bit from binaries that do not require it:

  chmod -s /path/to/binary

3. Patch and Update Services

- Keep all system packages and applications up to date with the latest security patches.
- Regularly update web applications (like WordPress) to prevent exploitation of known vulnerabilities.
- Disable or remove unused services to minimize the attack surface.

4. Prevent Privilege Escalation via SUID Binaries

- Audit all SUID binaries
- Remove the SUID permission from unnecessary binaries
- If a binary like nmap is required, ensure it is updated to a version that does not allow interactive mode for privilege escalation.

5. Web Application Hardening

- Restrict access to sensitive files such as robots.txt, .htaccess, and backup files.
- Disable directory listing in the web server configuration.
- Implement security headers such as:

  Header always set X-Frame-Options "DENY"
  Header always set X-XSS-Protection "1; mode=block"
  Header always set X-Content-Type-Options "nosniff"

6. Implement Intrusion Detection & Monitoring

- Use Intrusion Detection Systems (IDS) like Snort or Suricata to monitor suspicious activity.
- Implement logging and monitoring using auditd and SIEM solutions to detect unauthorized access attempts.

7. Restrict User Privileges

- Ensure that users do not have unnecessary sudo privileges.
- Limit shell access for non-administrative users.
- Use tools like AppArmor or SELinux to restrict what processes can execute.

By applying these security recommendations, an organization can significantly reduce the risk of exploitation and improve overall system security.

CONCLUSION

The penetration test conducted on the **Mr. Robot CTF** machine demonstrated several critical vulnerabilities that could lead to full system compromise. By leveraging misconfigurations, weak credentials, and SUID binaries, we successfully escalated privileges to root and captured all three keys.

This assessment highlights the importance of implementing strong security measures, including enforcing strict password policies, regularly updating software, restricting file permissions, and minimizing the use of SUID binaries. Additionally, monitoring and intrusion detection systems should be deployed to detect and prevent unauthorized activities.

By addressing the vulnerabilities identified in this penetration test and applying the recommended security best practices, organizations can significantly reduce their attack surface and improve overall system resilience against cyber threats.