

**UNIVERSIDAD DEL VALLE DE GUATEMALA**  
Facultad de Ingeniería  
CC3078 – Cifrados de Información  
Sección 10  
Ing. Ludwing Cano



# Proyecto 1

Samuel Argueta - 211024

**GUATEMALA, 23 de abril de 2025**

# ÍNDICE

<b>COMMITS HISTORY.....</b>	<b>3</b>
<b>RESUMEN.....</b>	<b>3</b>
<b>Clone de proyecto. Instalación librerias python.....</b>	<b>4</b>
<b>LUFFY – XOR Cifrado Simple.....</b>	<b>4</b>
Dificultad: Fácil.....	4
Descripción:.....	4
Pasos clave:.....	5
Aprendizaje.....	5
<b>COMANDOS.....</b>	<b>5</b>
Acceso como usuario del reto:.....	5
Bandera encriptada y desencriptada respectivamente.....	8
Texto decifrado de la imagen.....	10
<b>ZORO – Romper cifrado RC4.....</b>	<b>11</b>
<b>USOPP – Stream Cipher Personalizado.....</b>	<b>16</b>
<b>NAMI – ChaCha20 Playground.....</b>	<b>20</b>
Dificultad: Media.....	20
Descripción:.....	20
Pasos clave.....	20
<b>COMANDOS.....</b>	<b>20</b>
Bandera encriptada y desencriptada respectivamente.....	22
Texto decifrado de la imagen.....	23
Decifrado de texto.....	24
<b>Reflexión Final.....</b>	<b>25</b>

# COMMITS HISTORY

En estas capturas de pantalla, se deja registro de los commits que fueron parte del proceso para solución de este proyecto.

Esto pues, en un principio se había realizado un \*fork\* y se trabajó en ese \*fork\*, pero era imposible hacerlo privado, a menos que el repositorio original también lo fuera.

Así que mientras tanto, lo hice en otro repositorio, que sí es privado, y permite hacerlo más propio.

The image contains two side-by-side screenshots of GitHub commit histories. Both are dark-themed and show a list of commits with author, message, timestamp, and a 'Verified' badge for some entries.

**Screenshot 1 (Top):** This shows the commit history for a repository named 'overlord-slayer'. It includes commits from March 30, 2025, and March 29, 2025. Key messages include 'p1: advances part 1', 'p1: part1\_continue', 'p1: adding photos', 'p1: docker compose up', 'p1: Update README.md', and 'p1: hiding some unnecessary files'. A merge commit from 'origin/main' is also shown.

Date	Author	Message	Timestamp
Mar 30, 2025	Overlord-slayer	p1: advances part 1	89608ed1
Mar 30, 2025	Overlord-slayer	p1: part1_continue	05f22130
Mar 30, 2025	Overlord-slayer	p1: adding photos	bcd5dab8
Mar 30, 2025	Overlord-slayer	p1: docker compose up	e9574ec3
Mar 30, 2025	Overlord-slayer	p1: Update README.md	26688669
Mar 30, 2025	Overlord-slayer	p1: hiding some unnecessary files	f4a55d9d
Mar 29, 2025	Overlord-slayer	p1: adds steps of configuration	a13b3f9b
Mar 29, 2025	Overlord-slayer	p1: added file and readme file into the project to store images	61c1c488
Mar 24, 2025	locano	Merge remote-tracking branch 'origin/main'	64282613
Mar 22, 2025	locano	update references xor_cipher	b3d83899

**Screenshot 2 (Bottom):** This shows the commit history for a repository named 'main'. It includes commits from April 6, 2025, and April 5, 2025. Key messages include 'p1: moving from fork, to own branch. Just leaving the time commits', 'p1: cleaning up, just leaving the commits timing', 'p1: text from each image', 'p1: part 4 finished. Just text missing', 'p1: part 3, close to end', 'p1: part3, just the image', 'p1: part 2 in progress', 'p1: part 2, in progress', and 'p1: finished part.1'. All commits are from the same user, 'Overlord-slayer'.

Date	Author	Message	Timestamp
Apr 6, 2025	Overlord-slayer	p1: moving from fork, to own branch. Just leaving the time commits	423ff41c
Apr 6, 2025	Overlord-slayer	p1: cleaning up, just leaving the commits timing	9bae99b1
Apr 5, 2025	Overlord-slayer	p1: text from each image	13fc812
Apr 5, 2025	Overlord-slayer	p1: part 4 finished. Just text missing	f8ee929
Apr 5, 2025	Overlord-slayer	p1: part 3, close to end	a30fc15
Apr 5, 2025	Overlord-slayer	p1: part3, just the image	408665a
Apr 5, 2025	Overlord-slayer	p1: part 2 in progress	67c6583
Apr 5, 2025	Overlord-slayer	p1: part 2, in progress	33ebdec
Apr 5, 2025	Overlord-slayer	p1: finished part.1	a45xa184

## Clone de proyecto. Instalación librerías python

Se hace \*\*clone\*\* del repositorio en una carpeta nueva (con un nombre distintivo que incluye "P1" para diferenciarlo del original).

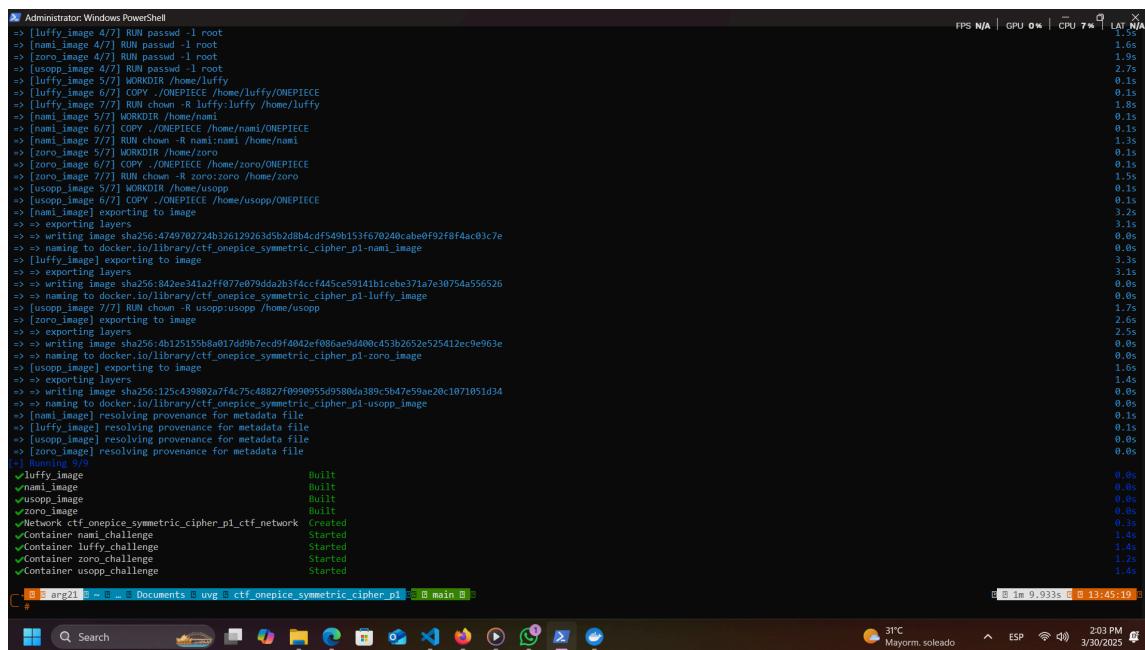
Luego, se ejecuta:

```
```bash
docker compose up -d
````
```

Este comando levanta los contenedores definidos en el archivo `docker-compose.yml`.

- `up` indica que se deben crear e iniciar los contenedores.
- `--d` ejecuta los contenedores en modo \*detached\* (en segundo plano), lo cual permite seguir usando la terminal mientras los servicios corren.

Una vez levantados los 4 contenedores, y generada la data necesaria, se procede a resolver los desafíos.



```
Administrator:Windows PowerShell
[+] [luffy_image 0/7] RUN passed -l root
-> [nami_image 4/7] RUN passed -l root
-> [zoro_image 4/7] RUN passed -l root
-> [usopp_image 4/7] RUN passed -l root
-> [luffy_image 5/7] WORKDIR /home/luffy
-> [luffy_image 6/7] COPY ./ONEPIECE /home/luffy/ONEPIECE
-> [luffy_image 7/7] RUN chown -R luffy:luffy /home/luffy
-> [nami_image 5/7] COPY ./ONEPIECE /home/nami/ONEPIECE
-> [nami_image 6/7] COPY ./ONEPIECE /home/nami/ONEPIECE
-> [nami_image 7/7] RUN chown -R nami:nami /home/nami
-> [zoro_image 5/7] WORKDIR /home/zoro
-> [zoro_image 6/7] COPY ./ONEPIECE /home/zoro/ONEPIECE
-> [zoro_image 7/7] RUN chown -R zoro:zoro /home/zoro
-> [usopp_image 5/7] WORKDIR /home/usopp
-> [usopp_image 6/7] COPY ./ONEPIECE /home/usopp/ONEPIECE
-> [usopp_image 7/7] RUN chown -R usopp:usopp /home/usopp
-> [nami_image] exporting to Image
->> exporting layers
->> writing image sha256:4749702724b326129263d5b2d8b4cdf549b153f670240cabe0f92f8f4ac03c7e
->> naming to docker.io/library/ctf.onepiece_symmetric_cipher_p1-nami_image
-> [luffy_image] exporting to image
->> exporting layers
->> writing image sha256:842ee341a2ff077e079dd2b3f4ccf4d45ce59141b1cbe371a7e30754a556526
->> naming to docker.io/library/ctf.onepiece_symmetric_cipher_p1-luffy_image
-> [usopp_image 7/7] RUN chown -R usopp:usopp /home/usopp
-> [zoro_image] exporting to image
->> exporting layers
->> writing image sha256:4b12515bb0a017dd9b7ecd9f4042ef086ae9d400c453b2652e525412ec9e963e
->> naming to docker.io/library/ctf.onepiece_symmetric_cipher_p1-zoro_image
-> [usopp_image] exporting to image
->> exporting layers
->> writing image sha256:125c439802a7f4c75c48827f0990955d9580da39c5b7e59ae20c1071051d34
->> naming to docker.io/library/ctf.onepiece_symmetric_cipher_p1-usopp_image
-> [nami_image] resolving provenance for metadata file
-> [luffy_image] resolving provenance for metadata file
-> [zoro_image] resolving provenance for metadata file
-> [usopp_image] resolving provenance for metadata file
-> [zoro_image] resolving provenance for metadata file
[+] Running 9/9
✓ luffy_image           Built
✓ nami_image            Built
✓ usopp_image           Built
✓ zoro_image             Built
Network ctf.onepiece_symmetric_cipher_p1_ctf_network Created
✓ Container nami_challenge Started
✓ Container zoro_challenge Started
✓ Container luffy_challenge Started
✓ Container usopp_challenge Started
[!] main B: 31°C
Mayorm. soleado
ESP
3/30/2025
2:03 PM
Im 0.933s
13:45:19
```

## LUFFY – XOR Cifrado Simple

Dificultad: Fácil

Descripción:

Se encontró un archivo `flag.txt` que contenía una cadena hexadecimal encriptada. Aplicando la técnica de \*\*XOR\*\* con una clave basada en el carné del estudiante, se logró desencriptar la bandera.

Pasos clave:

- Se accedió al contenedor `luffy\_challenge`.
- Se utilizó `grep` y `find` para localizar archivos relevantes.
- Se obtuvo el archivo `flag.txt` con la bandera encriptada.
- Se aplicó XOR con la clave personalizada.

\*\*Flag encontrada:\*\* `FLAG\_0a16e1e85da2dc414b4447cd580d63f3`

## Aprendizaje

El cifrado XOR, aunque simple, es efectivo si la clave es suficientemente aleatoria y desconocida. Aquí se reforzó el entendimiento del cifrado simétrico básico.

## COMANDOS

Se ejecuta el contenedor

```bash

docker exec -it luffy\_challenge /bin/bash

```

Esto permite ingresar al contenedor `luffy\_challenge` con una terminal interactiva (`-it`) y utilizar `/bin/bash` como shell. Esto es útil para explorar el sistema de archivos dentro del contenedor y ejecutar comandos.

Tomar en cuenta que se realizó desde windows.

The screenshot shows a Windows command-line interface (CMD) window. The user runs 'docker run' to create a container named 'luffy\_challenge'. Once the container is running, the user enters it using 'docker exec -it'. Inside the container, they run 'id' to check their user ID, which is 0 (root). They then attempt to run 'ls' to list files in the root directory but receive an error message: 'ls: cannot access .: Permission denied'. Finally, they run 'whoami' to verify they are indeed the root user.

```
d----- 3/30/2025 1:45 PM      challenges_volumes
d----- 3/29/2025 7:41 PM      docs
d----- 3/29/2025 7:41 PM      env
d----- 3/30/2025 1:38 PM      resources
d----- 3/30/2025 5:22 PM      resultados
d----- 3/30/2025 1:35 PM      utils
-a---- 3/30/2025 1:38 PM      54 .gitignore
-a---- 3/29/2025 7:41 PM      1548 docker-compose.yml
-a---- 3/29/2025 7:41 PM      3514 generate_challenges.py
-a---- 3/29/2025 7:41 PM      7837 README.md

[+] arge21 ~ % cd ..\Documents\uvwxyz\ctf_onepiece_symmetric_cipher_p1\main\B\B\74\~1\b
# .env\Scripts\activate
[+] arge21 ~ % cd ..\Documents\uvwxyz\ctf_onepiece_symmetric_cipher_p1\main\B\B\74\~1\b
[+] arge21 ~ % docker compose up -d
time="2025-03-29T19:22:34.06+00:00" level=warning msg="C:\Users\arge21\OneDrive\Documents\uvwxyz\ctf_onepiece_symmetric_cipher_p1\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 4/4
  ✓ Container luffy_challenge Started
  ✓ Container namei_challenge Started
  ✓ Container zero_challenge Started
  ✓ Container usopp_challenge Started

[+] arge21 ~ % cd ..\Documents\uvwxyz\ctf_onepiece_symmetric_cipher_p1\main\B\B\74\~1\b
# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS                               NAMES
93c00573092         ctf_onepiece_symmetric_cipher_p1_zoro_image   "tail -f /dev/null"   6 days ago        Up 13 minutes    8000/tcp, 0.0.0.0:2202->22/tcp, 0.0.0.0:8082->8080/tcp   zoro_challenge
49f637b26186        ctf_onepiece_symmetric_cipher_p1-luffy_image   "tail -f /dev/null"   6 days ago        Up 13 minutes    8000/tcp, 0.0.0.0:2201->22/tcp, 0.0.0.0:8081->8080/tcp   luffy_challenge
6b5731438380        ctf_onepiece_symmetric_cipher_p1-usopp_image   "tail -f /dev/null"   6 days ago        Up 13 minutes    8000/tcp, 0.0.0.0:2204->22/tcp, 0.0.0.0:8084->8080/tcp   usopp_challenge
2c90809919c24       ctf_onepiece_symmetric_cipher_p1_namei_image  "tail -f /dev/null"   6 days ago        Up 13 minutes    8000/tcp, 0.0.0.0:2203->22/tcp, 0.0.0.0:8083->8080/tcp   namei_challenge

[+] arge21 ~ % cd ..\Documents\uvwxyz\ctf_onepiece_symmetric_cipher_p1\main\B\B\74\~1\b
# docker exec -it luffy_challenge /bin/bash
OCI runtime exec failed: exec failed: unable to start container process: exec: "\bin\bash": executable file not found in $PATH: unknown
[+] arge21 ~ % cd ..\Documents\uvwxyz\ctf_onepiece_symmetric_cipher_p1\main\B\B\74\~1\b
[+] arge21 ~ % docker exec -it luffy_challenge /bin/bash
nobody@49f637b26186:/home/luffy$ su luffy
Password:
su: Authentication failure
nobody@49f637b26186:/home/luffy$ su luffy
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

luffy@49f637b26186:~$
```

Acceso como usuario del reto:

```bash

su - luffy

# Contraseña: onepiece

```

En caso de requerir instalar herramientas, utilizar

En caso de necesitar herramientas como .zip o alguna extra, se utiliza el comando para poder instalarlo, caso

contrario, no se pueden utilizar las herramientas.

```
```bash
docker exec -u root -it luffy_challenge /bin/bash
```

```

Este comando permite ingresar como \*\*root\*\*, lo cual es útil para instalar herramientas que no estén disponibles por defecto.

Para poder hallar las flags o valores necesarios, se hizo un

```
```bash
grep -r --color=auto "flag" / 2>/dev/null
```

```

Este comando busca recursivamente (con `'-r'`) cualquier texto que contenga la palabra "flag" desde la raíz del sistema. Se ignoran los errores de permisos ('`2>/dev/null`').

En este caso, se utilizó para buscar globalmente los .flag que hubiera, pero me di cuenta que era tan global o tan genérico, que daba resultados un poco raro, como el flag 010002 que no era lo esperado y no tenía nada que ver con el ejercicio.

```

luffy@49637b26186:~/ONEPIECE
lsof -i
<...>
lsof -i
<...>

```

```bash

```
find / -type f \( -name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null
```

```

Busca en todo el sistema archivos con extensiones `.txt`, `.flag`, `.hidden` o `.enc`.

- `‐‐type f`: solo busca archivos.

- `‐‐\(...\)`: agrupa condiciones.

- `‐‐o`: actúa como operador lógico OR.

- `‐‐2>/dev/null`: suprime errores (como falta de permisos).

Viendo que el comando inicial daba resultados muy generico, se probó con este otro. Este otro si dio el resultado esperado, pues hizo mapeo en el área esperada.

Donde estan las carpetas de Luffy.

---

```bash

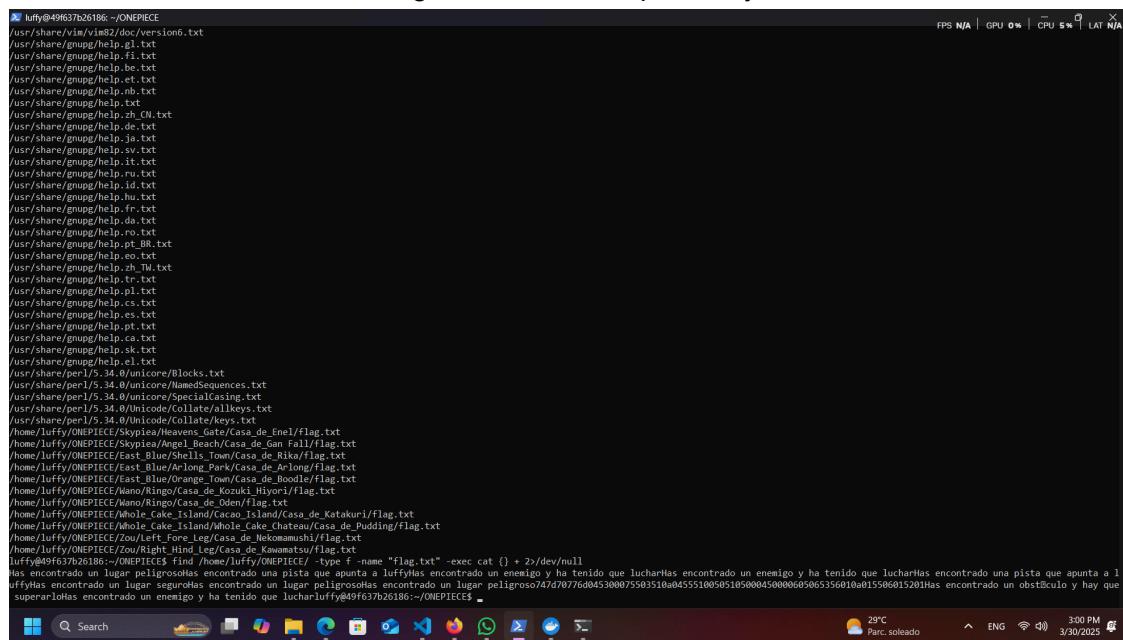
```
find /home/luffy/ONEPIECE/ -type f -name "flag.txt" -exec cat {} + 2>/dev/null
```

```

Busca el archivo `flag.txt` dentro de `/home/luffy/ONEPIECE/` y muestra su contenido:

- `-exec cat {} +`: imprime el contenido de los archivos encontrados.
- `2>/dev/null`: ignora errores de permisos.

Este comando ya permitio ver el contenido de los .flag hallados, ya que el desafio era de Luffy, se hizo unicamente en todos los .flag dentro de la carpeta luffy.



```
luffy@49f637b6186:~/ONEPIECE
/usr/share/vis/1m32/doc/version6.txt
/usr/share/gnupg/help_g1.txt
/usr/share/gnupg/help_f1.txt
/usr/share/gnupg/help_be.txt
/usr/share/gnupg/help_et.txt
/usr/share/gnupg/help_nb.txt
/usr/share/gnupg/help_zh.txt
/usr/share/gnupg/help_zh_Cn.txt
/usr/share/gnupg/help_de.txt
/usr/share/gnupg/help_ja.txt
/usr/share/gnupg/help_sv.txt
/usr/share/gnupg/help_it.txt
/usr/share/gnupg/help_pt.txt
/usr/share/gnupg/help_id.txt
/usr/share/gnupg/help_hu.txt
/usr/share/gnupg/help_fr.txt
/usr/share/gnupg/help_da.txt
/usr/share/gnupg/help_ro.txt
/usr/share/gnupg/help_pt_BR.txt
/usr/share/gnupg/help_tr.txt
/usr/share/gnupg/help_zh_TW.txt
/usr/share/gnupg/help_pl.txt
/usr/share/gnupg/help_cs.txt
/usr/share/gnupg/help_es.txt
/usr/share/gnupg/help_ca.txt
/usr/share/gnupg/help_sk.txt
/usr/share/gnupg/help_el.txt
/usr/share/peri/5.34.0/unicode/Blocks.txt
/usr/share/peri/5.34.0/unicode/NameSequences.txt
/usr/share/peri/5.34.0/unicode/SpecialCasing.txt
/usr/share/peri/5.34.0/unicode/Collate/allkeys.txt
/usr/share/peri/5.34.0/unicode/collate/keys.txt
/home/luffy/ONEPIECE/Skypia/Heavens_Gate/Casa_de_Enel/Flag.txt
/home/luffy/ONEPIECE/Skypia/Angel_Beach/Casa_de_Gan_Fai/Flag.txt
/home/luffy/ONEPIECE/Fairy_Island/Casa_de_Monkey/Dream/Flag.txt
/home/luffy/ONEPIECE/Port_Blu/Blue_Park/Casa_de_Arlong/Flag.txt
/home/luffy/ONEPIECE/last_blue/Orange_Town/Casa_de_Boodle/Flag.txt
/home/luffy/ONEPIECE/Wano/Ringo/Casa_de_Kozuki_Hiyori/Flag.txt
/home/luffy/ONEPIECE/Wano/Ringo/Casa_de_Oden/Flag.txt
/home/luffy/ONEPIECE/Whole_Cake_Island/Cacao_Island/Casa_de_Katakuri/Flag.txt
/home/luffy/ONEPIECE/Whole_Cake_Island/Chopper/Casa_de_Pudding/Flag.txt
/home/luffy/ONEPIECE/Zou/Left_Fore_Leg/Casa_de_Momonoshi/Flag.txt
/home/luffy/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Kowamatsu/Flag.txt
luffy@49f637b6186:~/ONEPIECE$ find /home/luffy/ONEPIECE/ -type f -name "flag.txt" -exec cat {} + 2>/dev/null
Has encontrado un lugar peligrosoHas encontrado una pista que apunta a luffyHas encontrado un enemigo y ha tenido que lucharHas encontrado un enemigo y ha tenido que lucharHas encontrado una pista que apunta a 1
luffy@49f637b6186:~/ONEPIECE$
```

Bandera encriptada y desencrpitada respectivamente

```bash

```
747d70776d045300075503510a0455510050510500045000605065356010a015506015201
FLAG_0a16e1e85da2dc414b4447cd580d63f3
````
```

```bash

```
find / -name "*.zip" 2>/dev/null
````
```

Busca archivos `.zip` en todo el sistema de archivos. Ya que hay una imagen encriptada en uno.

```

luffy@49f637b26186: ~/ONEPIECE
/usr/share/gnupg/help_nb.txt
/usr/share/gnupg/help_zh_CN.txt
/usr/share/gnupg/help_de.txt
/usr/share/gnupg/help_ja.txt
/usr/share/gnupg/help_sv.txt
/usr/share/gnupg/help_it.txt
/usr/share/gnupg/help_ru.txt
/usr/share/gnupg/help_id.txt
/usr/share/gnupg/help_ar.txt
/usr/share/gnupg/help_fr.txt
/usr/share/gnupg/help_da.txt
/usr/share/gnupg/help_ro.txt
/usr/share/gnupg/help_pt_BR.txt
/usr/share/gnupg/help_eo.txt
/usr/share/gnupg/help_zh_TW.txt
/usr/share/gnupg/help_tr.txt
/usr/share/gnupg/help_pl.txt
/usr/share/gnupg/help_cs.txt
/usr/share/gnupg/help_es.txt
/usr/share/gnupg/help_pt.txt
/usr/share/gnupg/help_ca.txt
/usr/share/gnupg/help_el.txt
/usr/share/gnupg/help_el.txt
/usr/share/perl/5.34.0/unicore/Blocks.txt
/usr/share/perl/5.34.0/unicore/NamedSequences.txt
/usr/share/perl/5.34.0/unicore/SpecialCasing.txt
/usr/share/perl/5.34.0/Unicode/Collate/allkeys.txt
/usr/share/perl/5.34.0/Unicode/Collate/keys.txt
/home/luffy/ONEPIECE/skypiea/Heavens_Gate/Casa_de_Enel/flag.txt
/home/luffy/ONEPIECE/skypiea/Angel_Beach/Casa_de_Gan_Fall/flag.txt
/home/luffy/ONEPIECE/east_Blue/Shells_Town/Casa_de_Rika/flag.txt
/home/luffy/ONEPIECE/east_Blue/Arlong_Park/Casa_de_Arlong/flag.txt
/home/luffy/ONEPIECE/east_Blue/Orange_Town/Casa_de_Boodle/flag.txt
/home/luffy/ONEPIECE/Mando/Ringo/Casa_de_Kozuki_Hiyori/flag.txt
/home/luffy/ONEPIECE/Mando/Oden/Casa_de_Katsukuri/flag.txt
/home/luffy/ONEPIECE/Mholic_Cake_Island/Coco_Island/Casa_de_Katakuri/flag.txt
/home/luffy/ONEPIECE/Mholic_Cake_Island/Coco_Island/Casa_de_Katakuri/flag.txt
/home/luffy/ONEPIECE/Mholic_Cake_Chateau/Casa_de_Pudding/flag.txt
/home/luffy/ONEPIECE/Zou/left_Fore_Leg/Casa_de_Nekomamushi/flag.txt
/home/luffy/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Kawamatsu/flag.txt
luffy@49f637b26186:~/ONEPIECE$ find /home/luffy/ONEPIECE/ -type f -name "flag.txt" -exec cat {} + >/dev/null
Has encontrado un lugar peligrosos encontrado una pista que apunta a luffy las encontrado un enemigo y ha tenido que lucharHas encontrado un enemigo y ha tenido que lucharHas encontrado una pista que apunta a luffy las encontrado un lugar seguros encontrado un lugar peligrosas encontrado un lugar peligroso747d0776d045300075503510a04551005105000450000605065356010a015506015201Has encontrado un obstaculo y hay que
luffy@49f637b26186:~/ONEPIECE$ find / -name "poneglyph.txt" 2>/dev/null26186:~/ONEPIECE$ find / -name "poneglyph.txt" 2>/dev/null
luffy@49f637b26186:~/ONEPIECE$ find / -name "poneglyph.txt" 2>/dev/null
luffy@49f637b26186:~/ONEPIECE$ find / -name ""*.zip" 2>/dev/null
/home/luffy/ONEPIECE/Zou/Left_Hind_Leg/Casa_de_Inuarashi/poneglyph.zip
luffy@49f637b26186:~/ONEPIECE$
```

```bash

```
find / -name "*.jpg" -o -name "*.png" 2>/dev/null
```

```

Busca archivos con extensi n `\*.jpg` o `\*.png`.

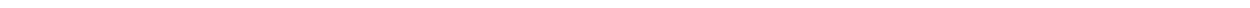
- `‐o`: operador OR para combinar ambas extensiones.

- `‐2>/dev/null`: suprime errores por falta de permisos.

Este era para ver si habia una imagen que se esperaria, tuviera el texto, pero no se hallo nada.

```

luffy@49f637b26186: ~/ONEPIECE
/usr/share/perl/5.34.0/unicore/NamedSequences.txt
/usr/share/perl/5.34.0/unicore/SpecialCasing.txt
/usr/share/perl/5.34.0/Unicode/Collate/allkeys.txt
/usr/share/perl/5.34.0/Unicode/Collate/keys.txt
/home/luffy/ONEPIECE/skypiea/Heavens_Gate/Casa_de_Enel/flag.txt
/home/luffy/ONEPIECE/skypiea/Angel_Beach/Casa_de_Gan_Fall/flag.txt
/home/luffy/ONEPIECE/east_Blue/Shells_Town/Casa_de_Rika/flag.txt
/home/luffy/ONEPIECE/east_Blue/Arlong_Park/Casa_de_Arlong/flag.txt
/home/luffy/ONEPIECE/east_Blue/Orange_Town/Casa_de_Boodle/flag.txt
/home/luffy/ONEPIECE/Mando/Ringo/Casa_de_Kozuki_Hiyori/flag.txt
/home/luffy/ONEPIECE/Mando/Oden/Casa_de_Katsukuri/flag.txt
/home/luffy/ONEPIECE/Mholic_Cake_Island/Coco_Island/Casa_de_Katakuri/flag.txt
/home/luffy/ONEPIECE/Mholic_Cake_Island/Coco_Island/Casa_de_Katakuri/flag.txt
/home/luffy/ONEPIECE/Mholic_Cake_Chateau/Casa_de_Pudding/flag.txt
/home/luffy/ONEPIECE/Zou/left_Fore_Leg/Casa_de_Nekomamushi/flag.txt
/home/luffy/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Kawamatsu/flag.txt
luffy@49f637b26186:~/ONEPIECE$ find /home/luffy/ONEPIECE/ -type f -name "flag.txt" -exec cat {} + >/dev/null
Has encontrado un lugar peligrosos encontrado una pista que apunta a luffy las encontrado un enemigo y ha tenido que lucharHas encontrado un enemigo y ha tenido que lucharHas encontrado una pista que apunta a luffy las encontrado un lugar seguros encontrado un lugar peligrosas encontrado un lugar peligroso747d0776d045300075503510a04551005105000450000605065356010a015506015201Has encontrado un obstaculo y hay que
luffy@49f637b26186:~/ONEPIECE$ find / -name "poneglyph.txt" 2>/dev/null26186:~/ONEPIECE$ find / -name "poneglyph.txt" 2>/dev/null
luffy@49f637b26186:~/ONEPIECE$ find / -name "poneglyph.txt" 2>/dev/null
luffy@49f637b26186:~/ONEPIECE$ find / -name ""*.zip" 2>/dev/null
/home/luffy/ONEPIECE/Zou/left_Hind_Leg/Casa_de_Inuarashi/poneglyph.zip
luffy@49f637b26186:~/ONEPIECE$ find / -name ""*.zip" 2>/dev/null
/home/luffy/ONEPIECE/Zou/left_Hind_Leg/Casa_de_Inuarashi/poneglyph.zip
luffy@49f637b26186:~/ONEPIECE$ unzip -l /home/luffy/ONEPIECE/Zou/left_Hind_Leg/Casa_de_Inuarashi/poneglyph.zip
bash: unzip: command not found
luffy@49f637b26186:~/ONEPIECE$ unzip -l /home/luffy/ONEPIECE/Zou/left_Hind_Leg/Casa_de_Inuarashi/poneglyph.zip
bash: unzip: command not found
luffy@49f637b26186:~/ONEPIECE$ apt-get update
apt-get install unzip
Reading package lists... Done
E: List directory /var/lib/apt/lists/partial is missing. - Acquire (13: Permission denied)
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
luffy@49f637b26186:~/ONEPIECE$ docker cp luffy_challenge:/home/luffy/ONEPIECE/Zou/left_Hind_Leg/Casa_de_Inuarashi/poneglyph.zip C:/Users/arg21/OneDrive/Documents/uvg/ctf_1onepiece_symmetric_cipher_p1/resultados/
bash: docker: command not found
luffy@49f637b26186:~/ONEPIECE$ docker cp luffy_challenge:/home/luffy/ONEPIECE/Zou/left_Hind_Leg/Casa_de_Inuarashi/poneglyph.zip C:/Users/arg21/OneDrive/Documents/uvg/ctf_1onepiece_symmetric_cipher_p1/resultados/
luffy@49f637b26186:~/ONEPIECE$ find / -name "*.jpg" -o -name "*.png" 2>/dev/null
/usr/share/pixmaps/debian-logo.png
/usr/share/info/gnupg-card-architecture.png
/usr/share/icons/hicolor/48x48/apps/gvim.png
/usr/share/icons/locolor/32x32/apps/gvim.png
/usr/share/icons/locolor/16x16/apps/gvim.png
/usr/share/gitweb/static/git-logo.png
/usr/share/gitweb/static/git-favicon.png
luffy@49f637b26186:~/ONEPIECE$
```



---

La contraseña para poder acceder a la imagen del .zip es la misma que la utilizada para ingresar al primer contenedor

```bash

docker cp

luffy\_challenge:/home/luffy/ONEPIECE/Zou/Left\_Hind\_Leg/Casa\_de\_Inuarashi/poneglyph.zip

C:\Users\arg21\OneDrive\Documents\uvg\ctf\_onepiece\_symetric\_cipher\_p1\resultados\part1

```

Copia el archivo `poneglyph.zip` desde la ruta interna del contenedor `luffy\_challenge` hacia la carpeta local `./onepiece`:

- `docker cp`: copia archivos entre contenedor y host.
- `luffy\_challenge:...`: indica el contenedor y la ruta de origen.
- `./resultados/part1`: destino en el host local.



---

Texto decifrado de la imagen

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

---

ejecutar el comando en la raiz del proyecto

```bash

python utils/extract\_text\_from\_image.py

```

Ejecuta un script de Python para extraer texto de una imagen.

- El script probablemente hace uso de herramientas como \*\*Tesseract OCR\*\*.

- La salida suele mostrar texto oculto o codificado en las imágenes relacionadas al reto.

## JUSTIFICACIÓN RESOLUCIÓN LUFFY CHALLENGE

En este caso, como muy bien se pudo apreciar, se realizó un recorrido general. Al principio fue a prueba y error con este primer desafío, pues no se sabía si era la mejor idea, pero se pensó en términos de que en la vida real, uno no debería buscar uno a uno los documentos de manera manual, pues desea saber el problema de manera inmediata. Quizás hizo un salto en cuanto a la búsqueda, pero se hizo en cuanto a la rúbrica o pistas proporcionadas en el README original. Al hacer la búsqueda, y lectura de todos los archivos sugeridos, agilizo el tema de la obtención de los resultados. Una vez realizada la lectura de los archivos y viendo el contenido, se hizo la búsqueda de los .zip, luego de, por cualquier cosa, se hizo la búsqueda global de las imágenes, pensando que estarían libres. Al final, si estaba en un .zip. Una vez obtenida la clave para el siguiente desafío, y el .zip, se trasladó el .zip del contenedor a una carpeta local para poder desencriptar. En este caso, la clave inicial fue proporcionada por el catedrático, así que con la misma, se descomprime el .zip, que requiere una clave. Teniendo la imagen, se utilizo el código provisto en utils, para poder extraer el texto/párrafo de la imagen,

## ZORO – Romper cifrado RC4

\*\*Dificultad:\*\* Media

\*\*Descripción:\*\*

Este reto implicó la lectura de una flag cifrada usando un esquema similar al algoritmo \*\*RC4\*\*. La flag debía descifrarse utilizando una implementación inversa del flujo de cifrado.

\*\*Pasos clave:\*\*

- Se ingresó al contenedor `zoro\_challenge`.
- Se localizó `flag.txt` con contenido cifrado.
- Se aplicaron scripts para emular el flujo de claves y revertir el cifrado tipo RC4.

\*\*Flag encontrada:\*\* `FLAG\_71fb5f88e9a62612bd9c1d030b1cab53`

\*\*Aprendizaje:\*\* Entender cómo funciona un cifrado por flujo como RC4, y cómo puede ser reversible si se conoce el esquema y la clave.

---

## COMANDOS

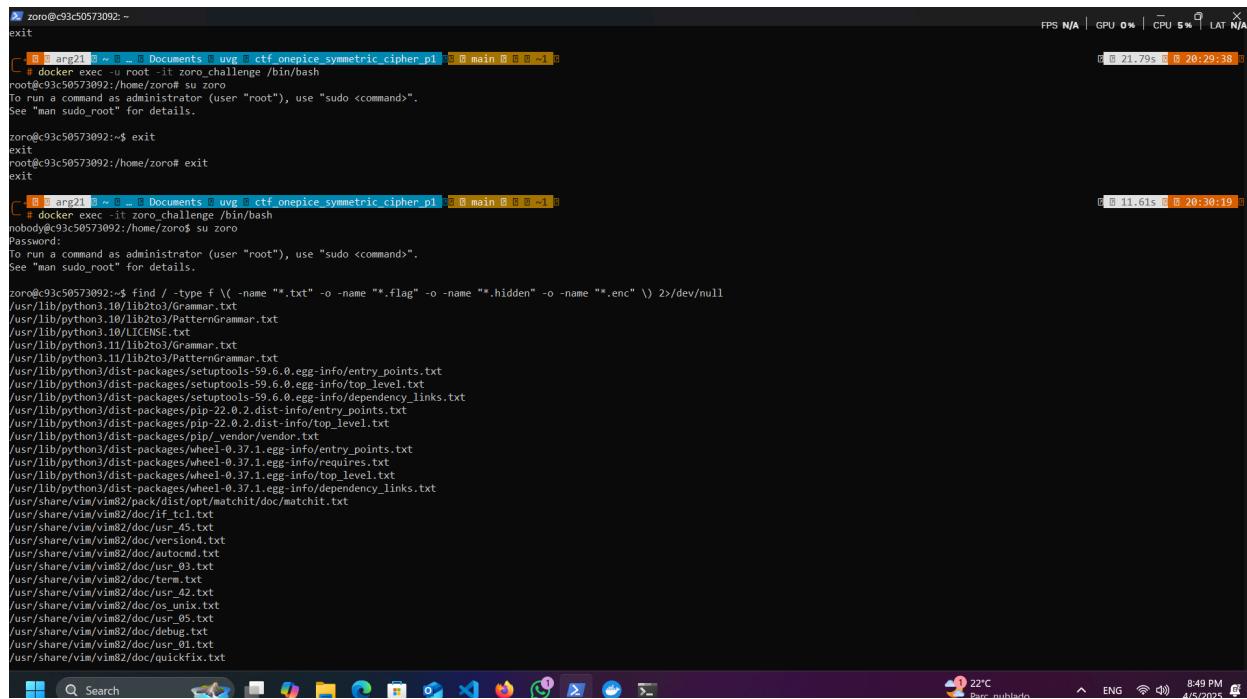
```bash

docker exec -it zoro\_challenge /bin/bash

```

Esto permite ingresar al contenedor `zoro\_challenge` con una terminal interactiva (`-it`) y utilizar `/bin/bash` como shell. Esto es útil para explorar el sistema de archivos dentro del contenedor y ejecutar comandos.

Tomar en cuenta que se realizó desde windows.



```
zoro@e93c50573092:~  
exit  
[1] 21 arg21 1 ~ -> Documents ② uvg ③ ctf onepice_symmetric_cipher.p1 ④ main ⑤ ⑥ ~1 ⑦  
# docker exec -u root -it zoro_challenge /bin/bash  
root@e93c50573092:/home/zoro# su zoro  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
zoro@e93c50573092:~$ exit  
exit  
root@e93c50573092:/home/zoro# exit  
exit  
[1] 21 arg21 1 ~ -> Documents ② uvg ③ ctf onepice_symmetric_cipher.p1 ④ main ⑤ ⑥ ~1 ⑦  
# docker exec -it zoro_challenge /bin/bash  
nobody@e93c50573092:/home/zoro$ su zoro  
Password:  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
zoro@e93c50573092:~$ find / -type f \(-name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null  
/usr/lib/python3.10/lib2to3/grammar.txt  
/usr/lib/python3.10/lib2to3/patternGrammar.txt  
/usr/lib/python3.10/LICENSING  
/usr/lib/python3.11/lib2to3/grammar.txt  
/usr/lib/python3.11/lib2to3/patternGrammar.txt  
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/entry_points.txt  
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/top_level.txt  
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/dependency_links.txt  
/usr/lib/python3/dist-packages/pip-22.0.2.dist-info/entry_points.txt  
/usr/lib/python3/dist-packages/pip-22.0.2.dist-info/top_level.txt  
/usr/lib/python3/dist-packages/pip/_vendor/vendor.txt  
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/entry_points.txt  
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/requirements.txt  
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/top_level.txt  
/usr/share/python3/dist-packages/wheel-0.37.1.egg-info/dependency_links.txt  
/usr/share/vim/vim82/doc/first/matchit/doc/matchit.txt  
/usr/share/vim/vim82/doc/ft_tcl.txt  
/usr/share/vim/vim82/doc/usr_45.txt  
/usr/share/vim/vim82/doc/version4.txt  
/usr/share/vim/vim82/doc/autocmd.txt  
/usr/share/vim/vim82/doc/usr_03.txt  
/usr/share/vim/vim82/doc/term.txt  
/usr/share/vim/vim82/doc/usr_42.txt  
/usr/share/vim/vim82/doc/os_unix.txt  
/usr/share/vim/vim82/doc/usr_05.txt  
/usr/share/vim/vim82/doc/debug.txt  
/usr/share/vim/vim82/doc/usr_01.txt  
/usr/share/vim/vim82/doc/quickfix.txt
```

### Lectura de los archivos .txt o cualquier .\* que pueda servir

```bash

```
find / -type f \(-name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null  
```
```

Busca en todo el sistema archivos con extensiones `.txt`, `.flag`, `.hidden` o `.enc`.

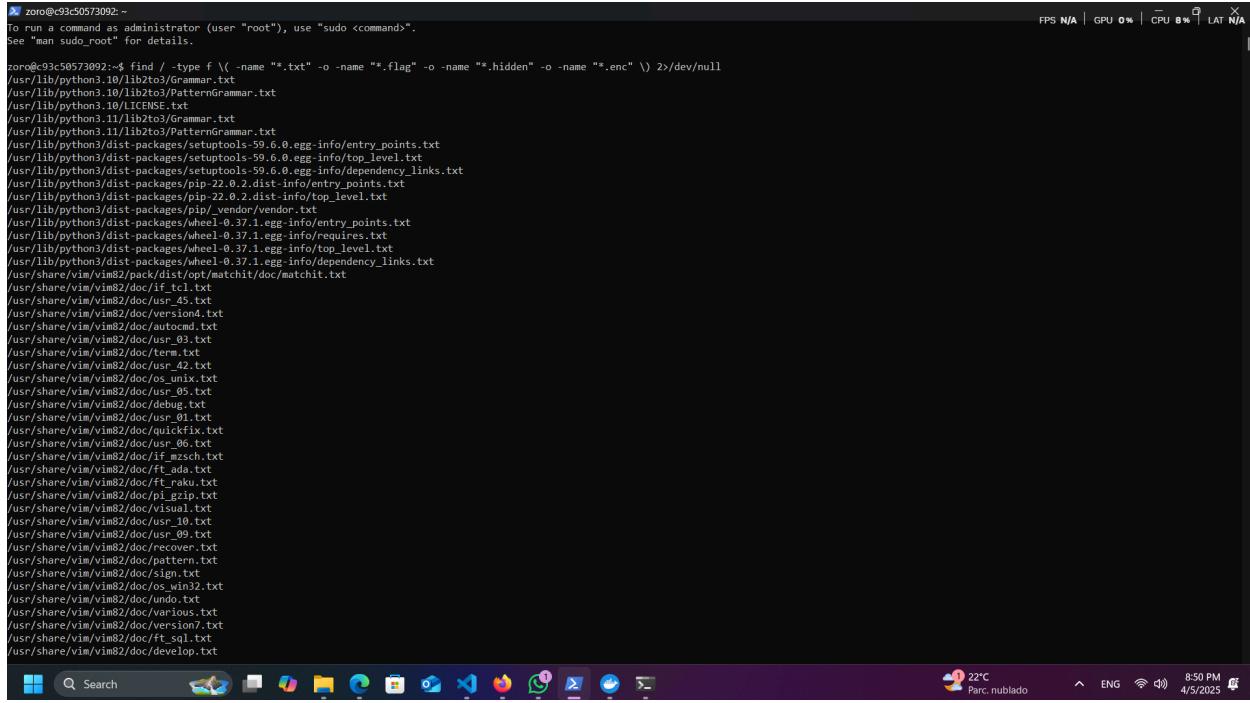
- `-type f`: solo busca archivos.

- `(...\)`: agrupa condiciones.

- `‐o`: actúa como operador lógico OR.

- `2>/dev/null`: suprime errores (como falta de permisos).

Se hizo lo mismo que en el contenedor anterior, esperando a que funcionara y lo hizo, simplificando el trabajo.



```
zoro@e93c50573092:~$ find / -type f \(-name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null
/usr/lib/python3.10/lib2to3/grammar.txt
/usr/lib/python3.10/lib2to3/patternGrammar.txt
/usr/lib/python3.10/LICENSE.txt
/usr/lib/python3.11/lib2to3/grammar.txt
/usr/lib/python3.11/lib2to3/patternGrammar.txt
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/entry_points.txt
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/top_level.txt
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/dependency_links.txt
/usr/lib/python3/dist-packages/pip-22.0.2.dist-info/entry_points.txt
/usr/lib/python3/dist-packages/pip-22.0.2.dist-info/top_level.txt
/usr/lib/python3/dist-packages/pip/_vendor/vendor.txt
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/entry_points.txt
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/requirements.txt
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/top_level.txt
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/dependency_links.txt
/usr/share/vim/vim82/pack/dist/opt/matchit/doc/matchit.txt
/usr/share/vim/vim82/doc/tcl/tcl.txt
/usr/share/vim/vim82/doc/tcl/tk45.txt
/usr/share/vim/vim82/doc/version80.txt
/usr/share/vim/vim82/doc/autocmd.txt
/usr/share/vim/vim82/doc/usr_03.txt
/usr/share/vim/vim82/doc/term.txt
/usr/share/vim/vim82/doc/usr_42.txt
/usr/share/vim/vim82/doc/os_unix.txt
/usr/share/vim/vim82/doc/usr_05.txt
/usr/share/vim/vim82/doc/debug.txt
/usr/share/vim/vim82/doc/usr_01.txt
/usr/share/vim/vim82/doc/quickfix.txt
/usr/share/vim/vim82/doc/usr_06.txt
/usr/share/vim/vim82/doc/tc_mszh.txt
/usr/share/vim/vim82/doc/tc_auds.txt
/usr/share/vim/vim82/doc/filetype.txt
/usr/share/vim/vim82/doc/pl_gzip.txt
/usr/share/vim/vim82/doc/visual.txt
/usr/share/vim/vim82/doc/usr_10.txt
/usr/share/vim/vim82/doc/usr_09.txt
/usr/share/vim/vim82/doc/recover.txt
/usr/share/vim/vim82/doc/pattern.txt
/usr/share/vim/vim82/doc/sign.txt
/usr/share/vim/vim82/doc/os_win32.txt
/usr/share/vim/vim82/doc/undo.txt
/usr/share/vim/vim82/doc/varioust.txt
/usr/share/vim/vim82/doc/version7.txt
/usr/share/vim/vim82/doc/fc_sql.txt
/usr/share/vim/vim82/doc/develop.txt
```

### Este comando sirve para hacer una cat de cada uno, y tenerlo mas ordenado que en el caso anterior. Se ve mejor y legible la verdad

```bash

```
for file in $(find /home/zoro -name "flag.txt"); do
echo "==" $file ==
cat "$file"
echo ""
done
````
```

En esta ocasion, se imprime de una manera mas ordenada, para poder distinguir el texto cifrado, que seria la flag a encontrar.

```
zoro@c93c50573092: ~
/usr/share/perl/5.34.0/unicode/Blocks.txt
/usr/share/perl/5.34.0/unicode/NamedSequences.txt
/usr/share/perl/5.34.0/unicode/SpecialCasing.txt
/usr/share/perl/5.34.0/Unicode/collate/allkeys.txt
/usr/share/perl/5.34.0/Unicode/collate/keys.txt
/home/zoro/ONEPICEE/Skypiea/Shandora/Casa_de_Calgara/flag.txt
/home/zoro/ONEPICEE/Skypiea/Pumpkin_Cafe/Casa_de_Pagaya/flag.txt
/home/zoro/ONEPICEE/Skypiea/Heaven_Gate/Casa_de_Enel/flag.txt
/home/zoro/ONEPICEE/East_Blue/Loguetown/Casa_de_Dragon/flag.txt
/home/zoro/ONEPICEE/East_Blue/Orange_Town/Casa_de_Nomi/flag.txt
/home/zoro/ONEPICEE/East_Blue/Romance_Dawn/Casa_de_Shanks/flag.txt
/home/zoro/ONEPICEE/Wano/Kuri/Casa_de_Kozuki_Momonosuke/flag.txt
/home/zoro/ONEPICEE/Whole_Cake_Island/Caramel_Mountain/Casa_de_Big_Mom/flag.txt
/home/zoro/ONEPICEE/Whole_Cake_Island/Liqueur_Island/Casa_de_Big_Mom/flag.txt
/home/zoro/ONEPICEE/Whole_Cake_Island/Liqueur_Island/Casa_de_Katakuri/flag.txt
/home/zoro/ONEPICEE/Zou/left_Hind_Leg/Casa_de_Inuarashi/flag.txt
zoro@c93c50573092:~$ find /home/luffy/ONEPICEE/ -type f -name "flag.txt" -exec cat {} + >/dev/null
zoro@c93c50573092:~$ for file in $(find /home/zoro -name "flag.txt"); do
echo "==" $file ==
cat "$file"
echo ==
done
== /home/zoro/ONEPICEE/Skypiea/Shandora/Casa_de_Calgara/flag.txt ==
Has encontrado un mapa que apunta a zoro
== /home/zoro/ONEPICEE/Skypiea/Pumpkin_Cafe/Casa_de_Pagaya/flag.txt ==
Has encontrado un tesoro que apunta a zoro
== /home/zoro/ONEPICEE/Skypiea/Heaven_Gate/Casa_de_Enel/flag.txt ==
Has encontrado un obstáculo y hay que superarlo
== /home/zoro/ONEPICEE/East_Blue/Loguetown/Casa_de_Dragon/flag.txt ==
Has encontrado un tesoro que apunta a zoro
== /home/zoro/ONEPICEE/East_Blue/Orange_Town/Casa_de_Nomi/flag.txt ==
Has encontrado un amigo y han decidido viajar juntos
== /home/zoro/ONEPICEE/East_Blue/Romance_Dawn/Casa_de_Shanks/flag.txt ==
Has encontrado un amigo y han decidido viajar juntos
== /home/zoro/ONEPICEE/Wano/Kuri/Casa_de_Kozuki_Momonosuke/flag.txt ==
Has encontrado un lugar que guarda secretos y misterios que apuntan a zoro
== /home/zoro/ONEPICEE/Whole_Cake_Island/Caramel_Mountain/Casa_de_Big_Mom/flag.txt ==
Has encontrado un lugar abandonado
== /home/zoro/ONEPICEE/Whole_Cake_Island/Liqueur_Island/Casa_de_Big_Mom/flag.txt ==
c5e698284ebf7b97371a0e693bc84a6fd1bb76346e9dc6d62d0dff5b8f06d16d53b3d1967
== /home/zoro/ONEPICEE/Whole_Cake_Island/Liqueur_Island/Casa_de_Katakuri/flag.txt ==
Has encontrado un enemigo y ha tenido que huir
== /home/zoro/ONEPICEE/Zou/left_Hind_Leg/Casa_de_Inuarashi/flag.txt ==
Has encontrado un tesoro que apunta a zoro
zoro@c93c50573092:~$
```

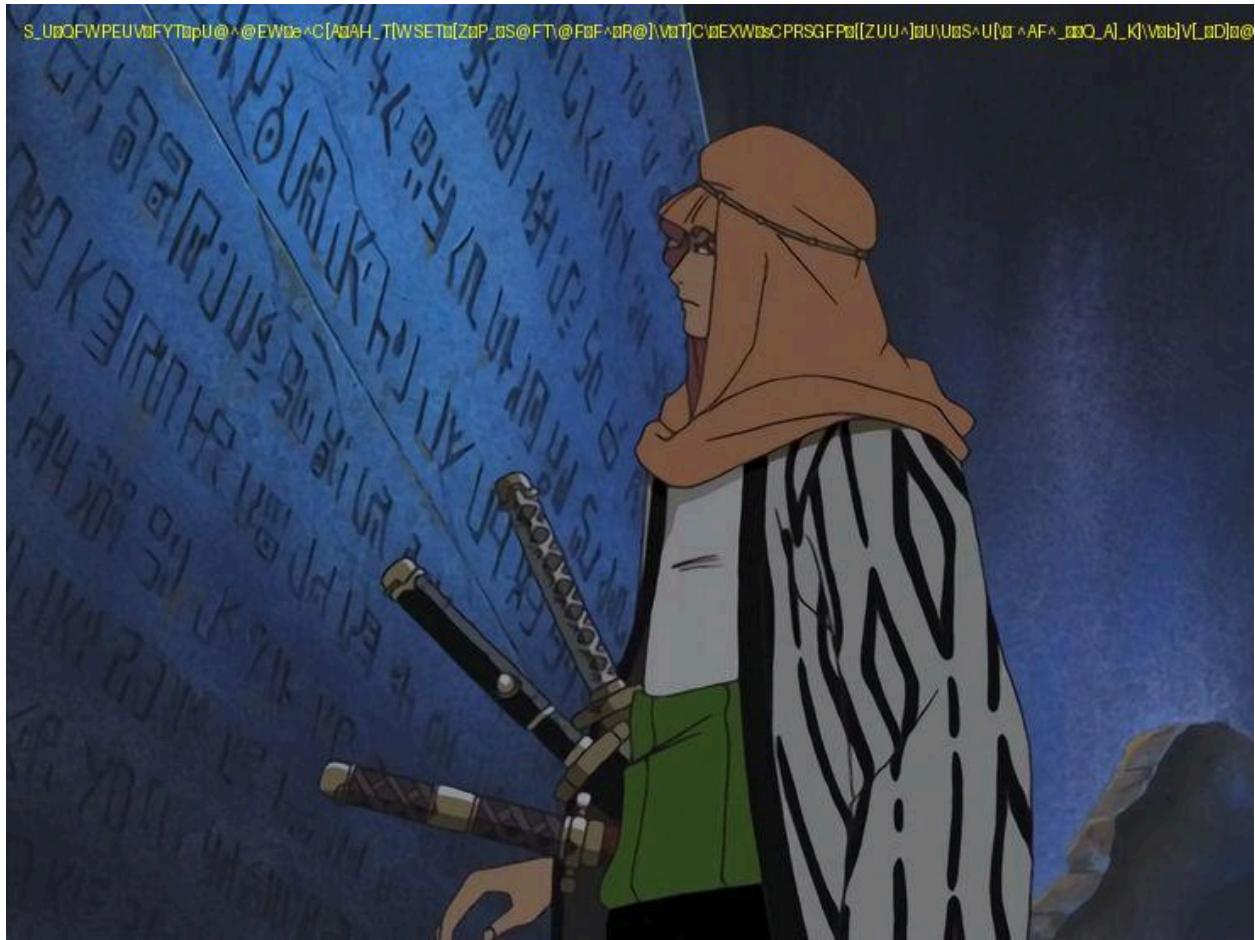
### Bandera encriptada y desencriptada respectivamente

```bash

c5c698284ebf7b97371a0e693bc84a6fd1bb76346e9dc6d62d0dff5b8f06d16d53b3d1967

FLAG\_71fb5f88e9a62612bd9c1d030b1cab53

```



```
```bash
find / -name "*.zip" 2>/dev/null
````
```

Se hace busqueda de los .zip en el sistema. Esto ayudo a hacer una barrido general de todos los archivos

### Copia a carpeta de windows

```
```bash
docker cp
zoro_challenge:/home/zoro/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Inuarashi/poneglyph.zip
"C:\Users\arg21\OneDrive\Documents\uvg\ctf_onepice_symmetric_cipher_p1\resultados\part2"
````
```

Copia el archivo `poneglyph.zip` desde la ruta interna del contenedor `zoro\_challenge` hacia la carpeta local `./onepiece`:

- `docker cp`: copia archivos entre contenedor y host.
- `zoro\_challenge:...`: indica el contenedor y la ruta de origen.
- `./resultados/part2`: destino en el host local.

### Texto decifrado de la imagen

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'

```
---
```

ejecutar el comando en la raiz del proyecto

```
```bash
python utils/extract_text_from_image.py
```

```

Ejecuta un script de Python para extraer texto de una imagen.

- El script probablemente hace uso de herramientas como \*\*Tesseract OCR\*\*.
- La salida suele mostrar texto oculto o codificado en las imágenes relacionadas al reto.

The screenshot shows a terminal window with two command executions. The first execution is at 23:24:40 and the second is at 23:26:23. Both commands are run from the directory 'Documents > uvg > ctf\_onepice\_symmetric\_cipher\_p1 > main'. The output of the first command includes the message 'Introduce tu carnet para descifrar el mensaje: 211024' and the text 'b'and created the Baroque Works syndicate in an attempt to bring down the Arabasta Kingdom and claim Pluto, employing Robin to read the Poneglyph'. The second command's output is partially visible.

```
arg21 ~ ... > Documents > uvg > ctf_onepice_symmetric_cipher_p1 > main > ?1 ~2 -1
$ python utils/extract_text_from_image.py
Introduce tu carnet para descifrar el mensaje: 211024
b'and created the Baroque Works syndicate in an attempt to bring down the Arabasta Kingdom and claim Pluto, employing Robin to read the Poneglyph'

arg21 ~ ... > Documents > uvg > ctf_onepice_symmetric_cipher_p1 > main > ?2 ~2 -1
$ 2.625s 23:26:23
$
```

## JUSTIFICACIÓN RESOLUCIÓN ZORO CHALLENGE

En este caso, una vez viendo que funcionó el tema de hacer un barrido global, se aplica el mismo proceso, esperando que fuera el mismo con resultados exitosos. En este caso funcionó, y se obtuvo la información de la misma manera, se leyeron todos los archivos de pista para ocultar la clave encriptada. Se busca y se traslada el .zip del contenedor a una carpeta local. Se descomprime, se utilizó la clave hallada para este desafío, igual, para descomprimirlo. Con el código de rails, se utilizó para extracción de texto/párrafo en la imagen y wala, desafío terminado.

## USOPP – Stream Cipher Personalizado

**\*\*Dificultad:\*\*** Media

**\*\*Descripción:\*\***

Se presentó un cifrado por flujo con un generador personalizado, no estándar. Requirió analizar la estructura del archivo cifrado y aplicar ingeniería inversa para deducir el método de generación del flujo.

**\*\*Pasos clave:\*\***

- Se accedió al contenedor `usopp\_challenge`.
- Se encontró la flag cifrada.
- Se descifró mediante código propio que replicaba el flujo de cifrado personalizado.

**\*\*Flag encontrada:\*\*** `FLAG\_6290739e295d64e0c37f4d839d2f3182`

**\*\*Aprendizaje:\*\*** Se profundizó en el diseño de cifrados por flujo personalizados y cómo su debilidad radica en la predictibilidad del generador.

---

**## COMANDOS**

```
```bash

```

```
docker exec -it usopp_challenge /bin/bash

```

...

Esto permite ingresar al contenedor `usopp\_challenge` con una terminal interactiva (`-it`) y utilizar `/bin/bash` como shell. Esto es útil para explorar el sistema de archivos dentro del contenedor y ejecutar comandos.

Tomar en cuenta que se realizó desde windows.

```bash

```
find / -type f \(` -name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null  
```
```

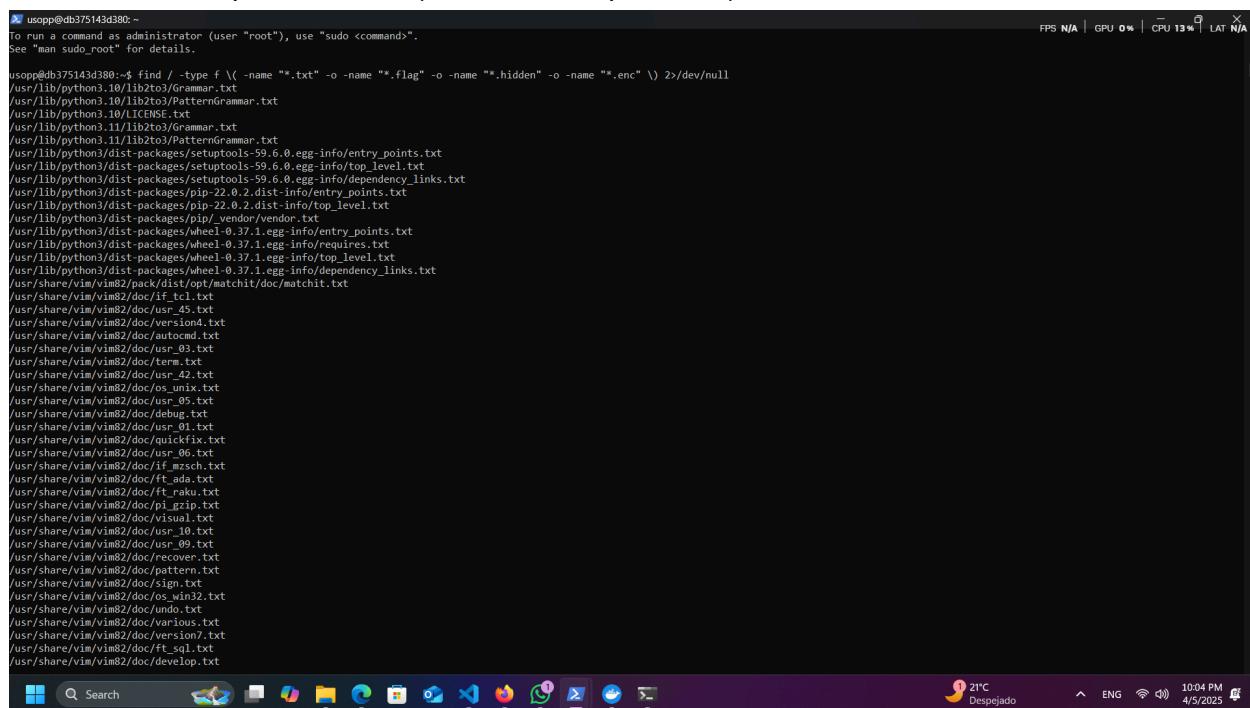
Busca en todo el sistema archivos con extensiones `.txt`, `.flag`, `.hidden` o `.enc`.

- `-type f`: solo busca archivos.

- `(...)`: agrupa condiciones.

- `-o`: actúa como operador lógico OR.

- `2>/dev/null`: suprime errores (como falta de permisos).



```
usopp@db375143d380:~  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
usopp@db375143d380:~$ find / -type f \(` -name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null  
/usr/lib/python3.10/lib2to3/Grammar.txt  
/usr/lib/python3.10/lib2to3/ternarygrammar.txt  
/usr/lib/python3.10/lib2to3/LiCnL.txt  
/usr/lib/python3.11/lib2to3/grammar.txt  
/usr/lib/python3.11/lib2to3/PatternGrammar.txt  
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/entry_points.txt  
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/top_level.txt  
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/dependency_links.txt  
/usr/lib/python3/dist-packages/pip-22.0.2.dist-info/entry_points.txt  
/usr/lib/python3/dist-packages/pip-22.0.2.dist-info/top_level.txt  
/usr/lib/python3/dist-packages/pip/_vendor/vendor.txt  
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/entry_points.txt  
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/requirements.txt  
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/top_level.txt  
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/dependency_links.txt  
/usr/share/vim/vim80/doc/ft_lisppt/matchit/doc/matchit.txt  
/usr/share/vim/vim80/doc/ft_tcl.txt  
/usr/share/vim/vim80/doc/usr_45.txt  
/usr/share/vim/vim80/doc/version4.txt  
/usr/share/vim/vim80/doc/autoloadcmd.txt  
/usr/share/vim/vim80/doc/usr_03.txt  
/usr/share/vim/vim80/doc/term.txt  
/usr/share/vim/vim80/doc/usr_42.txt  
/usr/share/vim/vim80/doc/os_unix.txt  
/usr/share/vim/vim80/doc/usr_05.txt  
/usr/share/vim/vim80/doc/debug.txt  
/usr/share/vim/vim80/doc/usr_01.txt  
/usr/share/vim/vim80/doc/quickfix.txt  
/usr/share/vim/vim80/doc/usr_06.txt  
/usr/share/vim/vim80/doc/filesize.txt  
/usr/share/vim/vim80/doc/ft_mst.txt  
/usr/share/vim/vim80/doc/ft_raku.txt  
/usr/share/vim/vim80/doc/pl_gzip.txt  
/usr/share/vim/vim80/doc/visual.txt  
/usr/share/vim/vim80/doc/usr_10.txt  
/usr/share/vim/vim80/doc/usr_09.txt  
/usr/share/vim/vim80/doc/recover.txt  
/usr/share/vim/vim80/doc/pattern.txt  
/usr/share/vim/vim80/doc/sign.txt  
/usr/share/vim/vim80/doc/os_win32.txt  
/usr/share/vim/vim80/doc/undo.txt  
/usr/share/vim/vim80/doc/undo.txt  
/usr/share/vim/vim80/doc/undo.txt  
/usr/share/vim/vim80/doc/undo.txt  
/usr/share/vim/vim80/doc/undo.txt  
/usr/share/vim/vim80/doc/ft_sml.txt  
/usr/share/vim/vim80/doc/develop.txt
```

### Este comando sirve para hacer una cat de cada uno, y tenerlo más ordenado que en el caso anterior. Se ve mejor y legible la verdad

```bash

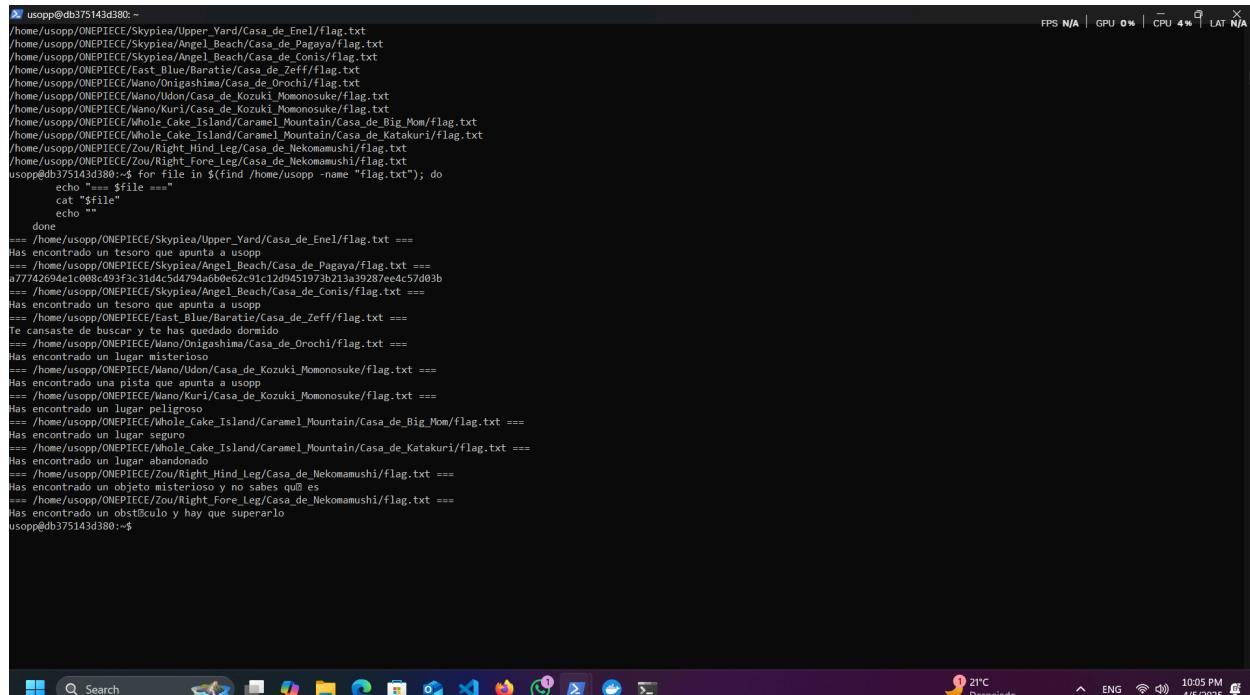
```
for file in $(find /home/usopp -name "flag.txt"); do  
echo "==== $file ===="  
cat "$file"  
echo ""  
done  
```
```

1. `find /home/zoro -name "flag.txt"` busca todos los archivos con ese nombre dentro del directorio del reto.

2. Por cada archivo encontrado, se imprime su ruta (`echo "==== \$file ===="`) y su contenido (`cat "\$file"`).

3. `echo` añade una línea en blanco para mejorar la legibilidad.

Este método permite revisar múltiples archivos `flag.txt` fácilmente, mostrando cada uno separado y bien organizado, ideal cuando hay más de una flag potencial o para validar contenido cifrado.



```
usopp@db375143d380: ~
/home/usopp/ONEPIECE/Skypiea/Upper_Yard/Casa_de_Enel/flag.txt
/home/usopp/ONEPIECE/Skypiea/Angel_Beach/Casa_de_Pagaya/flag.txt
/home/usopp/ONEPIECE/Skypiea/Angel_Beach/Casa_de_Conis/flag.txt
/home/usopp/ONEPIECE/East_Blue/Baratie/Casa_de_Zeff/flag.txt
/home/usopp/ONEPIECE/Wano/Onigashima/Casa_de_Orochi/flag.txt
/home/usopp/ONEPIECE/Wano/Ildon/Casa_de_Kozuki_Momonosuke/flag.txt
/home/usopp/ONEPIECE/Wano/Kuri/Casa_de_Kozuki_Momonosuke/flag.txt
/home/usopp/ONEPIECE/Whole_Cake_Island/Caramel_Mountain/Casa_de_Big_Mom/flag.txt
/home/usopp/ONEPIECE/Whole_Cake_Island/Caramel_Mountain/Casa_de_Katakuri/flag.txt
/home/usopp/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Nekomamushi/flag.txt
/home/usopp/ONEPIECE/Zou/Right_Fore_Leg/Casa_de_Nekomamushi/flag.txt
usopp@db375143d380: $ for file in $(find /home/usopp -name "flag.txt"); do
    echo "----"
    cat "$file"
    echo ""
done
---- /home/usopp/ONEPIECE/Skypiea/Upper_Yard/Casa_de_Enel/flag.txt ===
Has encontrado un tesoro que apunta a usopp
---- /home/usopp/ONEPIECE/Skypiea/Angel_Beach/Casa_de_Pagaya/flag.txt ===
a77742694e1c008c493f3c31d4c5d4794a6b0e62c91c12d9451973b213a39287ee4c57d03b
---- /home/usopp/ONEPIECE/Skypiea/Angel_Beach/Casa_de_Conis/flag.txt ===
Has encontrado un tesoro que apunta a usopp
---- /home/usopp/ONEPIECE/East_Blue/Baratie/Casa_de_Zeff/flag.txt ===
Te has encontrado en un lugar abandonado dormido
---- /home/usopp/ONEPIECE/Wano/Onigashima/Casa_de_Orochi/flag.txt ===
Has encontrado un lugar misterioso
---- /home/usopp/ONEPIECE/Wano/Ildon/Casa_de_Kozuki_Momonosuke/flag.txt ===
Has encontrado una pista que apunta a usopp
---- /home/usopp/ONEPIECE/Wano/Kuri/Casa_de_Kozuki_Momonosuke/flag.txt ===
Has encontrado un lugar peligroso
---- /home/usopp/ONEPIECE/Whole_Cake_Island/Caramel_Mountain/Casa_de_Big_Mom/flag.txt ===
Has encontrado un lugar seguro
---- /home/usopp/ONEPIECE/Whole_Cake_Island/Caramel_Mountain/Casa_de_Katakuri/flag.txt ===
Has encontrado un lugar abandonado
---- /home/usopp/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Nekomamushi/flag.txt ===
Has encontrado un objeto misterioso y no sabes qué es
---- /home/usopp/ONEPIECE/Zou/Right_Fore_Leg/Casa_de_Nekomamushi/flag.txt ===
Has encontrado un obstáculo y hay que superarlo
usopp@db375143d380: $
```

### Bandera encriptada y desencriptada respectivamente

```bash

```
a77742694e1c008c493f3c31d4c5d4794a6b0e62c91c12d9451973b213a39287ee4c57d03b
FLAG_6290739e295d64e0c37f4d839d2f3182
````
```

```bash

```
find / -name "*.zip" 2>/dev/null
````
```

Se hace búsqueda de los .zip en el sistema. Esto ayudó a hacer una barriada general de todos los archivos

### Copia a carpeta de windows

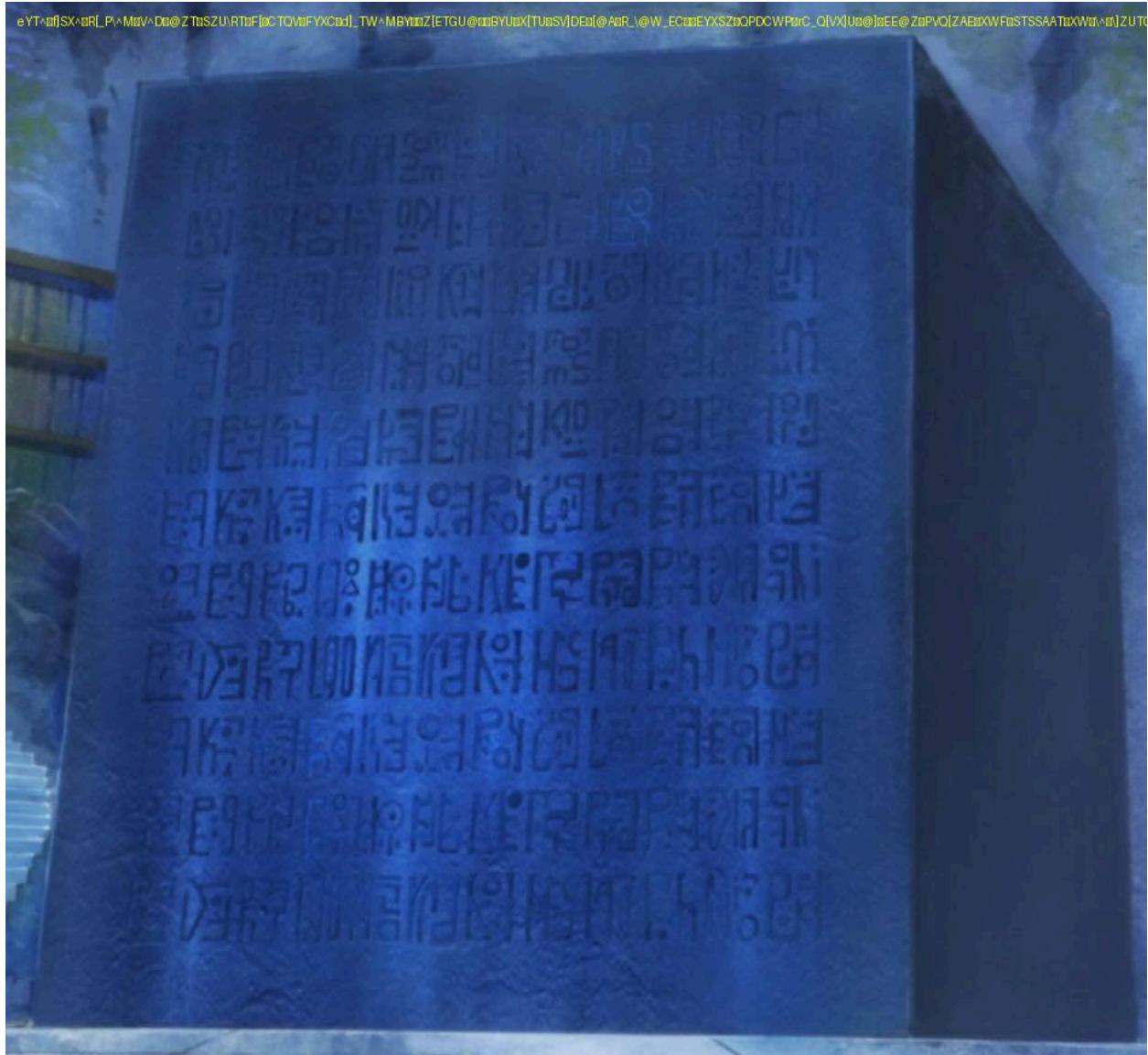
```bash

```
docker cp
```

```
usopp_challenge:/home/usopp/ONEPIECE/Wano/Onigashima/Casa_de_Yamato/poneglyph.zip
"C:\Users\arg21\OneDrive\Documents\uvg\ctf_onepice_symmetric_cipher_p1\resultados\part3"
````
```

Copia el archivo `poneglyph.zip` desde la ruta interna del contenedor `usopp\_challenge` hacia la carpeta local `./onepiece`:

- `docker cp`: copia archivos entre contenedor y host.
- `usopp\_challenge:...`: indica el contenedor y la ruta de origen.
- `./resultados/part3`: destino en el host local.



### ### Texto decifrado de la imagen

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'

---

ejecutar el comando en la raiz del proyecto

```
```bash
```

```
python utils/extract_text_from_image.py
```

```
```
```

Ejecuta un script de Python para extraer texto de una imagen.

- El script probablemente hace uso de herramientas como \*\*Tesseract OCR\*\*.

- La salida suele mostrar texto oculto o codificado en las imágenes relacionadas al reto.

```
arg21 ~ ... > Documents > uvg > ctf_onepice_symmetric_cipher_p1 > main.py ?2 ~2 -1
$ python utils/extract_text_from_image.py
Introduce tu carnet para descifrar el mensaje: 211024
b'When Robin finally got the chance to read this Pomegranate, however, she lied about its contents, which caused Crocodile to turn against her because he no longer needed her'

arg21 ~ ... > Documents > uvg > ctf_onepice_symmetric_cipher_p1 > main.py ?3 ~2 -1
$
```

## JUSTIFICACIÓN RESOLUCIÓN USOPP CHALLENGE

En este caso, como ya se ha probado en los 2 desafíos anteriores, se hizo el mismo proceso. Utilizar la flag desencriptada, para poder ingresar, hacer mapeo global, lectura de, luego mapeo de .zip, para luego trasladar del contenedor al computador local. Una vez tenido ello, extraer la imagen y wala, se terminó rápido.

## NAMI – ChaCha20 Playground

Dificultad: Media

Descripción:

La flag estaba cifrada usando \*\*ChaCha20\*\*, un algoritmo moderno y seguro. Se proporcionaban todos los parámetros necesarios (clave, nonce), por lo que el desafío era más técnico en cuanto al uso correcto del algoritmo.

Pasos clave

- Acceso al contenedor `nami\_challenge`.
- Se localizó el archivo cifrado.
- Se utilizó una implementación correcta de ChaCha20 para descifrar la flag.  
\*\*Flag encontrada:\*\* `FLAG\_c8886f1b7ab0ee2d4e12db4db5d2d4a9`  
\*\*Aprendizaje:\*\* El uso práctico de cifrados modernos como ChaCha20 refuerza conocimientos sobre seguridad real en sistemas actuales.

---

```
```bash
docker exec -it nami_challenge /bin/bash
```
```

Esto permite ingresar al contenedor `nami\_challenge` con una terminal interactiva (`-it`) y utilizar `/bin/bash` como shell. Esto es útil para explorar el sistema de archivos dentro del contenedor y ejecutar comandos.

Tomar en cuenta que se realizó desde windows.

# COMANDOS

```
```bash
find / -type f \(-name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null
```

```

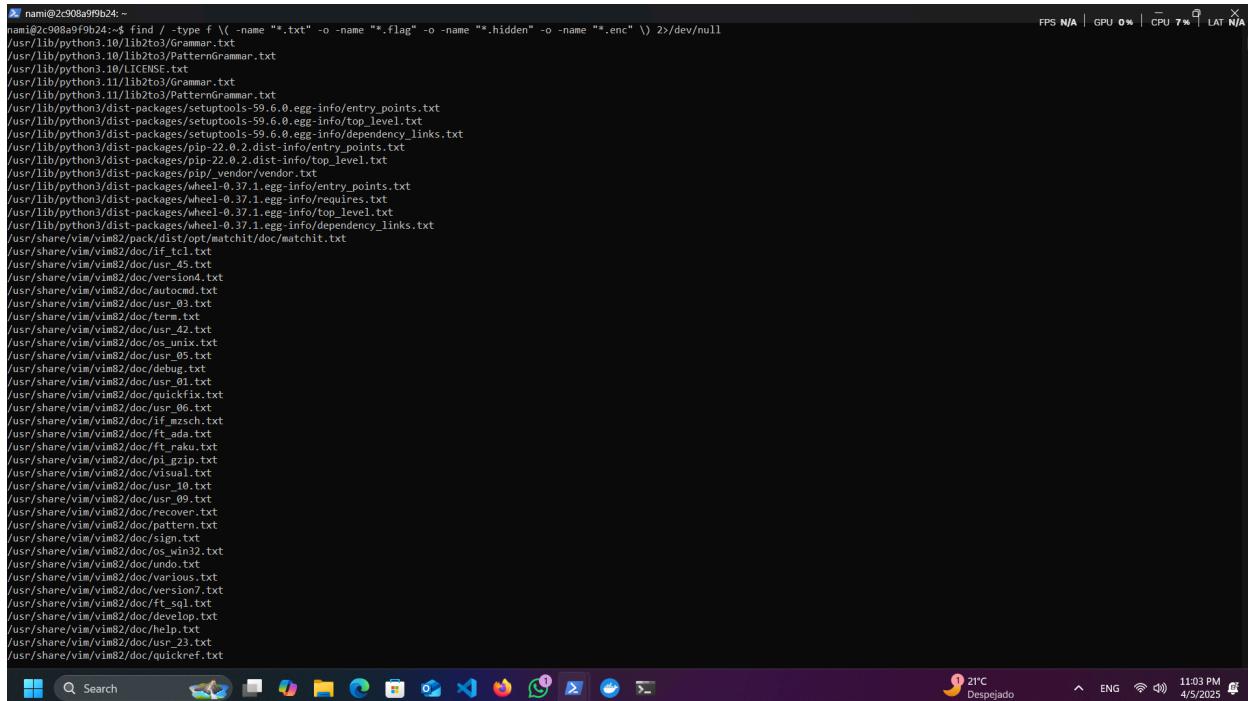
Busca en todo el sistema archivos con extensiones `\*.txt`, `\*.flag`, `\*.hidden` o `\*.enc`.

- `-type f`: solo busca archivos.

- `(...)`: agrupa condiciones.

- `-o`: actúa como operador lógico OR.

- `2>/dev/null`: suprime errores (como falta de permisos).



```
nami@2c908a9fb9b24:~$ find / -type f \(-name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null
/usr/lib/python3.10/lib2to3/grammar.txt
/usr/lib/python3.10/lib2to3/patternGrammar.txt
/usr/lib/python3.10/LICENSE.txt
/usr/lib/python3.11/lib2to3/grammar.txt
/usr/lib/python3.11/lib2to3/patternGrammar.txt
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/entry_points.txt
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/top_level.txt
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/dependency_links.txt
/usr/lib/python3/dist-packages/pip-22.0.2.dist-info/entry_points.txt
/usr/lib/python3/dist-packages/pip-22.0.2.dist-info/top_level.txt
/usr/lib/python3/dist-packages/pip/_vendor/urllib3/urllib3/entry_points.txt
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/entry_points.txt
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/requirements.txt
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/top_level.txt
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/dependency_links.txt
/usr/share/vim/vim82/doc/list/opt/matchit/doc/matchit.txt
/usr/share/vim/vim82/doc/if_tcl.txt
/usr/share/vim/vim82/doc/doc_45.txt
/usr/share/vim/vim82/doc/version4.txt
/usr/share/vim/vim82/doc/autocmd.txt
/usr/share/vim/vim82/doc/usr_03.txt
/usr/share/vim/vim82/doc/term.txt
/usr/share/vim/vim82/doc/usr_42.txt
/usr/share/vim/vim82/doc/doc_04.txt
/usr/share/vim/vim82/doc/usr_05.txt
/usr/share/vim/vim82/doc/debug.txt
/usr/share/vim/vim82/doc/usr_01.txt
/usr/share/vim/vim82/doc/quickfix.txt
/usr/share/vim/vim82/doc/usr_06.txt
/usr/share/vim/vim82/doc/if_mszh.txt
/usr/share/vim/vim82/doc/doc_ada.txt
/usr/share/vim/vim82/doc/ft_raku.txt
/usr/share/vim/vim82/doc/pl_gzip.txt
/usr/share/vim/vim82/doc/visual.txt
/usr/share/vim/vim82/doc/usr_10.txt
/usr/share/vim/vim82/doc/usr_09.txt
/usr/share/vim/vim82/doc/recover.txt
/usr/share/vim/vim82/doc/undotree.txt
/usr/share/vim/vim82/doc/syntax.txt
/usr/share/vim/vim82/doc/os_win32.txt
/usr/share/vim/vim82/doc/undo.txt
/usr/share/vim/vim82/doc/variouse.txt
/usr/share/vim/vim82/doc/version7.txt
/usr/share/vim/vim82/doc/ft_sql.txt
/usr/share/vim/vim82/doc/develop.txt
/usr/share/vim/vim82/doc/help.txt
/usr/share/vim/vim82/doc/usr_23.txt
/usr/share/vim/vim82/doc/quickref.txt
```

Este comando sirve para hacer una cat de cada uno, y tenerlo mas ordenado que en el caso anterior. Se ve mejor y legible la verdad

```
```bash
```

```
for file in $(find /home/nami -name "flag.txt"); do
echo "==== $file ===="
cat "$file"
echo ""
done
```

```

1. `find /home/zoro -name "flag.txt"` busca todos los archivos con ese nombre dentro del directorio del reto.

2. Por cada archivo encontrado, se imprime su ruta (`echo "==== \$file ===="`) y su contenido (`cat "\$file"`).

3. `echo ""` añade una línea en blanco para mejorar la legibilidad.

Este método permite revisar múltiples archivos `flag.txt` fácilmente, mostrando cada uno separado y bien organizado, ideal cuando hay más de una flag potencial o para validar contenido cifrado.

```

nami@2c908a9f9b24: ~
/usr/share/gnupg/help_pt.txt
/usr/share/gnupg/help_ca.txt
/usr/share/gnupg/help_sk.txt
/usr/share/gnupg/help_el.txt
/usr/share/perl/5.34.0/unicore/Blocks.txt
/usr/share/perl/5.34.0/unicore/NamedSequences.txt
/usr/share/perl/5.34.0/unicore/SpecialCasing.txt
/usr/share/perl/5.34.0/Unicode/collate/allkeys.txt
/usr/share/perl/5.34.0/Unicode/collate/keys.txt
/home/nami/ONEPIECE/Skypeia/Uppr_Yard/Casa_de_Enel/flag.txt
/home/nami/ONEPIECE/Skypeia/Shandora/Casa_de_Montblanc_Norland/flag.txt
/home/nami/ONEPIECE/East_Blue/Baratie/Casa_de_Zeff/flag.txt
/home/nami/ONEPIECE/East_Blue/Romance_Dawn/Casa_de_Makino/flag.txt
/home/nami/ONEPIECE/East_Blue/Romance_Dawn/Casa_de_Shanks/flag.txt
/home/nami/ONEPIECE/Wano/Onigashima/Casa_de_Kaido/flag.txt
/home/nami/ONEPIECE/Wano/Kuri/Casa_de_Oden/flag.txt
/home/nami/ONEPIECE/Whole_Cake_Island/Sweet_City/Casa_de_Big_Mom/flag.txt
/home/nami/ONEPIECE/Whole_Cake_Island/Caramel_Mountain/Casa_de_Big_Mom/flag.txt
/home/nami/ONEPIECE/Zou/Right_Fore_Leg/Casa_de_Nekomamushi/flag.txt
nami@2c908a9f9b24:~$ for file in $(find /home/nami -name "flag.txt"); do
    echo "===" $file ==="
    cat "$file"
    echo ""
done
== /home/nami/ONEPIECE/Skypeia/Uppr_Yard/Casa_de_Enel/flag.txt ===
Has encontrado un objeto misterioso y no sabes quié
== /home/nami/ONEPIECE/Skypeia/Shandora/Casa_de_Montblanc ===
cat: /home/nami/ONEPIECE/Skypeia/Shandora/Casa_de_Montblanc: No such file or directory
== Norland/flag.txt ===
cat: Norland/flag.txt: No such file or directory
== /home/nami/ONEPIECE/East_Blue/Baratie/Casa_de_Zeff/flag.txt ===
Has encontrado un lugar misterioso
== /home/nami/ONEPIECE/East_Blue/Romance_Dawn/Casa_de_Makino/flag.txt ===
Has encontrado una pista que apunta a nami
== /home/nami/ONEPIECE/East_Blue/Romance_Dawn/Casa_de_Shanks/flag.txt ===
3fc06ae08a1da3b0dfdd6cdf052435026360b73d778607f9c81ba4c5e3608269653ce10237
== /home/nami/ONEPIECE/East_Blue/Romance_Dawn/Casa_de_Kaido/flag.txt ===
Has encontrado un lugar lleno de secretos y misterios que apuntan a nami
== /home/nami/ONEPIECE/Wano/Kuri/Casa_de_Oden/flag.txt ===
Has encontrado un mapa que apunta a nami
== /home/nami/ONEPIECE/Whole_Cake_Island/Sweet_City/Casa_de_Big_Mom/flag.txt ===
Has encontrado un mapa que apunta a nami
== /home/nami/ONEPIECE/Whole_Cake_Island/Caramel_Mountain/Casa_de_Big_Mom/flag.txt ===
Has encontrado un enemigo y ha tenido que huir
== /home/nami/ONEPIECE/Zou/Right_Fore_Leg/Casa_de_Nekomamushi/flag.txt ===
Has encontrado una pista que apunta a nami
nami@2c908a9f9b24:~$
```

Bandera encriptada y desencrpitada respectivamente

```

```bash
3fc06ae08a1da3b0dfdd6cdf052435026360b73d778607f9c81ba4c5e3608269653ce10237
FLAG_c8886f1b7ab0ee2d4e12db4db5d2d4a9
```
```bash
find / -name "*.zip" 2>/dev/null
```

```

Se hace busqueda de los .zip en el sistema. Esto ayudo a hacer una barrido general de todos los archivos

```

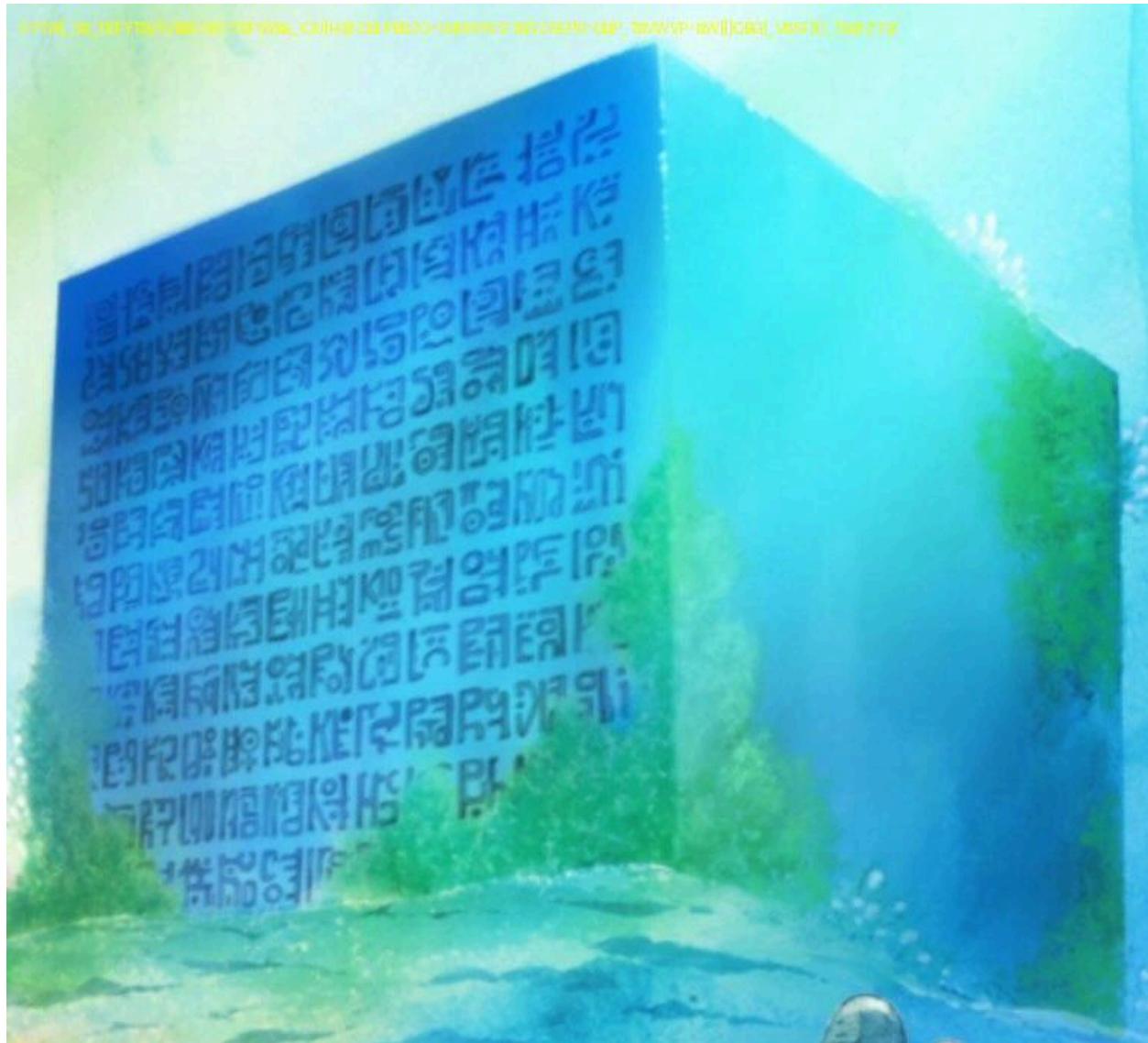
```bash
docker cp
nami_challenge:/home/nami/ONEPIECE/Whole_Cake_Island/Whole_Cake_Chateau/Casa_de_Big_Mom/poneglyph.zip
"C:\Users\arg21\OneDrive\Documents\uvg\ctf_onepice_symmetric_cipher_p1\resultados\part4"
```

```

Copia el archivo `poneglyph.zip` desde la ruta interna del contenedor `nami\_challenge` hacia la carpeta local `./onepiece`:

- `docker cp`: copia archivos entre contenedor y host.
- `nami\_challenge:...`: indica el contenedor y la ruta de origen.

- `./resultados/part4: destino en el host local.



Texto decifrado de la imagen

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

---

ejecutar el comando en la raiz del proyecto

```bash

python utils/extract\_text\_from\_image.py

```

Ejecuta un script de Python para extraer texto de una imagen.

- El script probablemente hace uso de herramientas como \*\*Tesseract OCR\*\*.

- La salida suele mostrar texto oculto o codificado en las imágenes relacionadas al reto.

## Decifrado de texto

En este caso, se utilizo el script provisto por el catedrático, el `extract_text_from_image.py`, ubicado en `utils`.

```
from PIL import Image
import piexif
from luffy_xor import xor_cipher

def extraer_texto_metadata(imagen_path):
    # Abrir la imagen
    img = Image.open(imagen_path)

    # Obtener los metadatos EXIF
    exif_dict = piexif.load(img.info.get('exif', b''))

    # Obtener el texto almacenado en 'Artist' (o en el campo que elegimos)
    texto = exif_dict['0th'].get(piexif.ImageIFD.Artist)
    if texto:
        return texto.decode('utf-8')
    return None

# Uso del código para descifrar la imagen
# Ejemplo de uso
image_path = "resultados/part4/poneglyph.jpeg"
student_id = input("Introduce tu carné para descifrar el mensaje: ")
texto_cifrado = extraer_texto_metadata(image_path)
decrypted_text = xor_cipher(texto_cifrado, student_id)
print(decrypted_text)
```

En el `image_path`, se coloca respectivamente la ruta de la imagen de cada uno de los desafíos. En este caso, como se pudo apreciar previamente, se traslado todo lo que se tenia en el contenedor a una carpeta local con el comando descrito. Y bueno, esto nos dió cada uno de los textos que se pudieron apreciar en cada uno de los desafíos previos.

## JUSTIFICACIÓN RESOLUCIÓN NAMI CHALLENGE

En este caso, como ya se ha probado en los 3 desafíos anteriores, se hizo el mismo proceso. Utilizar la flag desencriptada, para poder ingresar, hacer mapeo global, lectura de, luego mapeo de .zip, para luego trasladar del contenedor al computador local. Una vez tenido ello, extraer la imagen y wala, se terminó rápido el último desafío.

## Reflexión Final

Este proyecto permitió aplicar conocimientos prácticos sobre criptografía simétrica, manipulación de contenedores Docker, y técnicas de análisis de archivos cifrados o escondidos. Cada reto presentó un nivel de dificultad creciente, lo cual ayudó a reforzar distintas habilidades:

- Comprender e implementar XOR manualmente.
- Aplicar lógica inversa a cifrados por flujo como RC4.
- Descifrar algoritmos personalizados a través de ingeniería inversa.
- Trabajar con cifrados modernos como ChaCha20 de forma segura.
- Extraer información de imágenes utilizando OCR.

A mí parecer, perdió un poco de dificultad al poder hacer un barrido general haciendo fácil todos los desafíos. Así mismo, siento que quizás sería más entretenido hacerlo de manera inversa, que los estudiantes hagan su propio contenedor para que el catedrático pueda ver las habilidades de seguridad de cada estudiante, por cuestión de tiempo, hacerlo por vídeo. Quizás algo más divertido, seria que fuera 1 por cada, o que fuera provisto por el catedrático un grupo de actividades que fuera seleccionada por alumno para que fuera único y no se pudieran copiar, eso siento que seria más divertido, pero lo de aquí, si me gusto, no lo niego, esta interesante, pero como mencioné, siento que es más divertido hacerlo para.

Honestamente, sirvió demasiado el hacer barrido global, fue más rápido, al principio, si era un poco complicado porque no se sabía que era lo que se debía hacer, pero luego, definiendo bien lo que se debía hacer, o el área de búsqueda, salió rápido. Igual, lo pensé hacer así, porque como medio menciona en el primer desafío, y bueno, también tener pistas ayudó, era más fácil hacer búsqueda global, que uno a uno. El tiempo apremia y poder solucionar o hallar el problema antes de, es muy útil.