# Mr. Robot Penetration Test Report

Abinesh S

Date Completed:

12/02/2025

# Table of Contents

# EXECUTIVE SUMMARY

The objective of this penetration testing endeavor was to evaluate the security posture of the Mr. Robot machine hosted on a local network with the IP address 192.168.29.64. Beginning with an initial enumeration phase, the IP was identified using netdiscover, setting the stage for subsequent reconnaissance and exploitation.

Service enumeration revealed several open ports, including SSH (22) and HTTP (80), forming the foundation for further analysis. A thorough examination of the web application followed, leading to the discovery of critical vulnerabilities via manual inspection and tools such as Dirb and Burpsuite.

Credential enumeration played a pivotal role, with Hydra utilized to exhaustively probe for valid usernames and passwords. Successful acquisition of login credentials provided entry into the WordPress web application, which was subsequently exploited to gain initial access using a PHP reverse shell.

Privilege escalation tactics were employed, leveraging Linpeas for system enumeration. Ultimately, root access was achieved by exploiting a SUID misconfiguration on Nmap, revealing crucial system configurations and sensitive data.

The findings underscore the imperative for enhanced security measures within the Mr. Robot machine infrastructure, emphasizing the importance of robust password policies and routine application updates. This report presents detailed recommendations aimed at fortifying the security posture of the Mr. Robot machine, thereby mitigating potential vulnerabilities and enhancing resilience against cyber threats.

# TARGET INFORMATION

Target Name: Mr. Robot Machine

Platform: VulnHub

Target IP: 192.168.29.64

Objective: Locate and capture all three hidden keys

Environment:

- The Mr. Robot VM emulates a realistic environment inspired by the TV show “Mr. Robot,” designed to test ethical hacking skills by presenting real-world vulnerabilities.
- Hosted on VirtualBox, the VM is purposefully configured with misconfigurations and weaknesses, making it ideal for penetration testing practice.

# TOOLS USED

- Kali Linux: A Debian-based penetration testing distribution used as the primary operating system for this engagement. It provides a wide range of security tools for reconnaissance, exploitation, and post-exploitation.
- VulnHub: A platform hosting vulnerable virtual machines for cybersecurity practice. The Mr. Robot machine was downloaded from VulnHub for this test.
- VirtualBox: An open-source hypervisor used to host and run the Mr. Robot VM in an isolated environment.
- Nikto: A web server scanner used to identify potential vulnerabilities, misconfigurations, and outdated software on the web application hosted by the Mr. Robot machine.
- Nmap: A powerful network scanning tool employed for host discovery, service enumeration, and version detection.
- Burpsuite: An integrated platform for performing web application security testing, used to intercept HTTP requests, analyze responses, and manipulate web traffic.
- Hydra: A fast and flexible login cracker used to perform brute-force attacks on the WordPress login page to discover valid credentials.
- Metasploit: An advanced exploitation framework used for validating vulnerabilities and gaining initial access.
- Python: Utilized for executing shell commands and enhancing the functionality of the reverse shell.
- CrackStation: An online hash cracking tool used to decrypt password hashes obtained during post-exploitation.

# Methodology

The testing approach followed industry-standard penetration testing methodologies, including:

- Reconnaissance and Information Gathering
- Enumeration and Scanning
- Exploitation and Gaining Access
- Privilege Escalation
- Post-Exploitation and Cleanup

## Lab Environment Configuration

Provisioned VirtualBox and configured the KaliLinux and Mr.Robot images. Separated both virtual machines onto an individual subnet for isolation. Executed connectivity validation through ping tests between the two virtual machines.

## Initial Enumeration

1. Initiated the reconnaissance phase by identifying the target IP using netdiscover.
2. Conducted an Nmap scan with aggressive mode on 192.168.29.64 to identify open ports and running services.
3. Discovered open ports: 22 (SSH) and 80 (HTTP).
4. Accessed the web application at http://192.168.29.64 and performed manual exploration.
5. Checked http://192.168.29.64/robots.txt and discovered the first key.
6. Found a wordlist file named fsocity.dic in the robots.txt directory.
7. Processed fsocity.dic to remove duplicate entries and saved it as sorted.dic for future use.
8. Discovered a hidden WordPress login page and captured the POST request using Burpsuite.
9. Used Hydra with sorted.dic to perform brute-force attacks, revealing:
10. Username: Elliot
11. Password: ER28-0652
12. Logged into the WordPress admin panel and used Metasploit to search for a WordPress shell, successfully gaining a reverse shell.
13. Upgraded the shell using the command:
14. python -c 'import pty; pty.spawn("/bin/bash")'

15. Identified a SUID misconfiguration on Nmap using:
    find / -perm /4000 -type f 2>/tmp/2
16. Exploited the SUID Nmap to gain a root shell using the command:
    nmap --interactive
    !sh
17. Achieved root access and navigated to the /root/ directory to find the third key (key-3-of-3.txt).
18. Successfully captured all three keys, completing the objective of the penetration test.

## Penetration Testing Approach

The testing approach was structured around widely accepted penetration testing methodologies, encompassing the following phases:

- Reconnaissance and Information Gathering: Collecting initial information about the target to identify potential attack vectors.
- Enumeration and Scanning: Actively probing the target to discover open ports, running services, and exploitable vulnerabilities.
- Exploitation and Gaining Access: Utilizing identified vulnerabilities to penetrate the system and establish a foothold.
- Privilege Escalation: Elevating access rights to gain administrative or root-level control over the target.
- Post-Exploitation and Cleanup: Extracting valuable information, maintaining persistence if needed, and securely cleaning traces to maintain operational integrity.

# APPENDICES

## Appendix A: Visual Documentation

I have two virtual machines set up for this exercise: Kali Linux and Mr. Robot. Both VMs are connected to a host-only network with DHCP enabled, ensuring they can communicate with each other while remaining isolated from external networks.

To enhance security, I configured a firewall rule on my host machine to disallow any inbound or outbound traffic from this network. This setup not only eliminates potential noise from the host but also safeguards it from any activities occurring within the VMs.

For this test, I first booted up the Mr. Robot VM to initialize its services and network configurations. Following that, I launched the Kali Linux VM, ensuring an optimal sequence for reconnaissance and exploitation activities.

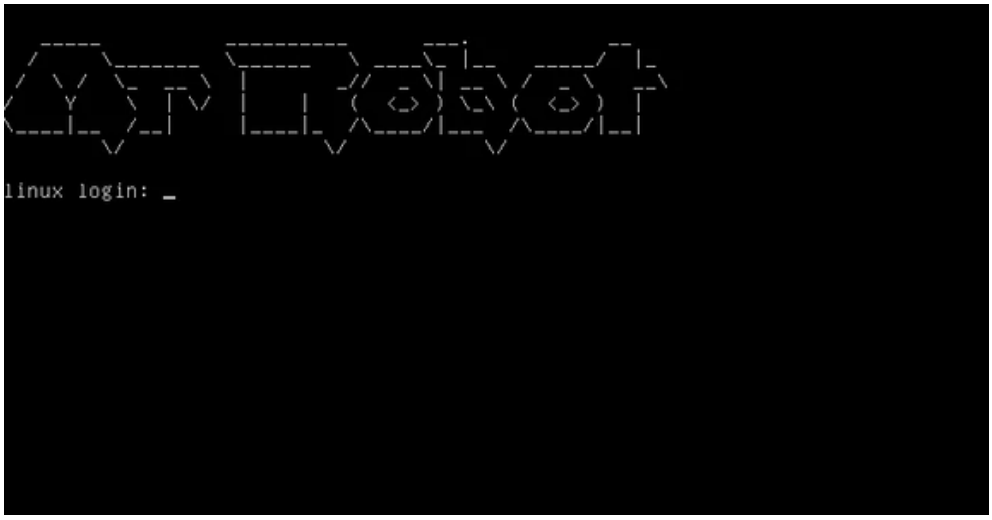



Figure 1: Mr Robot Initial Webpage

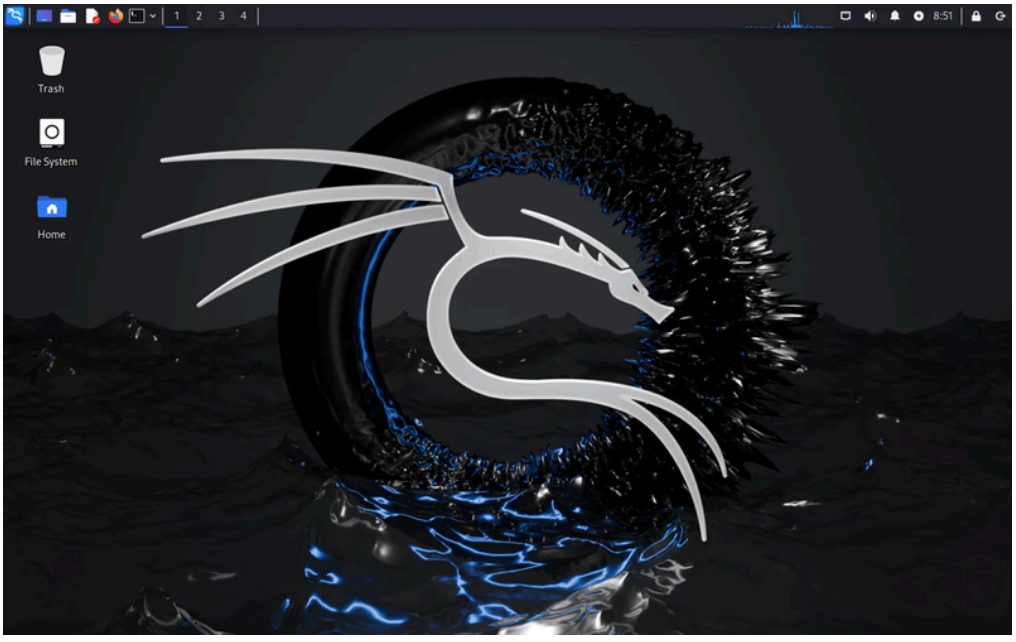
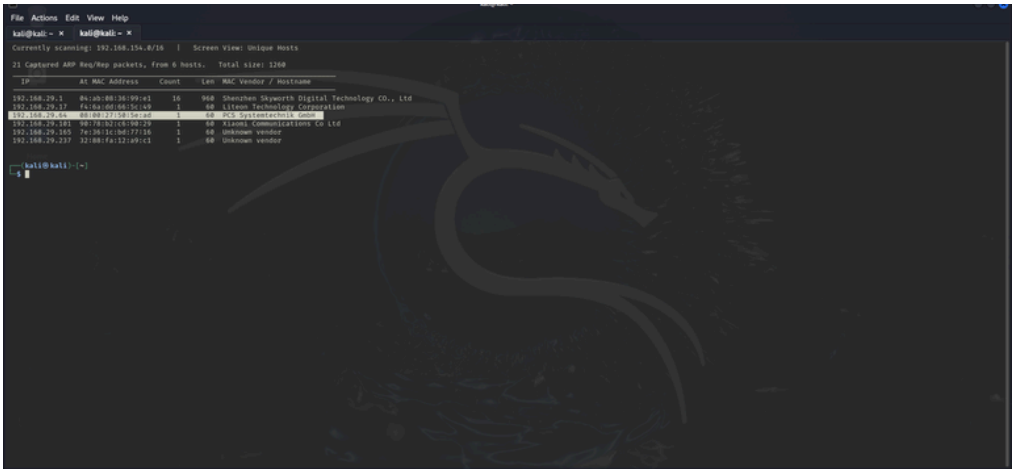



Figure 1: kali Linux Initial Webpage

## Reconnaissance

To initiate the assessment, it was essential to identify the target's IP address within the isolated network. The Kali Linux VM was assigned the IP address 192.168.29.178 by the DHCP server. Using the tool netdiscover, a passive network scanning utility, I conducted a sweep of the subnet to identify all active hosts. This scan successfully revealed the target IP address: 192.168.29.64, confirming the presence of the Mr. Robot VM on the network.




The Nmap scan revealed the presence of two open ports on the target machine:

- Port 80 (HTTP) – Indicating that a web server is accessible.
- Port 443 (HTTPS) – Suggesting the availability of a secure web service.

Additionally, Port 22 (SSH) was identified as closed, effectively eliminating the possibility of gaining initial access through SSH.

The presence of open HTTP and HTTPS ports suggests that the target is likely hosting a web application, providing a potential attack surface for further enumeration and exploitation.

To investigate the web application hosted on the target, I launched a browser and navigated to The Nmap scan revealed the presence of two open ports on the target machine:

- Port 80 (HTTP) – Indicating that a web server is accessible.
- Port 443 (HTTPS) – Suggesting the availability of a secure web service.

Additionally, Port 22 (SSH) was identified as closed, effectively eliminating the possibility of gaining initial access through SSH.

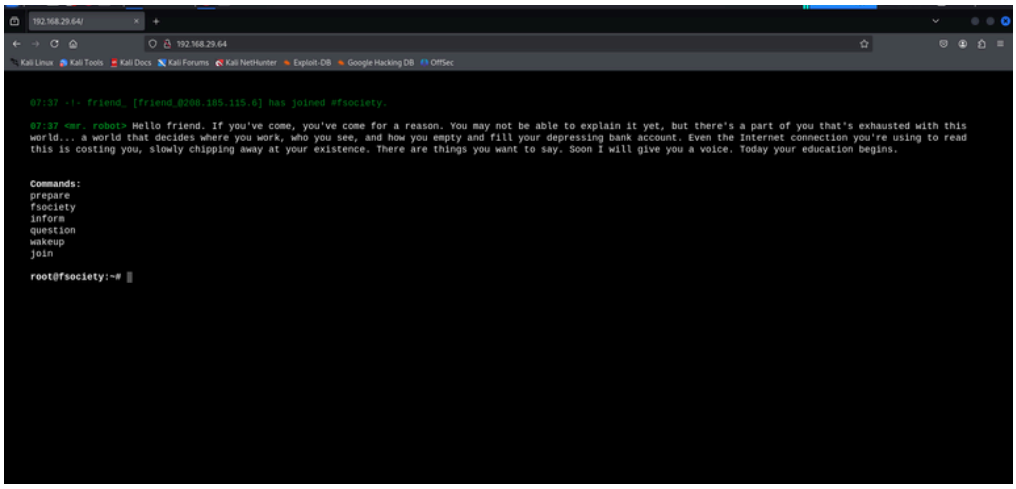
The presence of open HTTP and HTTPS ports suggests that the target is likely hosting a web application, providing a potential attack surface for further enumeration and exploitation.

. As anticipated, the page loaded successfully, revealing an animation themed around the TV series Mr. Robot.

Upon completion of the animation, the page transitioned to an interactive screen resembling a Linux terminal with a blinking cursor. A list of commands was displayed, suggesting user interaction.

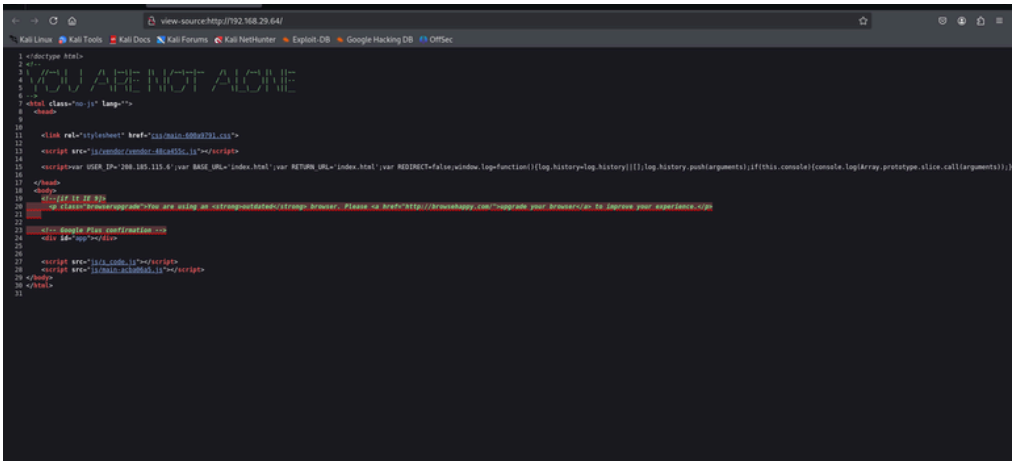
Exploring the interactive terminal, I entered various commands, each triggering a new animation or displaying a gallery of images. Regardless of the input, the session consistently returned to the terminal-like interface.

Repeating the process using the HTTPS protocol (https://192.168.29.64) yielded identical results, confirming that both HTTP and HTTPS traffic was being directed to the same web application.



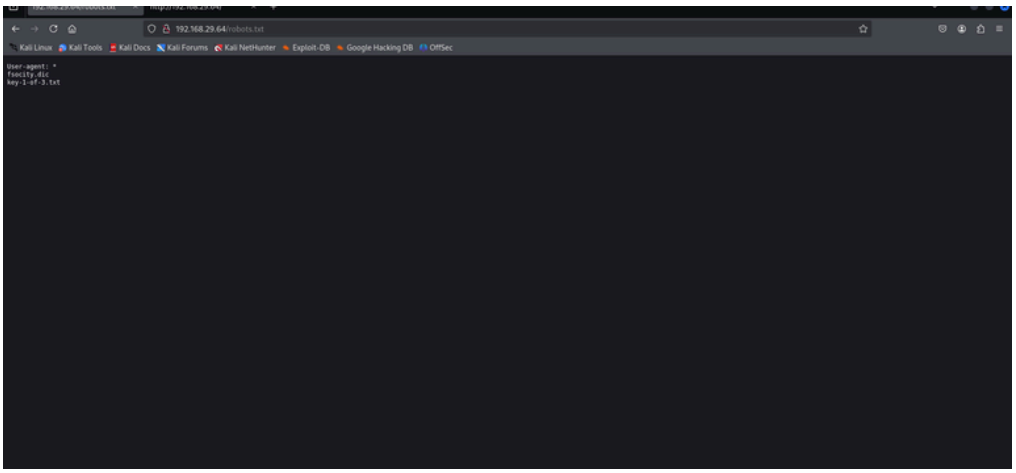

Checking the source provides no additional clues. The js scripts are a dead-end too.

Checking the source provides no additional clues. The js scripts are a dead-end too.

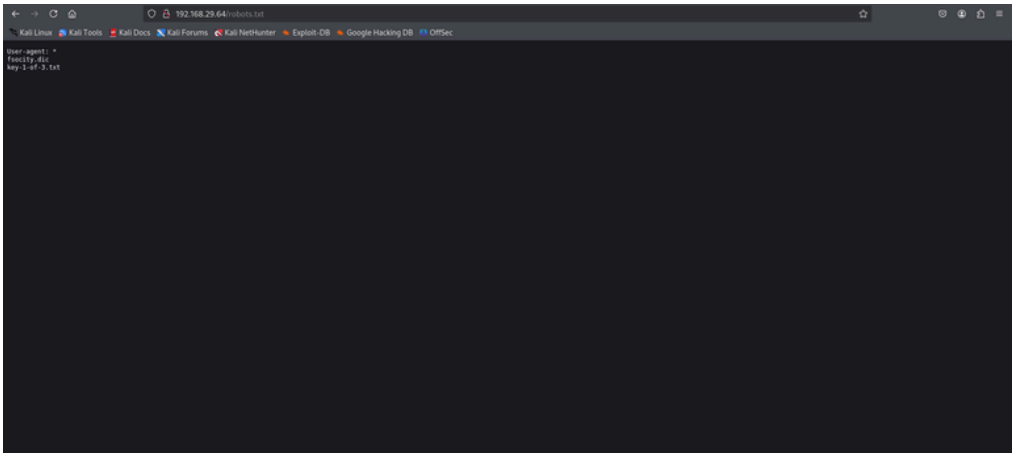


## Web Reconnaissance

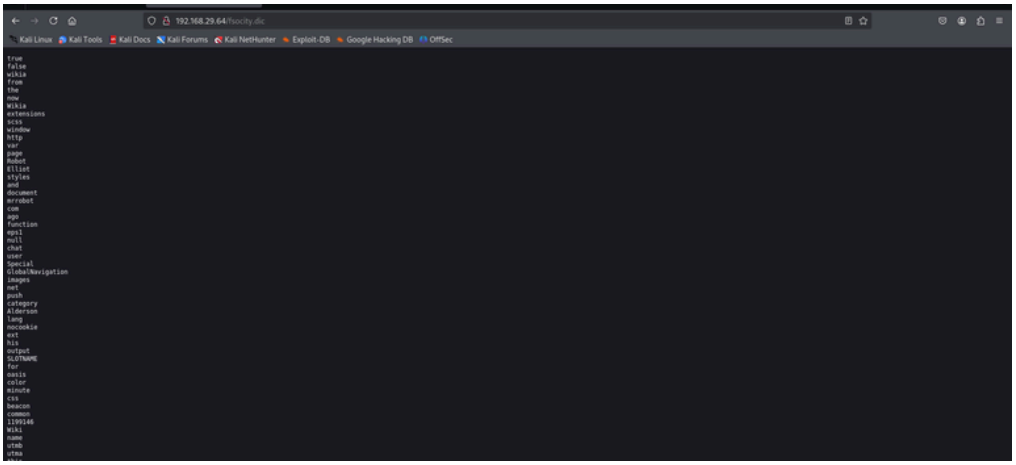
Before proceeding with directory enumeration, I examined the robots.txt file by navigating to:
**http://192.168.29.64/robots.txt**
The robots.txt file is typically used to provide instructions to web crawlers about which directories and files should not be indexed. However, it can sometimes unintentionally expose sensitive information.
In this case, the robots.txt file revealed two crucial pieces of information:



- A link to the first hidden key, which was accessed at:
- **http://192.168.29.64/key-1-of-3.txt**
- This confirmed the capture of Key 1, successfully achieving the first objective.




A reference to a file named fsocity.dic, which appeared to be a wordlist. This discovery suggested the potential for a brute-force attack, as wordlists are commonly used for password cracking.



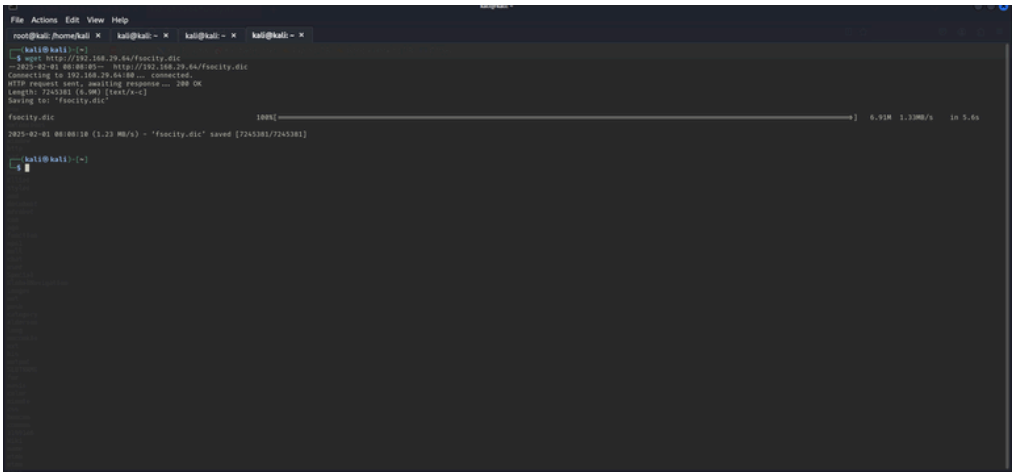
After discovering the fsocity.dic wordlist in the robots.txt file, I proceeded to download it for further analysis. Using the wget utility, the file was retrieved by executing the following command:

**wget http://192.168.29.64/fsocity.dic**

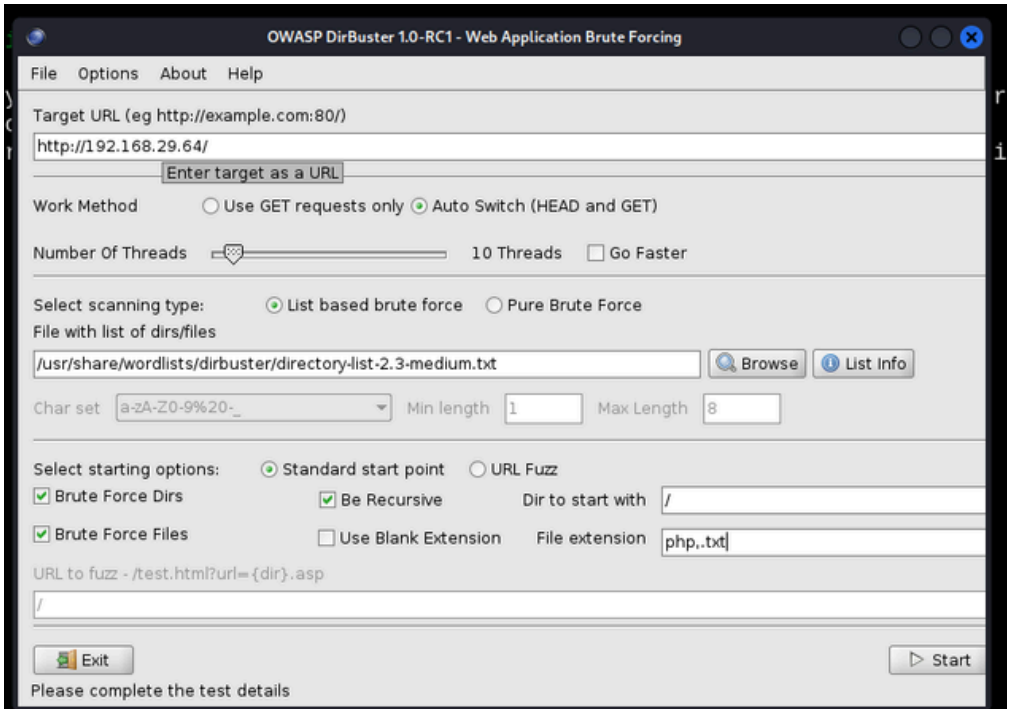
The download was successful, and the file was saved locally on the Kali Linux VM.

A preliminary inspection revealed that the file contained a large list of words, with many duplicate entries.

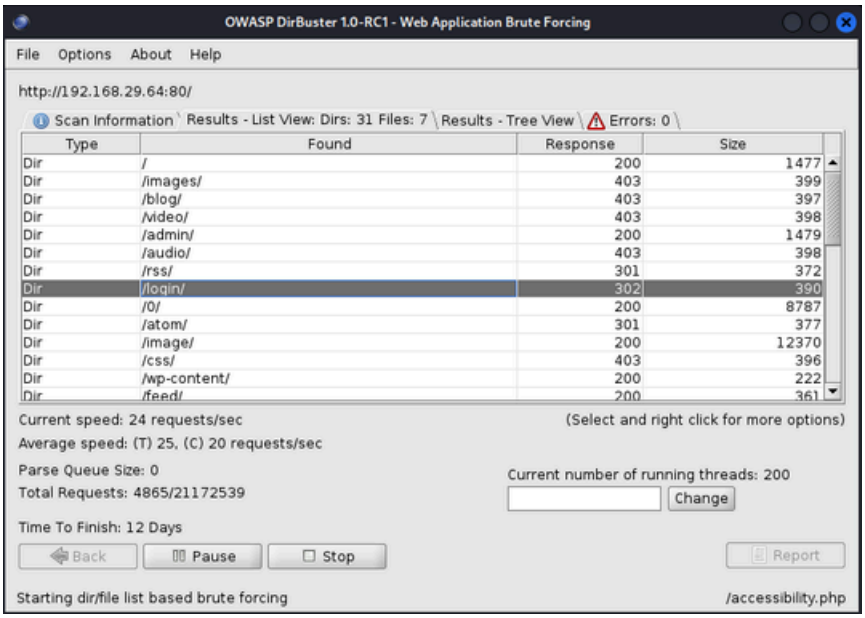
This wordlist was identified as a potential resource for performing brute-force attacks, especially on the WordPress login page discovered later during enumeration.



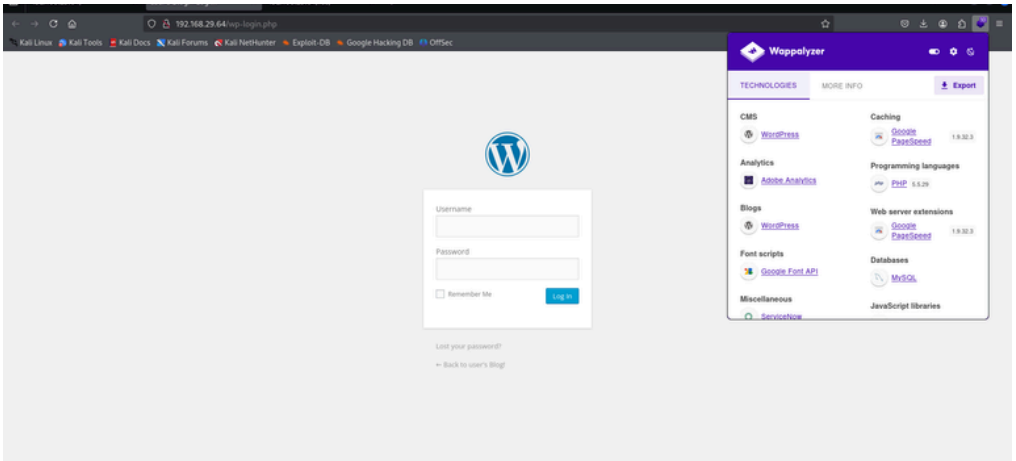
## Web Directory Enumeration

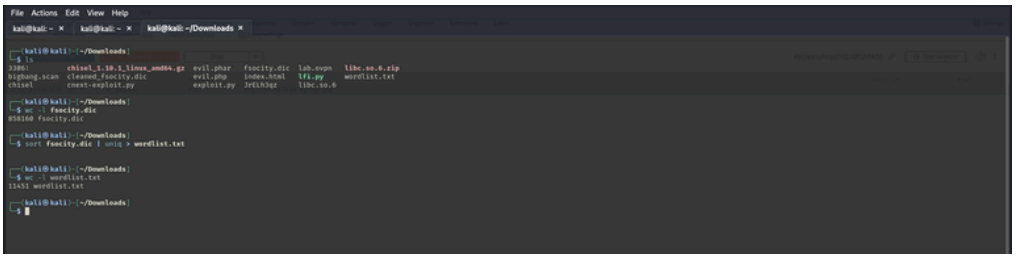


To further explore the web application, I initiated web directory enumeration using Dirbuster, a tool designed to identify hidden directories and files on a web server.
The enumeration process was conducted using a comprehensive wordlist to maximize coverage of potential directories. After a few minutes, the scan successfully revealed several accessible sub-directories associated with the website hosted on the target.

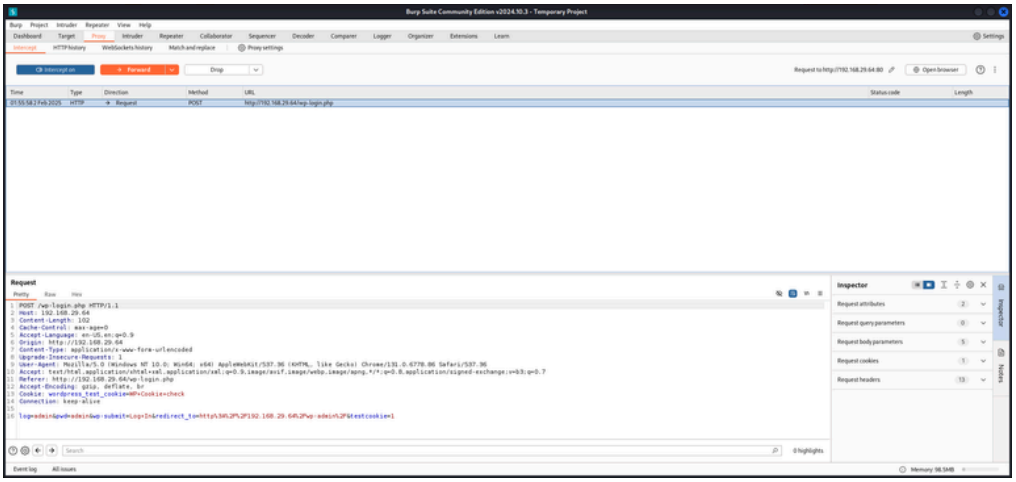



Upon completing the Dirbuster scan, several sub-directories were revealed. Among these, a directory named /login stood out as a potential entry point.
Navigating to: **http://192.168.29.64/login**
redirected to a login page. A closer examination of the page source and structure indicated that it was powered by WordPress, a popular content management system.
This discovery was significant, as WordPress is known for its extensive plugin ecosystem, which can sometimes introduce security vulnerabilities. Additionally, the presence of a login portal suggested the possibility of performing a brute-force attack using the fsocity.dic wordlist obtained earlier.

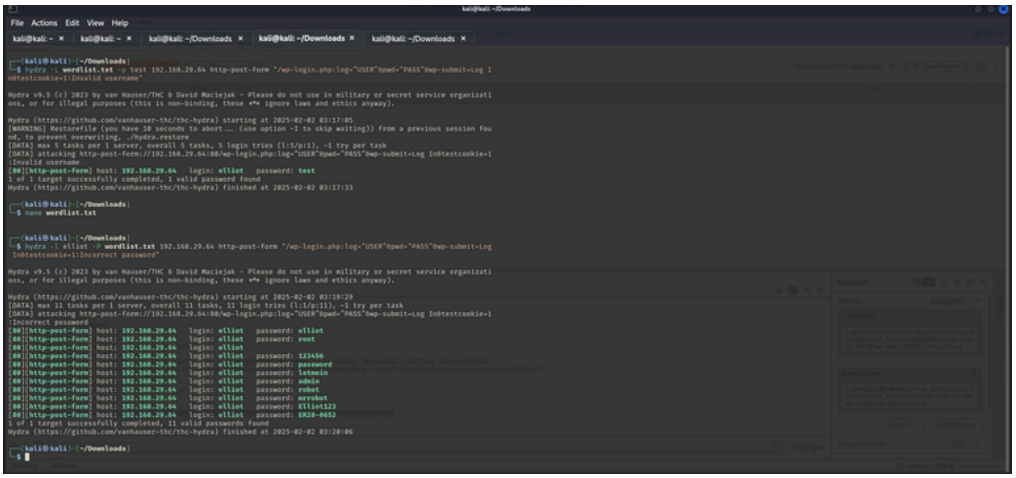




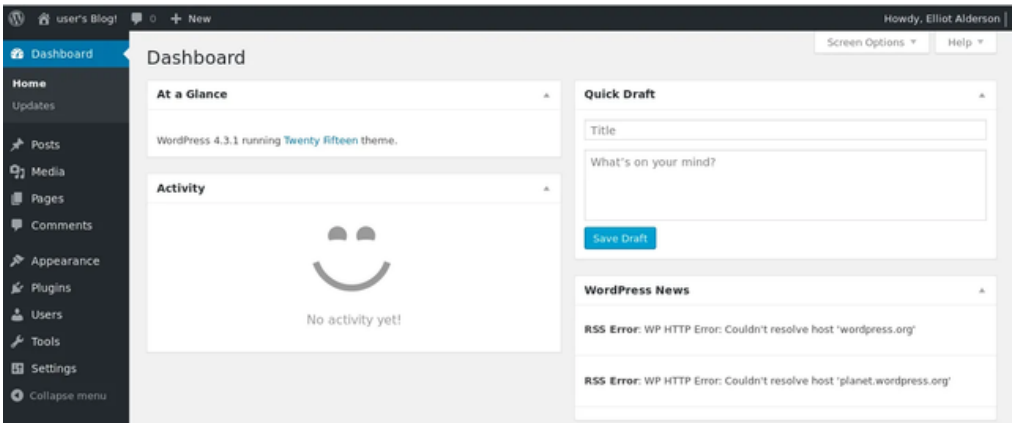
To analyze the fsocity.dic wordlist, I first checked the total number of words using:
**wc -l fsocity.dic**
Noticing many duplicate entries, I cleaned the list and improved efficiency by sorting and removing duplicates:
**sort fsocity.dic | uniq > wordlist**
I then verified the cleaned wordlist with:
**wc -l wordlist**
This process reduced the word count, optimizing the list for brute-force attacks.



Using Burpsuite, I intercepted the HTTP request on the WordPress login page.
This allowed me to analyze the POST parameters and identify the necessary fields required for a brute-force attack.
The captured information was then used to construct a targeted Hydra command, optimizing the brute-forcing process for valid credentials.



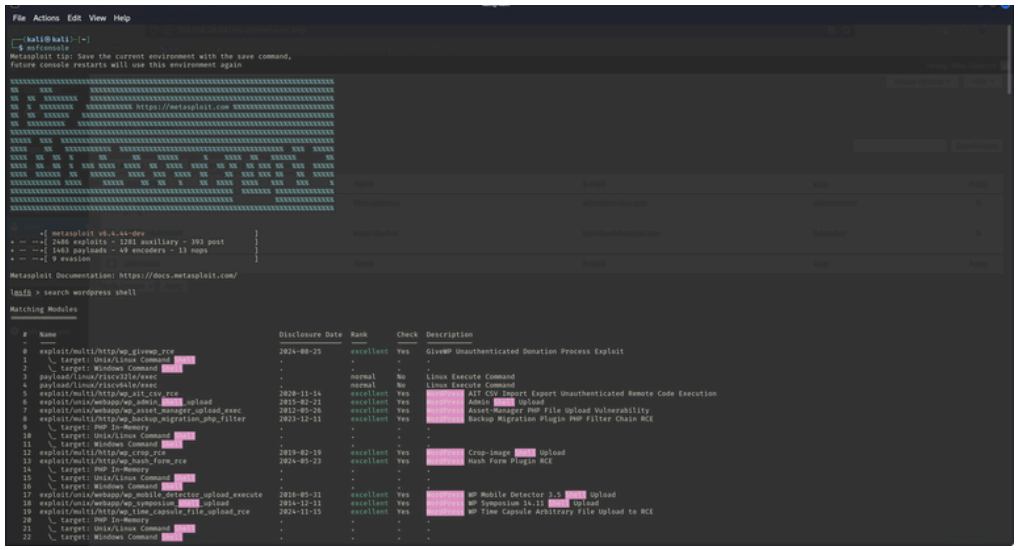
With the information gathered from Burpsuite, I used Hydra to perform a brute-force attack on the WordPress login page, targeting the username field.
By leveraging the cleaned wordlist, Hydra successfully identified the valid username:
**elliot**
Using Hydra, I brute-forced the password for elliot and found the correct one:
**ER28-0652**
This granted valid credentials for the WordPress login



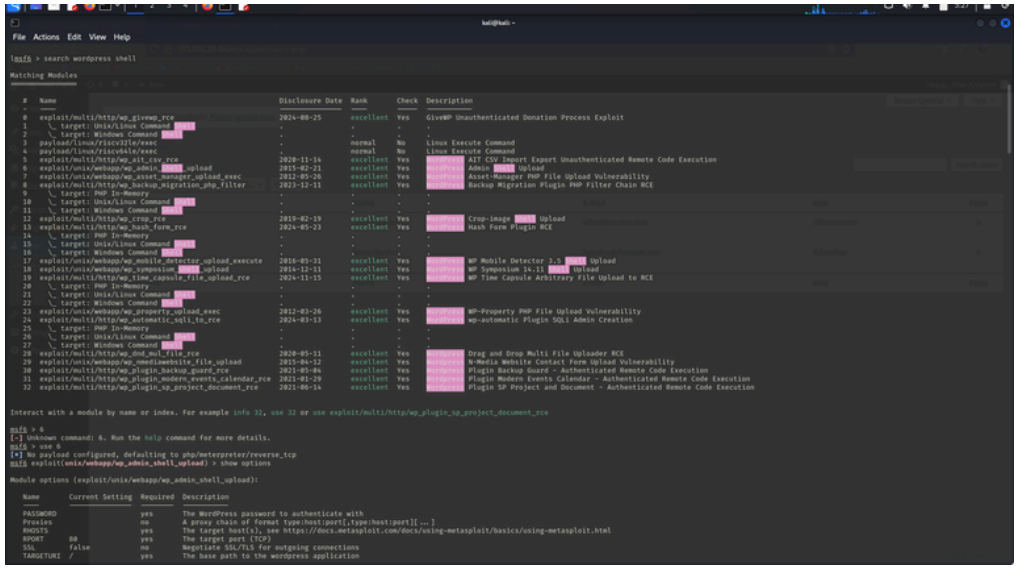

Unable to find anything useful after logging in, I used Metasploit to exploit the WordPress site and gain access

## Exploit Identification and Selection

I opened Metasploit using the msfconsole command and searched for WordPress shell exploits. During this search, I identified the exploit/unix/webapp/wp_admin_shell_upload module. This exploit is designed to upload a malicious shell through the WordPress admin panel, potentially allowing remote code execution on the target server.




To use this exploit, I executed the use 6 command, selecting the module for further configuration and execution.

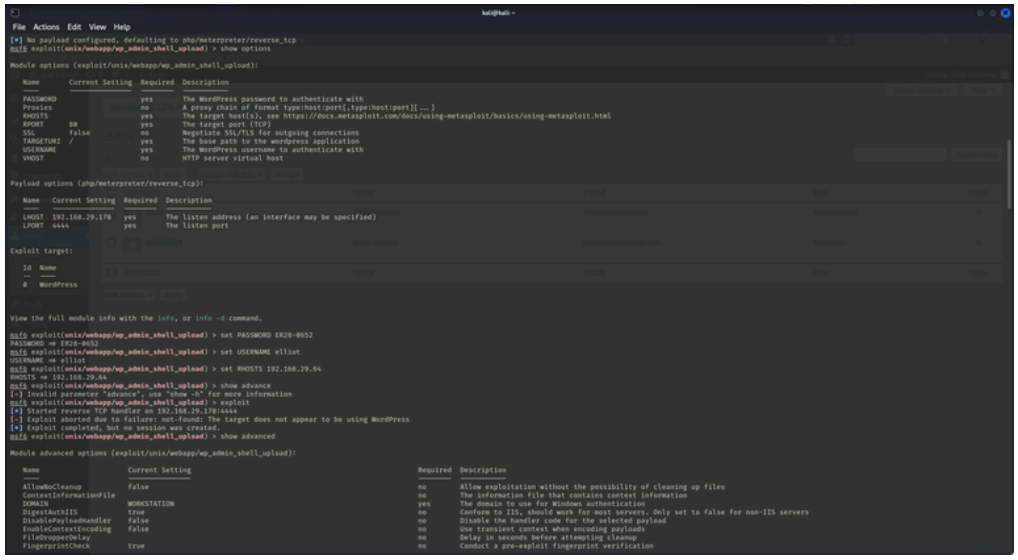


## Exploit Identification and Selection

After selecting the exploit/unix/webapp/wp_admin_shell_upload module using use 6, I proceeded by viewing the required parameters with the show options command. I then configured the necessary settings, including:

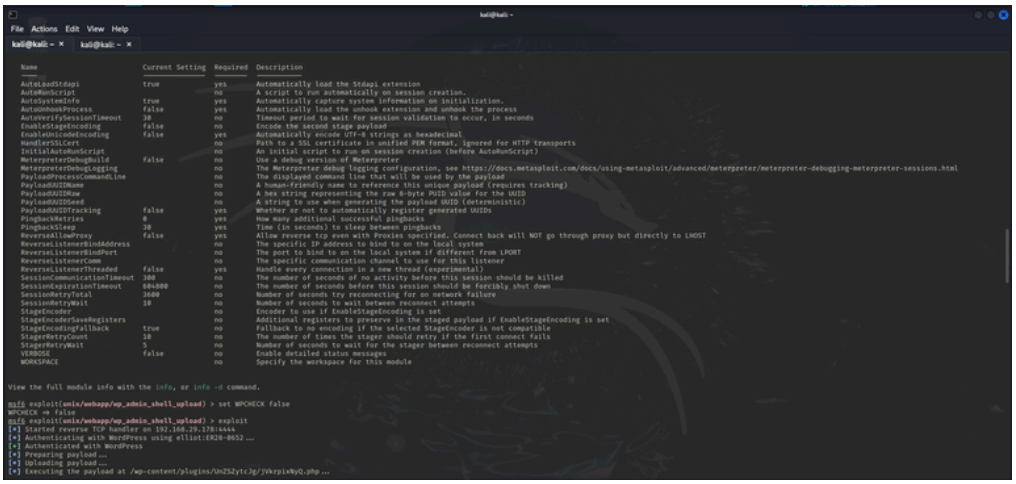
- set PASSWORD – Specifying the admin password for the target WordPress site.
- set USERNAME – Setting the admin username required for authentication.
- set RHOSTS – Defining the target's IP address.

Once all the options were correctly configured, I executed the exploit using the exploit command, initiating the shell upload and attempting to gain remote access to the target server.




After executing the exploit, I encountered an error stating, "The target does not appear to be using WordPress." To bypass this check, I used the show advanced command to display additional configuration options.

I then set WPCHECK to false to disable the WordPress version check, allowing the exploit to proceed without verifying the target's version. After adjusting this parameter, I continued by running the exploit command to initiate the attack.



After disabling the WordPress version check and running the exploit, I successfully obtained a limited shell on the target system.

To enhance the stability of the shell, I executed the following commands:

**shell**

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

This leveraged Python's pseudo-terminal (pty) module to spawn a more interactive bash shell.

I then executed whoami to confirm my current user, which returned **daemon.**

Navigating through the file system, I used the following commands: **cd /home ls**

I discovered a directory named robot and proceeded to access it using:**cd robot ls**

Within the directory, I located two files: **key-2-of-3.txt** and **password.raw-md5**.

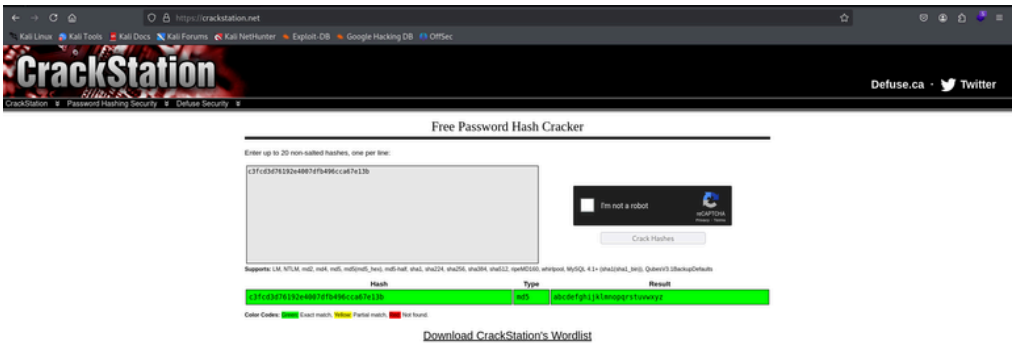
I then displayed their contents with the commands:**cat key-2-of-3.txt**

However, I encountered a Permission Denied error, indicating insufficient privileges to access the file.

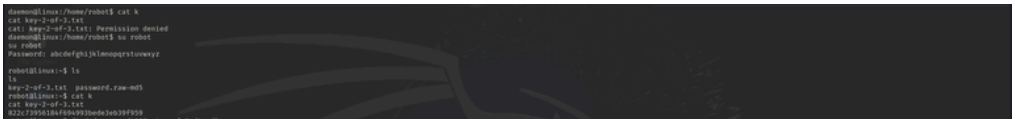
To investigate further, I opened the other file in the directory:**cat password.raw-md5**

The contents of this file appeared to be a hashed password, and the filename indicated that it was an MD5 hash. This discovery suggested that cracking the hash might provide the necessary credentials to access the second key.

To decrypt the MD5 hash found in password.raw-md5, I used CrackStation, an online hash-cracking tool. Upon entering the hash, CrackStation successfully decrypted it to:
**abcdefghijklmnopqrstuvwxyz**




Using this as the password, I switched to the robot user with the following command:
**su robot**
When prompted for the password, I entered the decrypted value and successfully gained access as the robot user.
Now having the necessary permissions, I revisited the robot directory and executed:
**cat key-2-of-3.txt**
This revealed the second key, successfully progressing towards the overall objective.

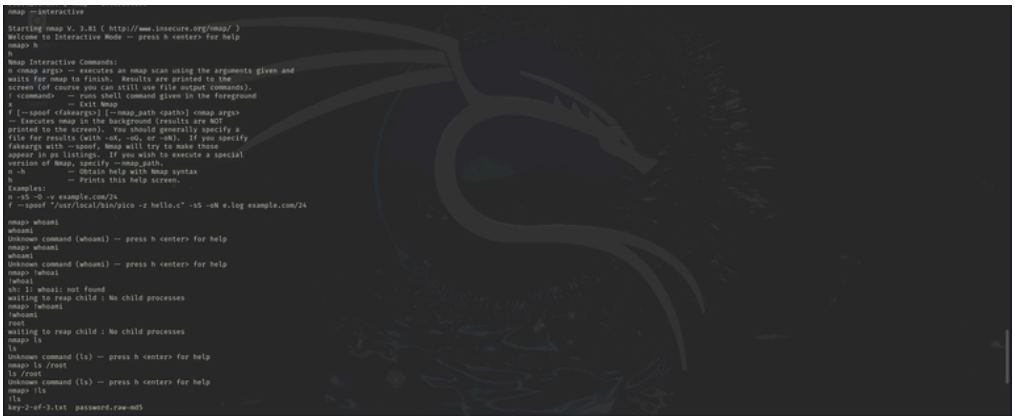


I used the command:
**find / -perm /4000 -type f 2>/tmp/2**
to search for SUID files, which can be exploited for privilege escalation. Among the results, I found several files, but the most interesting one was Nmap. To investigate further, I executed:
**nmap --interactive**
This initiated Nmap's interactive mode, which can potentially be leveraged for gaining elevated privileges.

While in Nmap's interactive mode, I entered the command:

**h**

to display the help menu. I discovered that Nmap allows executing system commands using the ! prefix.




I used this feature to navigate to the root directory and read the final key:

!ls /root

!cat /root/key-3-of-3.txt

This successfully revealed Key 3, completing the objective of capturing all three hidden keys.

# Recommendations

### Access Control and Authentication

- Enforce Strong Password Policies: Implement complex password requirements, including minimum length, special characters, and case sensitivity.
- Account Lockout Mechanism: Enable account lockout after multiple failed login attempts to prevent brute-force attacks.
- Avoid Default Usernames: Refrain from using common usernames like admin or elliot.
- Implement Multi-Factor Authentication (MFA): Strengthen access control by requiring multiple authentication factors.

### Web Application Security

- Restrict Access to Sensitive Files: Limit public access to robots.txt and sensitive directories.
- Secure Administrative Access: Disable file editing in the WordPress admin panel to prevent unauthorized shell uploads.
- Update WordPress and Plugins: Regularly update the CMS and plugins to patch known vulnerabilities.

### Privilege Escalation Prevention

- Review SUID Permissions: Audit and minimize files with SUID permissions, including Nmap, to prevent privilege escalation.
- Apply Principle of Least Privilege (PoLP): Limit user privileges to only what is necessary for their role

### Network Security

- Network Segmentation: Isolate network segments to limit lateral movement.
- Traffic Monitoring and Logging: Enable logging for web servers and authentication mechanisms to detect suspicious activities.

### General Security Best Practices

- Conduct Regular Security Audits: Perform periodic penetration tests and vulnerability assessments.
- User Awareness Training: Educate users on phishing, social engineering, and password hygiene.
- Incident Response Plan: Establish a security incident response plan to handle potential breaches.

# Mitigations

**Access Control and Authentication**

- Implement Account Lockout Policies: Enforce lockout policies after multiple failed attempts to mitigate brute-force attacks.
- Enforce Password Complexity Rules: Require complex passwords with a combination of letters, numbers, and special characters.
- Disable Default Accounts: Remove or rename default accounts to reduce the risk of targeted attacks.

**Web Application Security**

- Restrict Access to robots.txt: Limit public access or exclude sensitive directories from it.
- Disable File Editing in WordPress: Add the following line to the wp-config.php file to disable file editing:
- **define('DISALLOW_FILE_EDIT', true);**
- Apply Security Patches: Regularly update WordPress core, themes, and plugins.

**Privilege Escalation Mitigation**

- Audit SUID Permissions: Regularly check for SUID binaries using:
- **find / -perm /4000 -type f**
- Remove unnecessary SUID bits, especially on Nmap.
- Limit User Privileges: Implement the Principle of Least Privilege (PoLP) to restrict access rights.

**Network Security**

- Implement Network Segmentation: Use VLANs and firewalls to isolate network segments.
- Enable Logging and Monitoring: Set up monitoring tools to detect unusual activities, especially around authentication mechanisms.

**General Security Best Practices**

- Schedule Regular Security Audits: Conduct routine security assessments and patch management.
- Employee Security Awareness: Implement ongoing security training programs.
- Incident Response and Recovery: Establish an incident response team and recovery plan to handle security breaches.