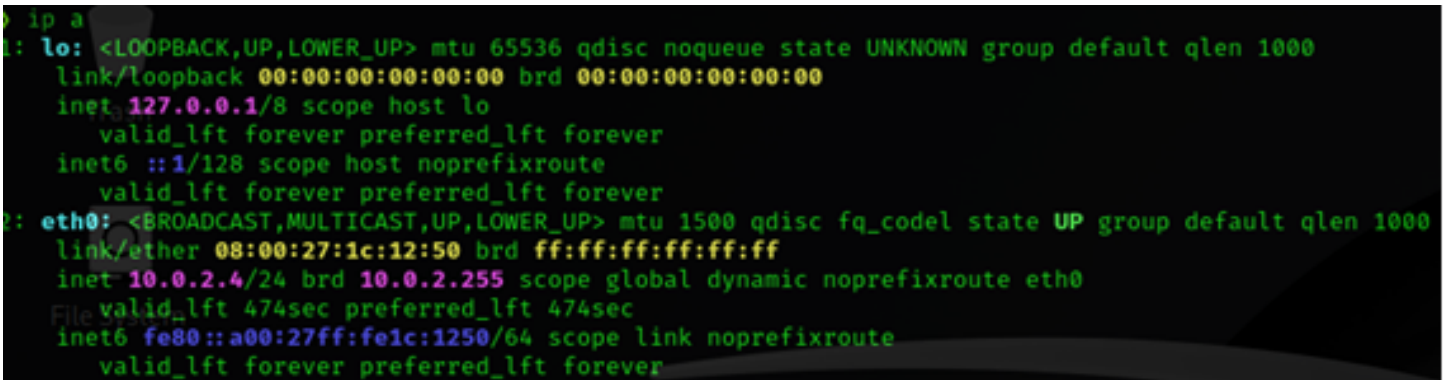
RETO PYTHON - TEAM CHALENGE - SPRINT 1 - CIBERSEGURIDAD

Elaborado por:	Fecha de creación:
Víctor Manuel Martínez Barberá	03/06/2024

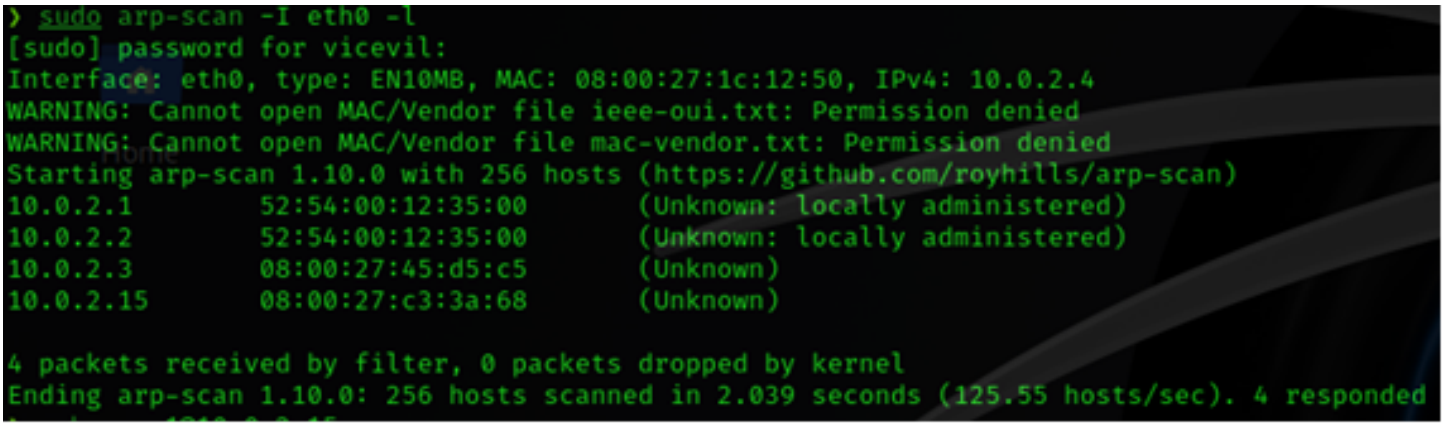
En relación al reto practico indicado, se han practicado las instrucciones recibidas en los archivos en pdf habiendo obtenido un resultado satisfactorio, como se puede observar en las siguientes capturas:

1 •— Una vez configurada la imagen .ova del reto, se abrió en el Virtual Vox esta maquina, aperturando además mi Kali Linux, ejecutando el comando ip a, con el cual se muestra información detallada sobre las interfaces de red de tu sistema Linux, tanto las físicas como las virtuales, usándose para verificas si tus interfaces están bien configuradas, con las IPs esperadas. También se puede usar esta información para hacer nuestro scripts.



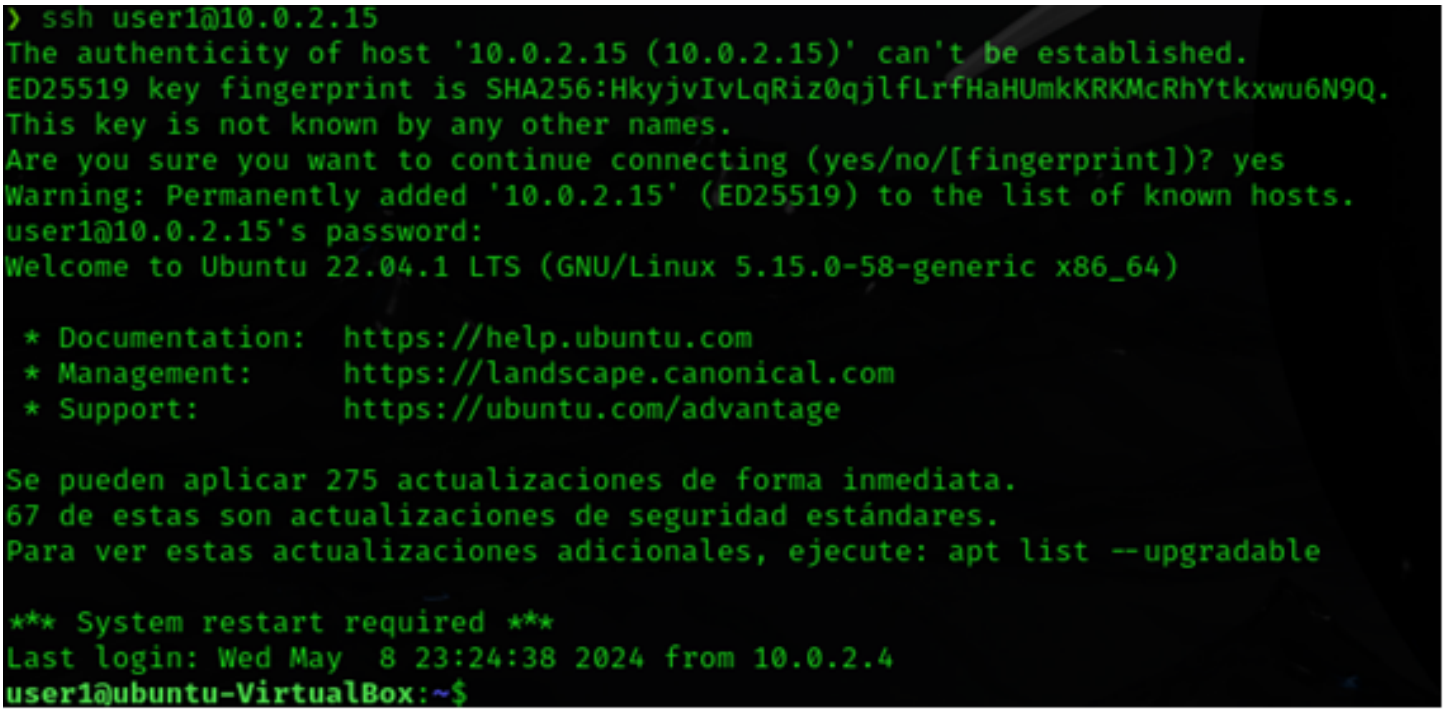
Extraemos que nuestra interface de red es eth0.

2 •— Ejecutamos el comando arp-scan -I eth0 -l, con el cual realizamos un mapeo de la red buscando los dispositivos que están conectados a la red eth0, sus IPs , para ellos envía solicitudes ARP (protocolo de resolución de direcciones) a todas las direcciones de mi red, esperando respuesta de las mismas. Este comando es muy útil en ciberseguridad para mapear la red en busca de intrusiones no autorizadas.



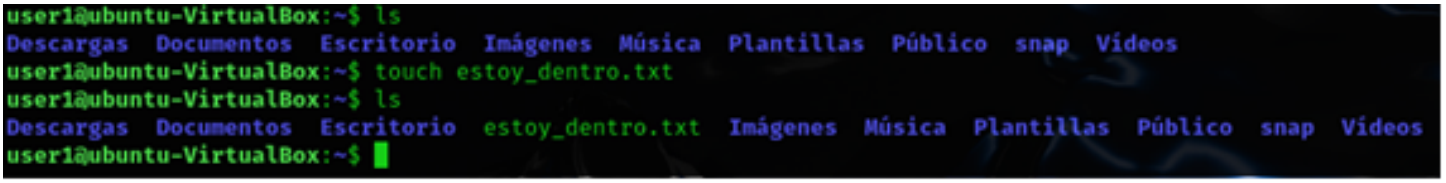
En el resultado observamos 4 direcciones IPs (10.0.2.1, 10.0.2.2 y 10.0.2.3), siendo éstas utilizadas por VirtualBox para facilitar la comunicación entre el sistema anfitrión y el sistema invitado, por lo que la IP donde realizar la conexión segura.

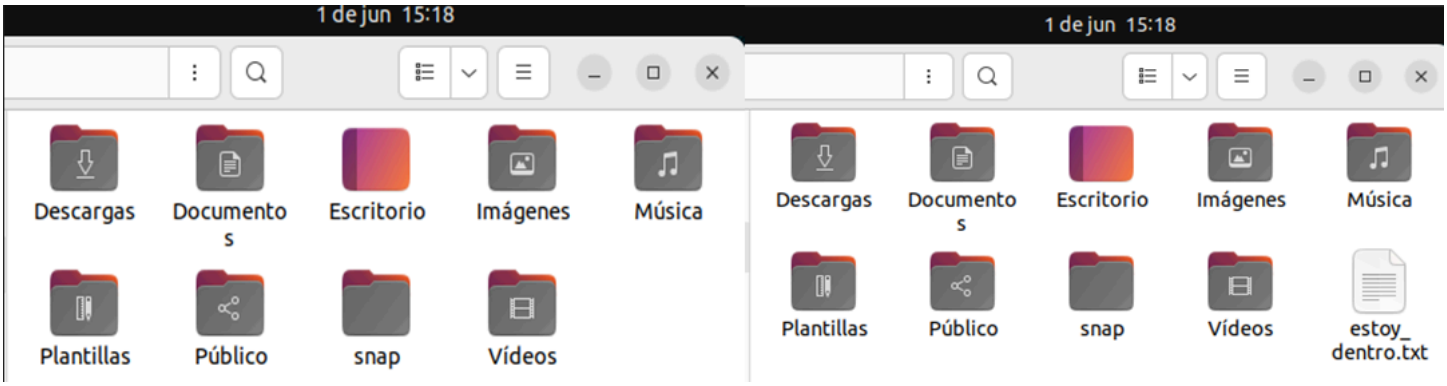
3 •— Se procede a establecer la conexión segura, tunelizada y cifrada simétricamente mediante clave pública entre el sistema anfitrión y la maquina Reto_python, siendo utilizado en ciberseguridad, para automatizar de tareas periódicas repetitivas en servidores remotos mediante scripts.



Aquí ya estaríamos conectados, por lo que podríamos ejecutar comandos, transferir archivos, instalar programas y realizar otras tareas de forma segura (los administradores de sistemas se conectan así, y además consume menos recursos que el TeamViewer).

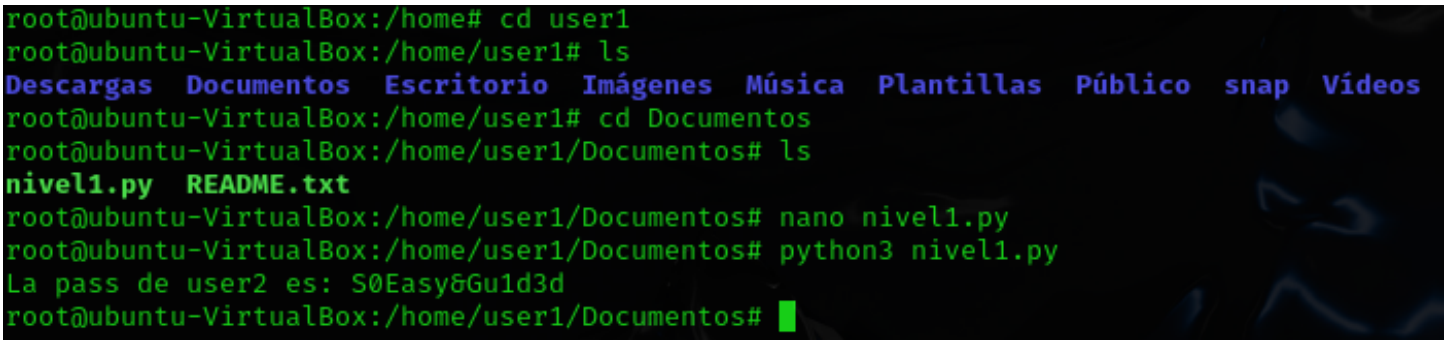
4 •— finalmente hacemos la comprobación si realmente están conectados, creando un archivo estoy_dentro.txt desde la maquina anfitrión, siendo las misma positiva:



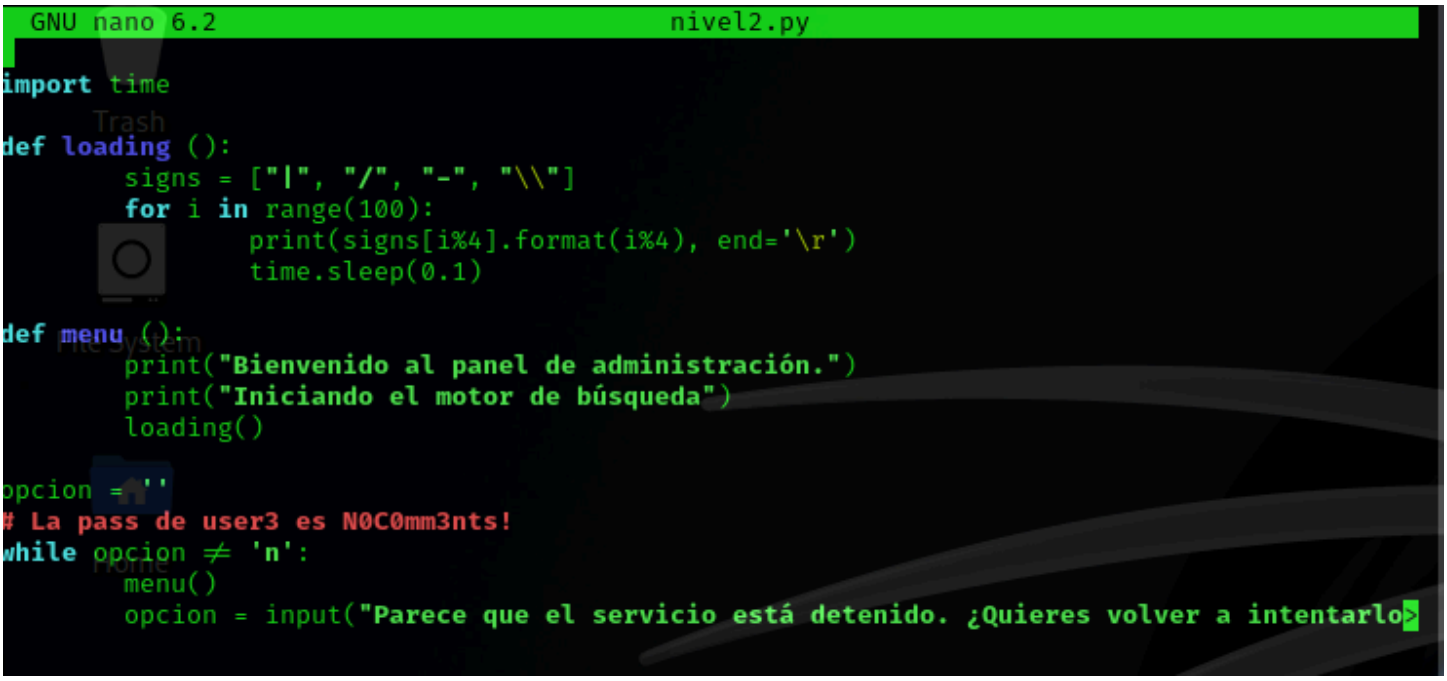


NIVELES

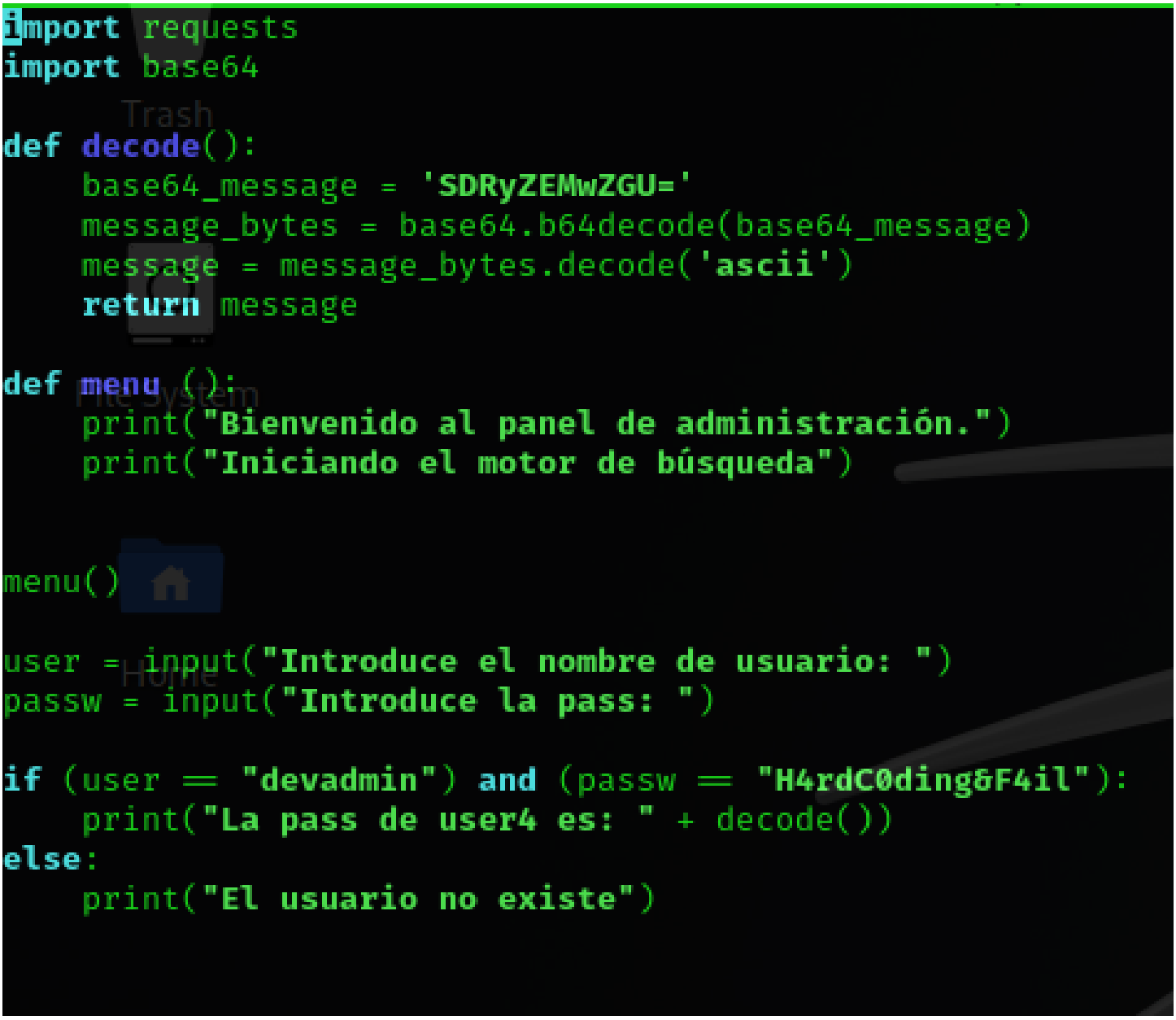
NIVEL1:



NIVEL 2: S0Easy&Gu1d3d



NIVEL 3 N0C0mm3nts!



```
root@ubuntu-VirtualBox:/home/user3/Documentos# nano nivel3.py
root@ubuntu-VirtualBox:/home/user3/Documentos# python3 nivel3.py
Bienvenido al panel de administración.
Iniciando el motor de búsqueda
Introduce el nombre de usuario: devadmin
Introduce la pass: H4rdC0ding&F4il
La pass de user4 es: H4rdC0de
```

NIVEL4: H4rdC0de

```python
import sys
import time
import base64

def decode():
    base64_message = 'UEByYW1ldDNycw=='
    message_bytes = base64.b64decode(base64_message)
    message = message_bytes.decode('ascii')
    return message

def loading ():
        signs = ["|", "/", "-", "\\"]
        for i in range(40):
                print(signs[i%4].format(i%4), end='\r')
                time.sleep(0.1)

def menu ():
    print("Bienvenido al panel de administración.")
    print("Iniciando el motor de búsqueda")
    loading()


args = sys.argv
menu()

if (len(args)%2 == 0) and (args[4] == 'debug'):
        print("La pass de user5 es: " + decode())

else:
        print("Parece que el servicio está detenido. Vuelve a intentarlo")
```

Aquí tengo que interpretar el código python. Este comienza con al arg0 y así sucesivamente, así que arg0 es el propio script, arg1,2 y 3 son los otros 3 argumentos, que cumple el total de argumentos el cociente es cero, y el ultimo deber se = debug

```
root@ubuntu-VirtualBox:/home/user4/Documentos# nano nivel4.py
root@ubuntu-VirtualBox:/home/user4/Documentos# python3 nivel4.py argumento1 argumento2 debu
g
Bienvenido al panel de administración.
Iniciando el motor de búsqueda
La pass de user5 es: P@ramet3rs
```

NIVEL5:P@ramet3rs

```python
import requests
import base64

def decode():
    base64_message = 'RXh0M3JuYWxLM1k='
    message_bytes = base64.b64decode(base64_message)
    message = message_bytes.decode('ascii')
    return message

response = requests.get('https://gist.githubusercontent.com/poundifdef/fd0901799a4aeb3a1f3956cfdb3c7746/raw/36f07f4
key = str(response.text).split('\n')[-2].strip()

user_response = str(input("¿Cómo puedo ayudarte? \n"))


if user_response == key:
    print("La pass de user6 es: " + decode())
else:
    print("No puedo ayudarte con eso")
```

Al parecer que el enlace web le faltan elementos y no llega a ningún sitio ni da respuesta, por lo que procedo a investigar por la web llegando a la web original :
“https://gist.githubusercontent.com/poundifdef/fd0901799a4aeb3a1f3956cfdb3c7746/raw/36f07f44ba9f80eb93fbb336e2bdf4096adeb484/trollface.txt”

Una vez hecho esto incluyo en el script una simple respuesta para que me de la clave a introducir en el mensaje, devolviéndose la password del user6

```python
import requests
import base64

def decode():
    base64_message = 'RXh0M3JuYWxLM1k='
    message_bytes = base64.b64decode(base64_message)
    message = message_bytes.decode('ascii')
    return message

response = requests.get('https://gist.githubusercontent.com/poundifdef/fd0901799a4aeb3a1f3956cfdb3c7746/raw/36f07f44ba9f80eb93fbb336e2bdf4096adeb484/trollface.txt')
key = str(response.text).split('\n')[-2].strip()  # Obtiene la clave del archivo

print("La clave es: " + str(key))  # Imprime la clave

user_response = str(input("¿Cómo puedo ayudarte? \n"))


if user_response == key:
    print("La pass de user6 es: " + decode())
else:
    print("No puedo ayudarte con eso")
```

```
✓ 11.1s
La clave es: -:o+/::..
La pass de user6 es: Ext3rnalK3Y
```

NIVEL6: Ext3rnalK3Y

Al no encontrar la bases de datos nombrada en el archivo python procedo a buscar el archivo oculto en la misma carpeta:
“find / -name ".registered_users.db" 2>/dev/null”

Una vez ubicado el archivo, se comprueba que no es una BBDD, así que se trata como un archivo normal
Procedo a ver el contenido del archivo , ejecutando la siguiente función, y comprando que hay mas de un millón de registros:

```python
def read_db():
    with open(PATH_BBDD, 'r') as file:
        lines = file.readlines()
        for line in lines:
            print(line.strip())  # Imprime cada línea (email) en el archivo

read_db()
```

Por lo que borramos los ultimo registros, dejando hasta el 999998.

```python
def delete_last_lines(filename, n):
    # Lee todas las líneas del archivo
    with open(filename, 'r') as file:
        lines = file.readlines()

    # Reescribe el archivo con todas las líneas excepto las últimas n
    with open(filename, 'w') as file:
        for line in lines[:-n]:
            file.write(line)

delete_last_lines(PATH_BBDD, 9)
```

Ahora ya ejecutamos dos veces el script del ejercicio, resultando la contraseña

```
❯ python nivel6.py
Bienvenido al concurso 1.000.000
Registrate para ganar un fantástico premio. ¿Serás tu el participa
nte 1.000.000?
Introduce tu dirección de mail: casies@ciber.es
Estamos verificando si eres el participante 1.000.000 ...
Lo sentimos, no eres el participante 1.000.000
❯ python nivel6.py
Bienvenido al concurso 1.000.000
Registrate para ganar un fantástico premio. ¿Serás tu el participa
nte 1.000.000?
Introduce tu dirección de mail: victormbar@gmail.com
Estamos verificando si eres el participante 1.000.000 ...
Enhorabuena, has ganado!
La pass de user7 es: =0neM1llion
```

NOTA: Coincide con el resultado que da la función decode al ejecutarla.

NIVEL 7: =0neMillion

Para la clave del User 8 Aparecen dos archivos.py:

```python
def getotp():
    return str(random.randint(10000000,99999999))

def decode_pass():
    base64_string = "TGEgcGFzcyBkZSB1c2VyOCBlczogVENQIVMwY2szdA=="
    base64_bytes = base64_string.encode("ascii")

    sample_string_bytes = base64.b64decode(base64_bytes)
    sample_string = sample_string_bytes.decode("ascii")
    return sample_string

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(("127.0.0.1", 8050))
conn.listen(1)
cli, addr = conn.accept()
otp = ""

while True:

    recibido = cli.recv(1024)

    print("Recibo conexion de la IP: " + str(addr[0]) + " Puerto: " + str(addr[1]))

    comando = recibido.decode()

    if comando == "user":
        otp=getotp()
        cli.send(otp.encode())
    elif comando == otp:
        cli.send(decode_pass().encode())
    else:
        cli.send("404 Command not found".encode())
```

```python
host = '127.0.0.1'
port = 8050

obj = socket.socket()
obj.connect((host, port))

print("Conectado al servidor")

while True:
    mens = input("Comando >> ")

    obj.send(mens.encode())

    respuesta = obj.recv(1024)

    print(respuesta.decode())

obj.close()

print("Conexión cerrada")
```

Una vez observados, se comienza ejecutando independientemente la función “decode_pass” , que nos muestra la contraseña valida.

```
def getotp():
    return str(random.randint(10000000,99999999))

def decode_pass():
    base64_string = "TGEgcGFzcyBkZSB1c2VyOCBlczogVENQIVMwY2szdA=="
    base64_bytes = base64_string.encode("ascii")

    sample_string_bytes = base64.b64decode(base64_bytes)
    sample_string = sample_string_bytes.decode("ascii")
    return sample_string
decode_pass()
```

✓ 0.0s

```
'La pass de user8 es: TCP!S0ck3t'
```

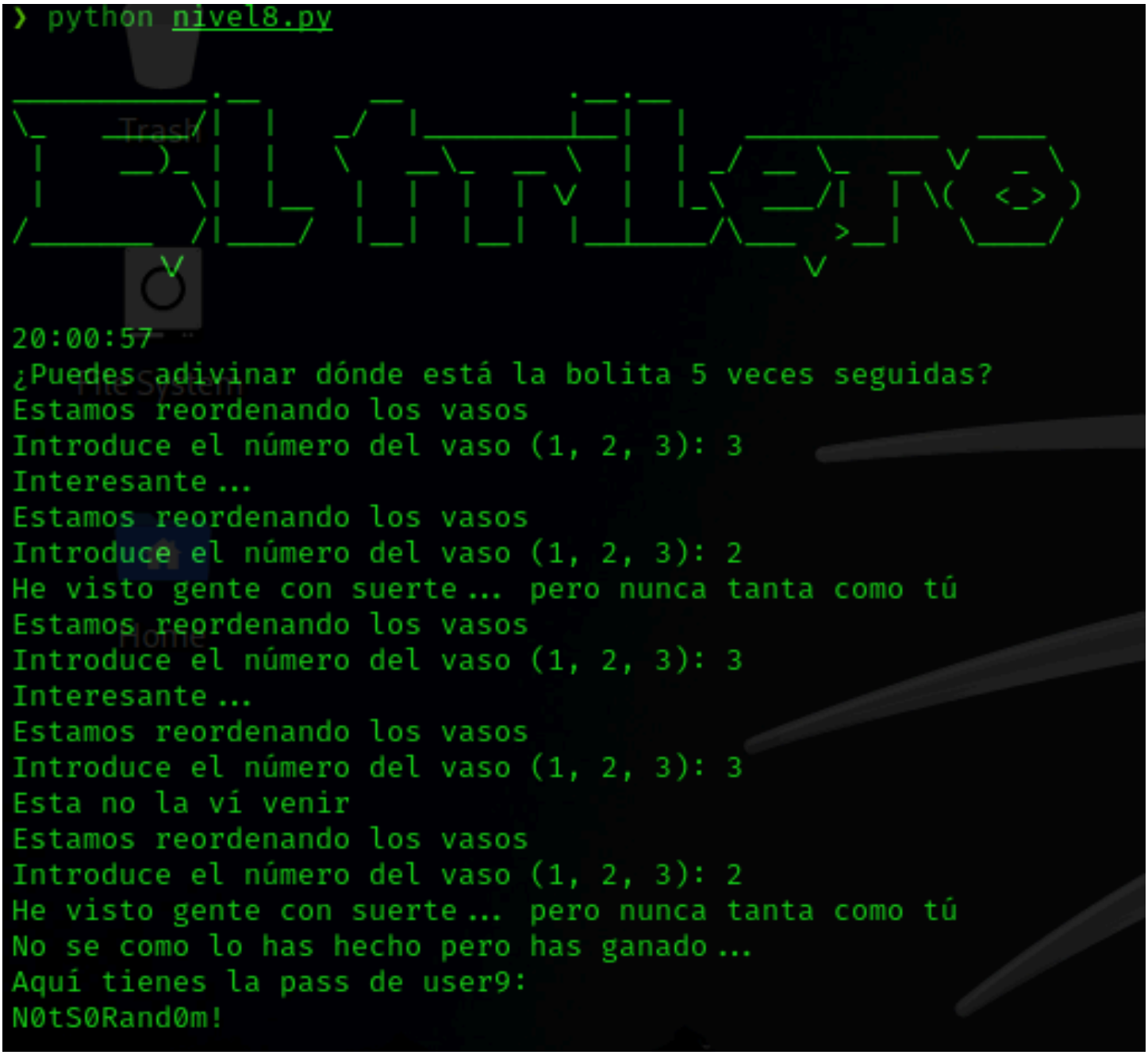
NIVEL8:TCP!S0ck3t

EL juego del trilero•– Se ejecuta la función “getNewPosition” observando un patrón que se repite:

```
def getNewPosition(timeexec,round):
        base = timeexec.split(':')[round%3]
        position = (int(base)+round)%3
        return str(position + 1)
```

```
LA SERIE ES 3,2,3,3,2,3,3,2,3,3,2....
```

Aplicando dicho patrón, y siguiendo la secuencia lógica de respuestas hallada:



NOTA: igual que anteriormente, misma contraseña que sale al ejecutar la función decode

NIVEL9: N0tS0Rand0m!

Se procede a decodificar la parte que viene en la función execute_low: "user10 - Encoded pass: 6e6957267376697250796669646f4d" en hexadecimal, creando una función para ello:

```python
#vamos a realziar una funcion de decode en basw hEXADECIMAL
def decode_hex(hex_string):
    bytes_object = bytes.fromhex(hex_string)
    ascii_string = bytes_object.decode("ASCII")
    return ascii_string


print(decode_hex(baseHX_message))
```

✓ 0.0s

```
niW&svirPyfidoM
```

Se observa también que en la función execute_high aparece otro print con la información "Hex(Reverse(password))", por lo que se intuye que hay que revertir la contraseña decodificada:

```python
clave_hex= "niW&svirPyfidoM"
clave_hex_invertida = clave_hex[::-1]
clave_hex_invertida
```

✓ 0.0s

```
'ModifyPrivs&Win'
```

NIVEL10: ModifyPrivs&Win