Universidad del Valle de Guatemala
Cifrado de Información
Sección 10

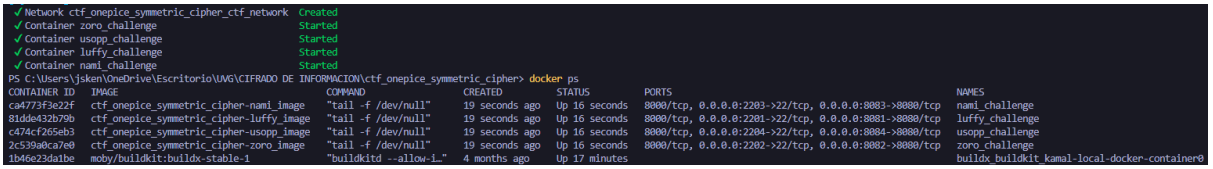
Proyecto 1

José Pablo Santisteban Vargas
Carnet - 21153

Guatemala, 24 de junio del 2025

Generación de retos de proyecto

Creación de imágenes de docker



Método de búsqueda

La estrategia a seguir para encontrar las flags consiste en utilizar un script en Python que recorre recursivamente todos los directorios y subdirectorios dentro de la carpeta ONEPIECE, listando cada archivo encontrado. De esta forma, se pueden identificar fácilmente todos los archivos presentes en el entorno del usuario, permitiendo examinar su contenido en busca de posibles flags. Esta búsqueda sistemática asegura que no se omita ningún archivo oculto o ubicado en subcarpetas. A pesar de que esta es una estrategia que no es muy eficiente, esta permite localizar todos los archivos que se encuentran dentro del directorio de ONEPIECE.

```python
import os

def buscar_archivos():
    base_dir = "./"
    for root, dirs, files in os.walk(base_dir):
        for file in files:
            ruta_completa = os.path.join(root, file)
            print(ruta_completa)

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

El código superior es el utilizado para revisar todos los archivos dentro de un directorio, teniendo una estructura simple, en donde se busca recursivamente hasta encontrar un archivo y finalmente muestra la ruta donde se encuentra el archivo.

Challenge 1: Luffy

Mediante el uso del script de python que se mencionó en el método de búsqueda se obtuvieron las siguientes rutas en donde se encontraron archivos:

```
luffy@81dde432b79b:~/ONEPIECE$ python3 listar_archivos_onepiece.py
./listar_archivos_onepiece.py
./Whole_Cake_Island/Cacao_Island/Casa_de_Katakuri/flag.txt
./Sabaody_Archipelago/Grove_1/Casa_de_Rayleigh/flag.txt
./Sabaody_Archipelago/Grove_1/Casa_de_Shakky/poneglyph.zip
./Sabaody_Archipelago/Grove_66/Casa_de_Keimi/flag.txt
./Sabaody_Archipelago/Grove_80/Casa_de_Rayleigh/flag.txt
./Pirate_Island/Pirates_Den/Casa_de_Whitebeard/flag.txt
./Pirate_Island/Pirates_Hideout/Casa_de_Shanks/flag.txt
./Pirate_Island/Pirates_Tavern/Casa_de_Whitebeard/flag.txt
./Dressrosa/Acacia/Casa_de_Viola/flag.txt
./Water_7/Carpenters_Cafe/Casa_de_Paulie/flag.txt
./Water_7/Franky_House/Casa_de_Luffy/flag.txt
./Water_7/Blue_Station/Casa_de_Paulie/flag.txt
```

Flag

Después de buscar en todos los directorios listados, se determinó que la flag se encontraba en la siguiente ruta:

```
luffy@81dde432b79b:~$ ls
ONEPIECE
luffy@81dde432b79b:~$ cd ONEPIECE/
luffy@81dde432b79b:~/ONEPIECE$ ls
Dressrosa  Pirate_Island  Sabaody_Archipelago  Water_7  Whole_Cake_Island
luffy@81dde432b79b:~/ONEPIECE$ cd Pirate_Island
luffy@81dde432b79b:~/ONEPIECE/Pirate_Island$ cd Pirates_Hideout
luffy@81dde432b79b:~/ONEPIECE/Pirate_Island/Pirates_Hideout$ cd Casa_de_Shanks
luffy@81dde432b79b:~/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_Shanks$ ls
flag.txt
```

Después de encontrar la flag, se extrajo el contenido usando el comando cat:
747d70726c5709000d0a0355545650070855 0d5100530456015405500 20a03530750035652

Descifrado de Flag

Gracias a las instrucciones del proyecto, se sabe que el método de cifrado del reto de luffy es xor, usando como clave el carnet del estudiante, que en mi caso es 21153. El código desarrollado es el siguiente:

```python
def xor_decrypt_hex(hex_string: str, key: str) -> str:
    """
    Descifra un texto cifrado con XOR usando una clave, donde el texto está en hexadecimal.
    """
    ciphertext = bytes.fromhex(hex_string)
    key_bytes = key.encode()

    decrypted = bytearray()
    for i in range(len(ciphertext)):
        decrypted.append(ciphertext[i] ^ key_bytes[i % len(key_bytes)])

    try:
        return decrypted.decode("utf-8")
    except UnicodeDecodeError:
        # Si no se puede decodificar, lo mostramos como bytes
        return str(decrypted)


texto_hex = "747d70726c5709000d0a0355545650070855 0d5100530456015405500 20a03530750035652"
clave = "21153"

texto_descifrado = xor_decrypt_hex(texto_hex, clave)
print("Texto descifrado:\n", texto_descifrado)
```

Este script toma como entrada una cadena en formato hexadecimal que representa el texto cifrado, y utiliza la clave (el carné) para aplicar una operación XOR byte por byte entre el texto cifrado y la clave, que se repite de manera cíclica. Luego, se intenta decodificar el resultado como texto UTF-8 para obtener el mensaje original. Esta técnica permite descifrar correctamente la flag siempre que la clave utilizada para el cifrado coincida con la del descifrado, demostrando cómo funciona la lógica básica de un cifrado XOR simétrico.

Después de decodificar el resultado fue el siguiente: FLAG_e81891decc59d8b2b5c2f4a791b6e0dc

Poneglyph

Mediante el buscador de archivos utilizado, se determinó que la ruta de la imagen es la siguiente, ya que fue la única ruta en donde se encontró un poneglyph.zip

```
luffy@81dde432b79b:~$ cd ONEPIECE/
luffy@81dde432b79b:~/ONEPIECE$ cd Sabaody_Archipelago/
luffy@81dde432b79b:~/ONEPIECE/Sabaody_Archipelago$ cd Grove_1
luffy@81dde432b79b:~/ONEPIECE/Sabaody_Archipelago/Grove_1$ cd Casa_de_Shakky/
luffy@81dde432b79b:~/ONEPIECE/Sabaody_Archipelago/Grove_1/Casa_de_Shakky$ ls
poneglyph.zip
```

Después de descargar el .zip y utilizar la contraseña inicial ‘onepiece’ se pudo obtener la siguiente imágen:

En el repositorio que se proporcionó en clase se brindó una función para poder extraer el texto dentro de las imágenes de los poneglyphs y el resultado fue el siguiente:

```
PS C:\Users\jsken\OneDrive\Escritorio\UVG\CIFRADO DE INFORMACION\CIFRADO_INFO\PROYECTO_DES> & C:/Users/jsken/AppData/Local/Programs/Python/Python311/pyth
INFO/PROYECTO_DES/read_image.py"
Poneglyph de Luffy
Introduce tu carné para descifrar el mensaje: 21153
b'Crocodile targeted the Arabasta Kingdom because of its Poneglyph, which contained information on the whereabouts of Pluton, '
```

Crocodile targeted the Arabasta Kingdom because of its Poneglyph, which contained information on the whereabouts of Pluton,

Challenge 2: Zoro

Gracias a la flag encontrada y descifrada en el reto anterior se pudo ingresar exitosamente en el usuario de zoro.

```
nobody@2c539a0ca7e0:/home/zoro$ su zoro
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

zoro@2c539a0ca7e0:~$
```

Mediante el uso del script de python se encontraron los siguientes directorios con archivos:

```
zoro@2c539a0ca7e0:~/ONEPIECE$ python3 listar_archivos_onepiece.py
./listar_archivos_onepiece.py
./Whole_Cake_Island/Caramel_Mountain/Casa_de_Big_Mom/poneglyph.zip
./Whole_Cake_Island/Liqueur_Island/Casa_de_Pudding/flag.txt
./Whole_Cake_Island/Sweet_City/Casa_de_Pudding/flag.txt
./Sabaody_Archipelago/Grove_109/Casa_de_Shakky/flag.txt
./Sabaody_Archipelago/Grove_1/Casa_de_Shakky/flag.txt
./Sabaody_Archipelago/Grove_80/Casa_de_Rayleigh/flag.txt
./Pirate_Island/Pirates_Ship/Casa_de_GolD_Roger/flag.txt
./Dressrosa/Acacia/Casa_de_Riku_Dold_III/flag.txt
./Dressrosa/Toy_House/Casa_de_Trebol/flag.txt
./Dressrosa/Toy_House/Casa_de_Sugar/flag.txt
./Water_7/Carpenters_Cafe/Casa_de_Paulie/flag.txt
./Water_7/Franky_House/Casa_de_Iceburg/flag.txt
```

Poneglyph

Mediante el buscador de archivos utilizado, se determinó que la ruta de la imagen es la siguiente, ya que fue la única ruta en donde se encontró un poneglyph.zip

```
zoro@2c539a0ca7e0:~$ cd ONEPIECE/
zoro@2c539a0ca7e0:~/ONEPIECE$ cd Whole_Cake_Island/
zoro@2c539a0ca7e0:~/ONEPIECE/Whole_Cake_Island$ cd Caramel_Mountain/
zoro@2c539a0ca7e0:~/ONEPIECE/Whole_Cake_Island/Caramel_Mountain$ cd Casa_de_Big_Mom/
zoro@2c539a0ca7e0:~/ONEPIECE/Whole_Cake_Island/Caramel_Mountain/Casa_de_Big_Mom$ ls
poneglyph.zip
```

Después de descargar el .zip y utilizar la flag encontrada como contraseña se pudo obtener la siguiente imágen:

Como se mencionó anteriormente en el repositorio que se proporcionó en clase se brindó una función para poder extraer el texto dentro de las imágenes de los poneglyphs y el resultado fue el siguiente:

```
PS C:\Users\jsken\OneDrive\Escritorio\UVG\CIFRADO DE INFORMACION\CIFRADO_INFO\PROYECTO_DES> & C:/Users/jsken/AppData/Local/Programs/Python/Python311/p
INFO/PROYECTO_DES/read_image.py"
Poneglyph de Zoro
Introduce tu carné para descifrar el mensaje: 21153
b'and created the Baroque Works syndicate in an attempt to bring down the Arabasta Kingdom and claim Pluton, employing Robin to read the Poneglyph'
```

and created the Baroque Works syndicate in an attempt to bring down the Arabasta Kingdom and claim Pluton, employing Robin to read the Poneglyph

Flag

Después de buscar en todos los directorios listados, se determinó que la flag se encontraba en la siguiente ruta:

```
zorc@2c539a0ca7e0:~/ONEPIECE/Whole_Cake_Island/Caramel_Mountain/Casa_de_Big_Mom$ cd ../../../..
zorc@2c539a0ca7e0:~$ cd ONEPIECE/
zorc@2c539a0ca7e0:~/ONEPIECE$ cd Dressrosa/
zorc@2c539a0ca7e0:~/ONEPIECE/Dressrosa$ cd Toy_House/
zorc@2c539a0ca7e0:~/ONEPIECE/Dressrosa/Toy_House$ cd Casa_de_Sugar
zorc@2c539a0ca7e0:~/ONEPIECE/Dressrosa/Toy_House/Casa_de_Sugar$ ls
flag.txt
zorc@2c539a0ca7e0:~/ONEPIECE/Dressrosa/Toy_House/Casa_de_Sugar$ cat flag.txt
87158c3af198fe8f7ef2e302251772f396de084b30830634389d54d6530f8ef099e695f536zorc@2c539a0ca7e0:~/ONEPIECE/Dressrosa/Toy_House/Casa_de_Sugar$
```

Después de encontrar la flag, se extrajo el contenido usando el comando cat:

87158c3af198fe8f7ef2e302251772f396de084b30830634389d54d6530f8ef099e695f536

Descifrado de Flag

Gracias a las instrucciones del proyecto, se sabe que el método de cifrado del reto de zoro es Rompiendo RC4, usando como clave el carnet del estudiante, que en mi caso es 21153. El código desarrollado es el siguiente:

```
from Crypto.Cipher import ARC4

def rc4_decrypt(cipher_hex, key):
    ciphertext = bytes.fromhex(cipher_hex)
    cipher = ARC4.new(key.encode())
    plaintext = cipher.decrypt(ciphertext)
    try:
        return plaintext.decode("utf-8")
    except UnicodeDecodeError:
        return plaintext

cipher_hex =
"87158c3af198fe8f7ef2e302251772f396de084b30830634389d54d6530f8ef099e695f536"
key = "21153"

print(rc4_decrypt(cipher_hex, key))
```

Este script utiliza la biblioteca pycryptodome para implementar el algoritmo de cifrado simétrico RC4. Primero, convierte el texto cifrado en formato hexadecimal a una secuencia de bytes. Luego, se inicializa el cifrador RC4 con la clave proporcionada (el carné) y se descifra el mensaje. Finalmente, se intenta decodificar el texto resultante como UTF-8 para obtener la flag original. Este enfoque permite recuperar correctamente el contenido cifrado siempre que la clave sea la misma utilizada en el proceso de cifrado.

Después de decodificar el resultado fue el siguiente: FLAG_62a86da7a83da28a0ae5fbd3fb7dd024

Challenge 3: Usopp

Gracias a la flag encontrada y descifrada en el reto anterior se pudo ingresar exitosamente en el usuario de usopp.

```
nobody@c474cf265eb3:/home/usopp$ su usopp
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

usopp@c474cf265eb3:~$
```

Mediante el uso del script de python se encontraron los siguientes directorios con archivos:

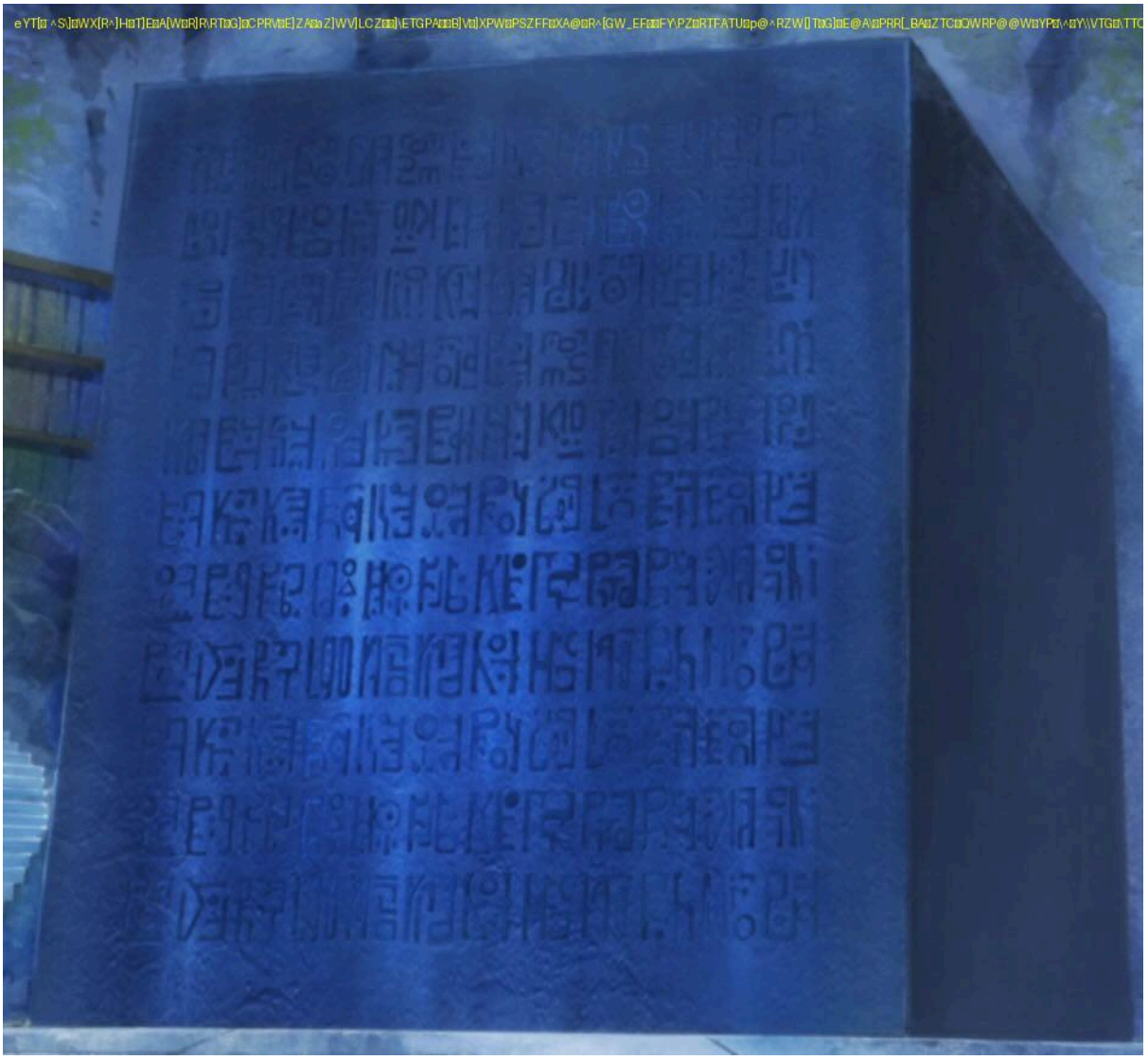
```
usopp@c474cf265eb3:~/ONEPIECE$ python3 listar_archivos_onepiece.py
./listar_archivos_onepiece.py
./Whole_Cake_Island/Caramel_Mountain/Casa_de_Pudding/flag.txt
./Whole_Cake_Island/Whole_Cake_Chateau/Casa_de_Big_Mom/flag.txt
./Sabaody_Archipelago/Grove_66/Casa_de_Keimi/flag.txt
./Sabaody_Archipelago/Grove_80/Casa_de_Rayleigh/poneglyph.zip
./Sabaody_Archipelago/Grove_24/Casa_de_Keimi/flag.txt
./Pirate_Island/Pirates_Ship/Casa_de_GolD_Roger/flag.txt
./Pirate_Island/Pirates_Den/Casa_de_Gol_Roger/flag.txt
./Pirate_Island/Pirates_Cove/Casa_de_GolD_Roger/flag.txt
./Dressrosa/Corrida_Colosseum/Casa_de_Riku_Dold_III/flag.txt
./Dressrosa/Flower_Hill/Casa_de_Rebecca/flag.txt
./Water_7/Carpenters_Cafe/Casa_de_Paulie/flag.txt
./Water_7/Blue_Station/Casa_de_Lucci/flag.txt
```

Poneglyph

Mediante el buscador de archivos utilizado, se determinó que la ruta de la imagen es la siguiente, ya que fue la única ruta en donde se encontró un poneglyph.zip

```
usopp@c474cf265eb3:~/ONEPIECE$ cd Sabaody_Archipelago/
usopp@c474cf265eb3:~/ONEPIECE/Sabaody_Archipelago$ cd Grove_80
usopp@c474cf265eb3:~/ONEPIECE/Sabaody_Archipelago/Grove_80$ cd Casa_de_Rayleigh/
usopp@c474cf265eb3:~/ONEPIECE/Sabaody_Archipelago/Grove_80/Casa_de_Rayleigh$ ls
poneglyph.zip
```

Después de descargar el .zip y utilizar la flag encontrada como contraseña se pudo obtener la siguiente imágen:



Como se mencionó anteriormente en el repositorio que se proporcionó en clase se brindó una función para poder extraer el texto dentro de las imágenes de los poneglyphs y el resultado fue el siguiente:

```
PS C:\Users\jsken\OneDrive\Escritorio\UVG\CIFRADO DE INFORMACION\CIFRADO_INFO\PROYECTO_DES> & C:/Users/jsken/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/jsk
INFO/PROYECTO_DES/read_image.py"
Poneglyph de Usopp
Introduce tu carné para descifrar el mensaje: 21153
b'When Robin finally got the chance to read this Poneglyph, however, she lied about its contents, which caused Crocodile to turn against her because he no longer needed her'
```

When Robin finally got the chance to read this Poneglyph, however, she lied about its contents, which caused Crocodile to turn against her because he no longer needed her

Flag

Después de buscar en todos los directorios listados, se determinó que la flag se encontraba en la siguiente ruta:

```
usopp@c474cf265eb3:~/ONEPIECE$ cd Water_7/
usopp@c474cf265eb3:~/ONEPIECE/Water_7$ cd Blue_Station/
usopp@c474cf265eb3:~/ONEPIECE/Water_7/Blue_Station$ cd casa
bash: cd: casa: No such file or directory
usopp@c474cf265eb3:~/ONEPIECE/Water_7/Blue_Station$
usopp@c474cf265eb3:~/ONEPIECE/Water_7/Blue_Station$ cd Casa_de_Lucci/
usopp@c474cf265eb3:~/ONEPIECE/Water_7/Blue_Station/Casa_de_Lucci$ ls
flag.txt
```

Después de encontrar la flag, se extrajo el contenido usando el comando cat:
a77742694e4801874d3d6938d295db291f6c0a33c94b458c124923ee10aac384e91d55df39

Descifrado de Flag

En el caso de este reto se planteó que la flag está cifrada mediante stream cipher custom, por lo que no se sabe que tipo de cifrado se utilizó, sin embargo dentro del repositorio se encontraba el archivo para cifrar la flag, por lo que se realizó ingeniería inversa para poder descifrar la flag.

```python
import random

# Convierte el texto hexadecimal a bytes una sola vez
ciphertext = bytes.fromhex("a77742694e4801874d3d6938d295db291f6c0a33c94b458c124923ee10aac384e91d55df39")
length = len(ciphertext)

# Prueba de todas las semillas posibles hasta 100000
for seed in range(100000):
    random.seed(seed)
    keystream = bytearray(random.randint(0, 255) for _ in range(length))

    # XOR entre ciphertext y keystream
    plaintext = bytearray(c ^ k for c, k in zip(ciphertext, keystream))

    # Verifica si el texto comienza con "FLAG_"
    if plaintext.startswith(b"FLAG_"):
        print(f"Semilla encontrada: {seed}")
        print(f"Texto descifrado: {plaintext.decode()}")
        break
```

El código desarrollado realiza un ataque por fuerza bruta para encontrar la semilla utilizada durante el cifrado. Primero, convierte el texto cifrado de hexadecimal a bytes. Luego, prueba todas las posibles semillas desde 0 hasta 99,999, generando para cada una un keystream del mismo tamaño que el texto cifrado. Este keystream se genera utilizando la función random.randint, que depende directamente de la semilla establecida. A continuación, el código realiza una operación XOR entre cada byte del texto cifrado y el keystream correspondiente, intentando recuperar el texto original. Si el resultado comienza con el prefijo FLAG_, se asume que se ha encontrado la semilla correcta y se imprime el texto descifrado. Esta técnica aprovecha el hecho de que el keystream puede ser reproducido si se conoce la semilla inicial del generador.

Después de decodificar el resultado fue el siguiente:
FLAG_b3245f0cb6e110404db1d4d0051ab370

Challenge 4: Nami

Gracias a la flag encontrada y descifrada en el reto anterior se pudo ingresar exitosamente en el usuario de nami.

```
PS C:\Users\jsken\OneDrive\Escritorio\UVG\CIFRADO DE INFORMACION\ctf_onepice_symmetric_cipher> docker exec -it nami_challenge bash
nobody@ca4773f3e22f:/home/nami$ su nami
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

nami@ca4773f3e22f:~$
```

Mediante el uso del script de python se encontraron los siguientes directorios con archivos:

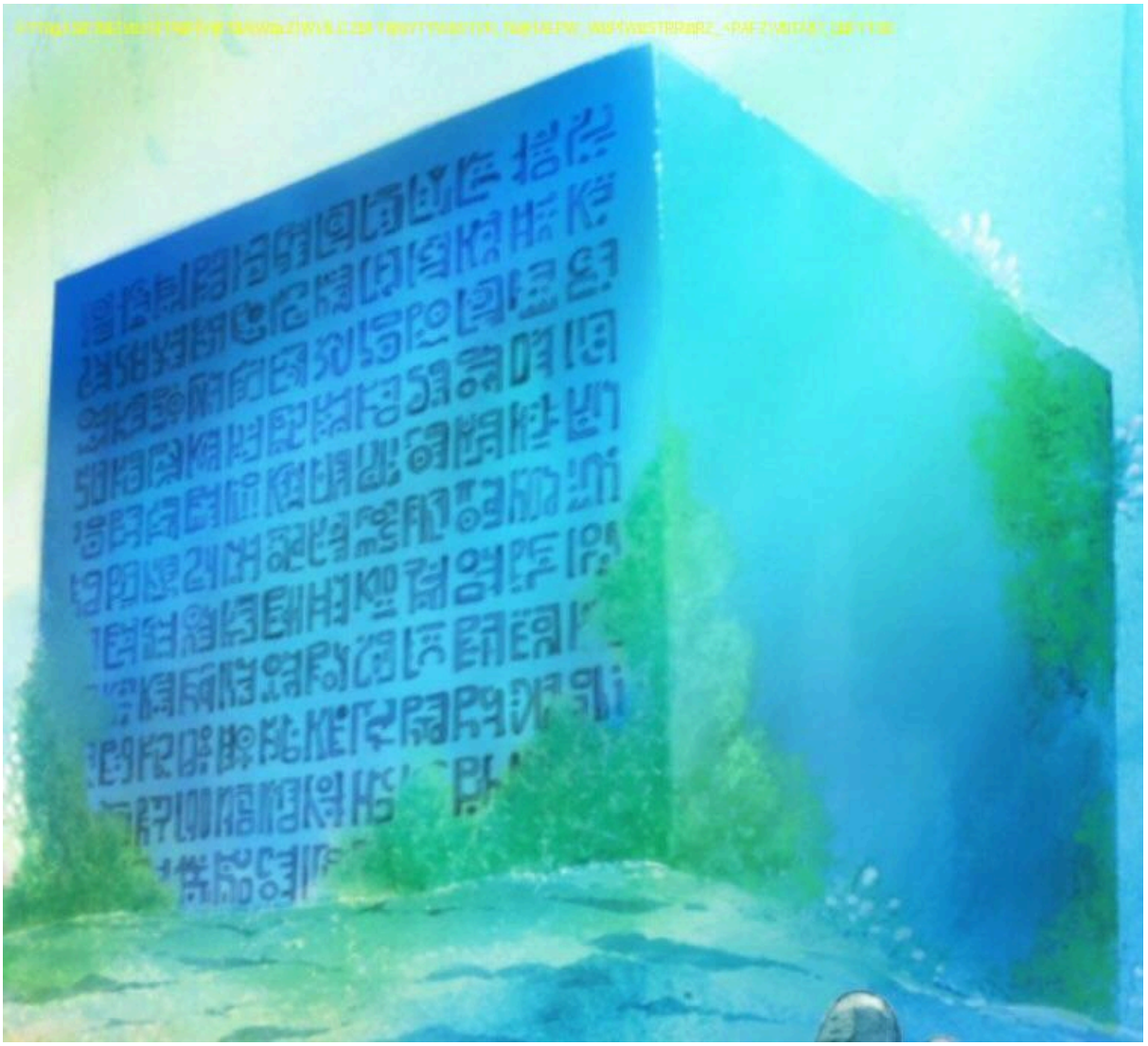
```
nami@ca4773f3e22f:~/ONEPIECE$ python3 listar_archivos_onepiece.py
./listar_archivos_onepiece.py
./Whole_Cake_Island/Caramel_Mountain/Casa_de_Big_Mom/flag.txt
./Whole_Cake_Island/Sweet_City/Casa_de_Big_Mom/flag.txt
./Whole_Cake_Island/Whole_Cake_Chateau/Casa_de_Big_Mom/poneglyph.zip
./Sabaody_Archipelago/Grove_41/Casa_de_Rayleigh/flag.txt
./Sabaody_Archipelago/Grove_24/Casa_de_Keimi/flag.txt
./Sabaody_Archipelago/Grove_24/Casa_de_Rayleigh/flag.txt
./Pirate_Island/Pirates_Ship/Casa_de_GolD_Roger/flag.txt
./Pirate_Island/Pirates_Cove/Casa_de_GolD_Roger/flag.txt
./Dressrosa/Royal_Palace/Casa_de_Riku_Dold_III/flag.txt
./Water_7/Blue_Station/Casa_de_Iceburg/flag.txt
./Water_7/Blue_Station/Casa_de_Paulie/flag.txt
```

Poneglyph

Mediante el buscador de archivos utilizado, se determinó que la ruta de la imagen es la siguiente, ya que fue la única ruta en donde se encontró un poneglyph.zip

```
nami@ca4773f3e22f:~$ cd ONEPIECE/
nami@ca4773f3e22f:~/ONEPIECE$ cd Whole_Cake_Island/
nami@ca4773f3e22f:~/ONEPIECE/Whole_Cake_Island$ cd Whole_Cake_Chateau/
nami@ca4773f3e22f:~/ONEPIECE/Whole_Cake_Island/Whole_Cake_Chateau$ cd Casa_de_Big_Mom/
nami@ca4773f3e22f:~/ONEPIECE/Whole_Cake_Island/Whole_Cake_Chateau/Casa_de_Big_Mom$ ls
poneglyph.zip
```

Después de descargar el .zip y utilizar la flag encontrada como contraseña se pudo obtener la siguiente imágen:



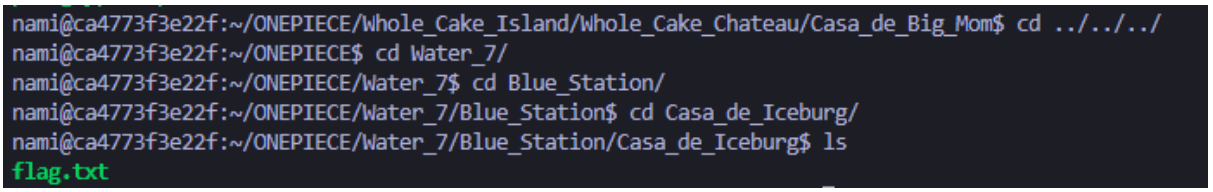
Como se mencionó anteriormente en el repositorio que se proporcionó en clase se brindó una función para poder extraer el texto dentro de las imágenes de los poneglyphs y el resultado fue el siguiente:

```
PS C:\Users\jsken\OneDrive\Escritorio\UVG\CIFRADO DE INFORMACION\CIFRADO_INFO\PROYECTO_DES> & C:/Users/jsken/
INFO/PROYECTO_DES/read_image.py"
Poneglyph de Nami
Introduce tu carné para descifrar el mensaje: 21153
b'the Tomb of the Kings where the Poneglyph was held became unstable and began collapsing around them.'
```

the Tomb of the Kings where the Poneglyph was held became unstable and began collapsing around them

Flag

Después de buscar en todos los directorios listados, se determinó que la flag se encontraba en la siguiente ruta:



Después de encontrar la flag, se extrajo el contenido usando el comando cat: 4ff81a93941b29fe7f5fa5902b7bde49824f5e972fb5cda7318c59b9252101470a9e3ad419

Descifrado de Flag

Gracias a las instrucciones del proyecto, se sabe que el reto de Nami fue cifrado utilizando el algoritmo de flujo ChaCha20, con una clave derivada del carné del estudiante (en mi caso, "21153").

```
from Crypto.Cipher import ChaCha20

def generate_key_nonce(user_id):
    key = (user_id.encode() * 32)[:32]
    nonce = (user_id.encode() * 8)[:8]
    return key, nonce

def chacha20_decrypt(ciphertext_hex, user_id):
    key, nonce = generate_key_nonce(user_id)
    ciphertext = bytes.fromhex(ciphertext_hex)
    cipher = ChaCha20.new(key=key, nonce=nonce)
    plaintext = cipher.decrypt(ciphertext)
    return plaintext.decode(errors="ignore")


ciphertext                                                          =
"4ff81a93941b29fe7f5fa5902b7bde49824f5e972fb5cda7318c59b9252101470a9e3ad419"
user_id = "21153"
```

```
mensaje = chacha20_decrypt(ciphertext, user_id)
print("Texto descifrado:", mensaje)
```

El código desarrollado toma la clave (user_id) y genera tanto la clave de 256 bits como un nonce de 64 bits, repitiendo el user_id las veces necesarias para alcanzar el tamaño requerido por el algoritmo. Posteriormente, se crea un objeto ChaCha20 con estos parámetros y se procede a descifrar el mensaje cifrado, que está representado en formato hexadecimal. Finalmente, el texto se decodifica ignorando errores de caracteres inválidos y se imprime en pantalla el mensaje descifrado.

Después de decodificar el resultado fue el siguiente:
FLAG_0d71649a1533d1540272d16347b1ad18