

**UNIVERSIDAD DEL VALLE DE GUATEMALA**  
Cifrado de información  
Sección 10  
Ludwing Cano



# Proyecto 1

Cifrados de Flujo – Desafíos en Seguridad

Abner Iván García Alegría 21285

# Instrucciones generales para correr el código

Para empezar vamos a tener que correr algunos comando antes de iniciar, a continuación se muestran los comandos a correr:

- **git clone {dirección del repositorio que se desea clonar}**
- **python -m venv env**
- **.\env\Scripts\activate**
- **pip3 install -r .\resources\requirements.txt**

Al correr todos los comandos en ese orden deberá aparecerte algo como la imagen de abajo

```
c:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado>git clone https://github.com/locano-uvg/ctf_onepice_symmetric_cipher.git
Cloning into 'ctf_onepice_symmetric_cipher'...
remote: Enumerating objects: 103, done.
remote: Counting objects: 100% (103/103), done.
remote: Compressing objects: 100% (56/56), done.
Receiving objects: 100% (103/103), 370.82 KiB | 1.98 MiB/s, done. (from 0)
Resolving deltas: 100% (41/41), done.

c:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado>cd ctf_onepice_symmetric_cipher
c:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado>ctf_onepice_symmetric_cipher>python -m venv env
Error: Command '['C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher\env\Scripts\python.exe', '-Im', 'ensurepip', '--upgrade', '--default-pip]' returned non-zero exit status 1.

c:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado>ctf_onepice_symmetric_cipher>python -m venv env
c:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado>ctf_onepice_symmetric_cipher>.\env\Scripts\activate
(env) C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>pip3 install -r .\resources\requirements.txt
Collecting Pillow
  Downloading pillow-11.1.0-cp311-cp311-win_amd64.whl (2.6 MB)
    2.6/2.6 MB 3.7 MB/s eta 0:00:00
Collecting numpy
  Downloading numpy-1.2.2.4-cp311-cp311-win_amd64.whl (12.9 MB)
    12.9/12.9 MB 2.1 MB/s eta 0:00:00
Collecting pixieF
  Downloading pixieF-1.1.3-py2.py3-none-any.whl (20 kB)
Collecting pyzipper
  Downloading pyzipper-0.3.6-py2.py3-none-any.whl (67 kB)
    67.7/67.7 kB ? eta 0:00:00
Collecting pycryptodome
  Downloading pycryptodome-3.22.0-cp37-ab13-win_amd64.whl (1.8 MB)
    1.8/1.8 MB 2.9 MB/s eta 0:00:00
Collecting pycryptodomex
  Downloading pycryptodomex-3.22.0-cp37-ab13-win_amd64.whl (1.8 MB)
    1.8/1.8 MB 2.9 MB/s eta 0:00:00
Installing collected packages: pycryptodome, pycryptodome, Pillow, pixieF, numpy, pyzipper
Successfully installed Pillow-11.1.0 numpy-1.2.2.4 pixieF-1.1.3 pycryptodome-3.22.0 pycryptodomex-3.22.0 pyzipper-0.3.6
[notice] A new release of pip available: 22.3 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Procedemos ahora a generar los challenges corriendo el archivo, lo pueden correr de la siguiente forma.

Nota: asegúrense de estar dentro de la carpeta `ctf_onepiece_symmetric_cipher`

- **python .\generate\_challenges.py e ingresar mi carné en este caso es 21285**

al correrlo debería aparecer algo como esto

```
Directorio de C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher

31/03/2025  09:26      <DIR>          .
31/03/2025  09:26      <DIR>          ..
31/03/2025  09:25              10 .gitignore
31/03/2025  09:25          1,540 docker-compose.yml
31/03/2025  09:25      <DIR>          docs
31/03/2025  09:26      <DIR>          env
31/03/2025  09:25          3,514 generate_challenges.py
31/03/2025  09:25          7,837 README.md
31/03/2025  09:25      <DIR>          resources
31/03/2025  09:25      <DIR>          utils
31/03/2025  09:25          4 archivos        12,901 bytes
31/03/2025  09:25          6 dirs       14,299,447,296 bytes libres

(env) C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>python .\generate_challenges.py
Ingrese su carné: 21285
Laberinto generado con éxito.
Ruta1: challenges/luffy/ONEPIECE\Sabaody_Archipelago\Grove_24\Casa_de_Shakky
Ruta2: challenges/luffy/ONEPIECE\Alabasta\Rainbase\Casa_de_Vivi
Windows nt
flag location: challenges/luffy/ONEPIECE\Sabaody_Archipelago\Grove_24\Casa_de_Shakky
Laberinto generado con éxito.
Ruta1: challenges/zoro/ONEPIECE\Sabaody_Archipelago\Grove_80\Casa_de_Shakky
Ruta2: challenges/zoro/ONEPIECE\Alabasta\Nanohana\Casa_de_Toto
Windows nt
flag location: challenges/zoro/ONEPIECE\Sabaody_Archipelago\Grove_80\Casa_de_Shakky
Laberinto generado con éxito.
Ruta1: challenges/usopp/ONEPIECE\Alabasta\Katorea\Casa_de_Kohza
Ruta2: challenges/usopp/ONEPIECE\Zou\Right_Hind_Leg\Casa_de_Kawamatsu
Windows nt
flag location: challenges/usopp/ONEPIECE\Alabasta\Katorea\Casa_de_Kohza
Laberinto generado con éxito.
Ruta1: challenges/nami/ONEPIECE\Sabaody_Archipelago\Grove_80\Casa_de_Keimi
Ruta2: challenges/nami/ONEPIECE\Zou\Left_Hind_Leg\Casa_de_Nekomamushi
Windows nt
flag location: challenges/nami/ONEPIECE\Sabaody_Archipelago\Grove_80\Casa_de_Keimi
Retos generados con éxito!

(env) C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>
```

Ahora procedemos a correr docker para ello deben correr lo siguiente:

→ docker compose up -d

Al correrlo veremos lo siguiente

```
[+] Building 0.0s (0/0) docker:desktop-linuxx9 Semestre\Cifrado\ctf_onepiece_symmetric_cipher>docker compose up -d
[+] Building 177.7s (40/48) FINISHED
=> [zoro_image internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.10kB
=> [zoro_image internal] load metadata for docker.io/library/ubuntu:22.04
=> [luffy_image internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.09kB
=> [usopp_image internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.11kB
=> [nami_image internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.10kB
=> [zoro_image auth] library/ubuntu:pull token for registry-1.docker.io
=> [usopp_image internal] load .dockerrignore
=> => transferring context: 2B
=> [luffy_image internal] load .dockerrignore
=> => transferring context: 2B
=> [nami_image Internal] load .dockerrignore
=> => transferring context: 2B
=> [zoro_image internal] load .dockerrignore
=> => transferring context: 2B
=> [nami_image 1/7] FROM docker.io/library/ubuntu:22.04@sha256:ed1544e454989078f5dec1bfdbd8c5cc9c48e0705d07b678ab6ae3fb61952d2
=> => resolve docker.io/library/ubuntu:22.04@sha256:ed1544e454989078f5dec1bfdbd8c5cc9c48e0705d07b678ab6ae3fb61952d2
=> sha256:9c31e2e37ea1bf5ff7272e979fcac509e225fb853433a6fe21d2fb34e6305 29.54MB
=> sha256:ed1544e454989078f5dec1bfdbd8c5cc9c48e0705d07b678ab6ae3fb61952d2 6.69kB / 6.69kB
=> sha256:33d782143e3a763110570db1673fdad65b17c854190b74a9e99d9e05c5cf 424B / 424B
=> sha256:a24be041d9576937f62435f8564c2:a6e429d2760537b84c50ca59adbc6d212 2.30kB / 2.30kB
=> extracting sha256:9c31e2e37ea1bf5ff7272e979fcac509e225fb853433a6fe21d2fb34e6305
[+] nami_image Internal] load build context
=> transferring context: 60.57kB
[+] luffy_image internal] load build context
=> transferring context: 77.23kB
[+] zoro_image internal] load build context
=> transferring context: 61.84kB
[+] usopp_image internal] load build context
=> transferring context: 14.14kB
[+] nami_image 2/7] RUN apt-get update && apt-get install -y sudo python3 python3-pip python3-dev git vim nano curl tesseract
[+] usopp_image 2/7] RUN useradd -ms /bin/bash usopp && echo "usopp:FLAG_023d2aaaf9036ca35ee54bf8f894d:01" | chpasswd && adduser usopp sudo
[+] zoro_image 3/7] RUN useradd -ms /bin/bash zoro && echo "zoro:FLAG_1912245f868935872c43e1ef726b0c38d" | chpasswd && adduser zoro sudo
[+] luffy_image 3/7] RUN useradd -ms /bin/bash luffy && echo "luffy:onepiece" | chpasswd && adduser luffy sudo
[+] nami_image 3/7] RUN useradd -ms /bin/bash nami && echo "nami:FLAG_5c0cf71f5e6f690b682d3f1328b906" | chpasswd && adduser nami sudo
[+] usopp_image 4/7] RUN passwd -l root
[+] usopp_image 5/7] WORKDIR /home/usopp

=> [zoro_image 5/7] WORKDIR /home/zoro
=> [nami_image 5/7] WORKDIR /home/nami
=> [luffy_image 5/7] WORKDIR /home/luffy
[+] zoro_image 6/7] COPY ./ONEPIECE /home/zoro/ONEPIECE
=> [usopp_image] exporting image
=> exporting layers
=> writing image sha256:a46ae952f7346a482bed9c1f39c47dd7040d27d69a8498d39bfd6dbd9213ed
=> naming to docker.io/library/ctf_onepiece_symmetric_cipher-usopp_image
=> [nami_image 6/7] COPY ./ONEPIECE /home/nami/ONEPIECE
=> [luffy_image 6/7] COPY ./ONEPIECE /home/luffy/ONEPIECE
[+] zoro_image 7/7] RUN chown -R zoro:zoro /home/zoro
=> [nami_image 7/7] RUN chown -R nami:nami /home/nami
=> [luffy_image 7/7] RUN chown -R luffy:luffy /home/luffy
[+] zoro_image exporting to image
=> exporting layers
=> writing image sha256:6bc4b6ea44a3fafe40fa348aa14b021b3f4b5ac1145523092b5a5831bccb9c4a
=> naming to docker.io/library/ctf_onepiece_symmetric_cipher-zoro_image
=> [nami_image] exporting to image
=> exporting layers
=> writing image sha256:e1d0e7b44ff5f36e8ba1e12ba9c282aaeffd/d985fd4294b28d53365473e42f8e
=> naming to docker.io/library/ctf_onepiece_symmetric_cipher-nami_image
[+] luffy_image exporting to image
=> writing image sha256:d9eda3979c456b05c6Fc69e1a10a84de1e5ed60ba953447f54d8c2e4389a7c971
=> naming to docker.io/library/ctf_onepiece_symmetric_cipher-luffy_image
[+] Running 5/3
  Network ctf_onepiece_symmetric_cipher_ctf_network  Created
  Container zoro_challenge  Started
  Container nami_challenge  Started
  Container usopp_challenge  Started
  Container luffy_challenge  Started

```

(env) C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf\_onepiece\_symmetric\_cipher>

## Reto # 1 Luffy

## ✓ Introducción del reto

En este reto se pondrá a prueba las habilidades de análisis criptográfico enfrentándose a un mensaje cifrado mediante la operación XOR, una técnica común en la criptografía básica. Aunque XOR puede ser efectiva si se usa correctamente, presenta vulnerabilidades cuando se reutiliza la clave o se emplean claves débiles. Aprovechando estas debilidades, se aplicarán técnicas de análisis de frecuencia para identificar patrones en el texto cifrado y descifrado del mensaje original. Este tipo de ataque es una forma práctica y fundamental de comprender cómo funcionan los cifrados simples y cómo pueden ser vulnerables frente a un análisis estadístico.

## ✓ Objetivos del reto

- Analizar un mensaje cifrado con XOR.
- Encontrar patrones para descifrar el mensaje original.
- Identificar vulnerabilidades inherentes al uso repetido o débil de claves en cifrados XOR.
- Fortalecer la intuición criptográfica y el pensamiento crítico frente a problemas de seguridad en la información.

## ✓ Flag Encontrada

→ **FLAG\_1912245f868935872c43e1ef726bc38d**

## ✓ Párrafo del poneglyphs

Imagen encontrada



### Párrafo encontrado

→ b'By the time the Straw Hat Pirates arrived at Fish-Man Island, its Road Poneglyph had been removed.'

## ✓ Documentación del proceso, Presentación y claridad

**Paso 1:** Primero tuve que ingresar al primer challenge, ya entrando al primer challenge estuve haciendo pruebas de permisos y me aparecía denegado, entonces en el repo estaba la contraseña y como ingresamos al usuario con todos los permisos para ello corrí los siguientes comandos:

→ **docker exec -it luffy\_challenge bash**

→ **su luffy con el password onepiece**

```
root@06962099a084b:~$ ls
bin boot dev etc home lib lib32 lib64 libx32 luffy_ctf media mnt opt proc root run sbin srv sys tmp usr var
nobody@06962099a084b:~/home/$ cd home/
nobody@06962099a084b:~/home$ cd luffy/
bash: cd: luffy/: Permission denied
nobody@06962099a084b:~/home$ su luffy
Password:
su: Authentication failure
nobody@06962099a084b:~/home$ su luffy
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

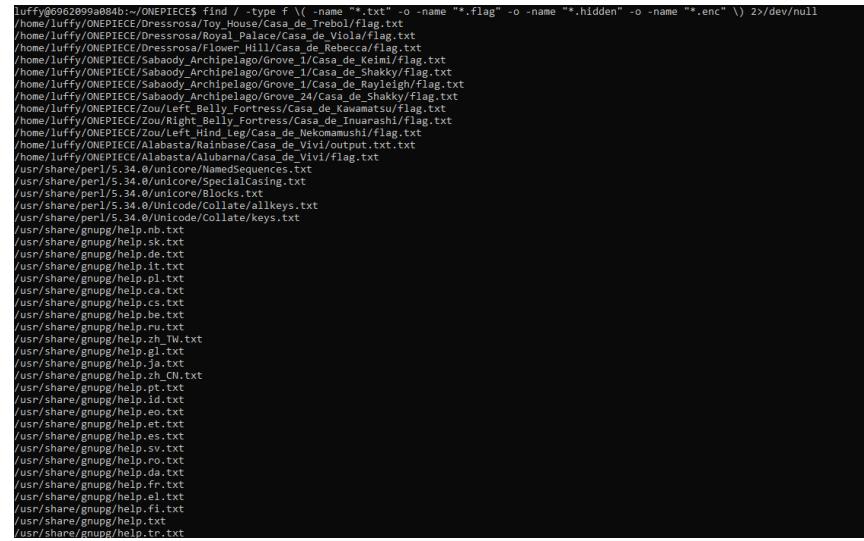
luffy@06962099a084b:~/home$ ls
luffy
luffy@06962099a084b:~/home$ cd luffy/
luffy@06962099a084b:~/luffy$ ls
ONEPIECE
luffy@06962099a084b:~/luffy$ cd ONEPIECE/
luffy@06962099a084b:~/ONEPIECE$ ls
Alabasta Dressrosa Sabaody Archipelago Skypiea Zou
luffy@06962099a084b:~/ONEPIECE$
```

**Paso 2:** Ahora era momento de encontrar la flag, para ello estuve investigando comandos que me pudieran ayudar a encontrarla súper rápido, al principio estuve viendo la manera de como hacer para que me muestre todos los archivos txt o los que tengan flag, vi que habian

muchos entonces encontré un comando en docker el cual me busca todos los archivos que yo desee encontrar el comando es el siguiente:

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

Este comando me ayudó mucho ya que me muestra todos los archivos con las extensiones que busque y hasta me muestra en qué carpeta están, abajo les dejo una imagen de como se mira.

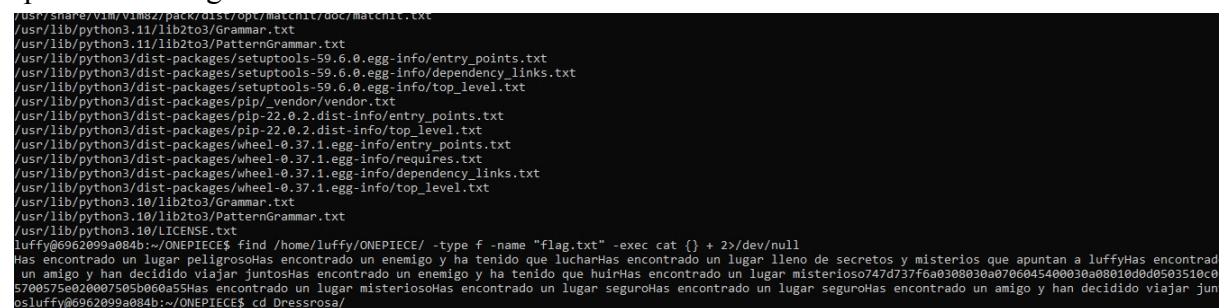


```
luffy@6962099a084b:~/ONEPIECE$ find / -type f \( -name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null
/home/luffy/ONEPIECE/Dressrosa/Toy_House/Casa_de_Trebol/flag.txt
/home/luffy/ONEPIECE/Dressrosa/Royal_Palace/Casa_de_Viola/flag.txt
/home/luffy/ONEPIECE/Sabadoy_Archipelago/Grove_1/Casa_de_Rainbase/output
/home/luffy/ONEPIECE/Sabadoy_Archipelago/Grove_1/Casa_de_Kelmi/flag.txt
/home/luffy/ONEPIECE/Sabadoy_Archipelago/Grove_1/Casa_de_Shaky/flag.txt
/home/luffy/ONEPIECE/Sabadoy_Archipelago/Grove_1/Casa_de_Rayleigh/flag.txt
/home/luffy/ONEPIECE/Sabadoy_Archipelago/Grove_24/Casa_de_Shaky/flag.txt
/home/luffy/ONEPIECE/Zou_Luffy_Belly_Fortress/Casa_de_Kawamatsu/flag.txt
/home/luffy/ONEPIECE/Zou_Ruffy_Belly_Fortress/Casa_de_Inuarashi/flag.txt
/home/luffy/ONEPIECE/Zou_Ruffy_Belly_Fortress/Casa_de_Nekomamushi/flag.txt
/home/luffy/ONEPIECE/Alabasta/Rainbase/Casa_de_Rainbase/output.txt
/home/luffy/ONEPIECE/Alabasta/Aluhama/Casa_de_Yivi/flag.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/unicore/Blocks.txt
/usr/share/perl/5.34.0/Unicode/Collate/allkeys.txt
/usr/share/share/grnupg/Help/unicode
/usr/share/share/grnupg/help_sk.txt
/usr/share/share/grnupg/help_de.txt
/usr/share/share/grnupg/help_it.txt
/usr/share/share/grnupg/help_pl.txt
/usr/share/share/grnupg/help_ca.txt
/usr/share/share/grnupg/help_es.txt
/usr/share/share/grnupg/help_en.txt
/usr/share/share/grnupg/help_zh_TW.txt
/usr/share/share/grnupg/help_ja.txt
/usr/share/share/grnupg/help_zh_CN.txt
/usr/share/share/grnupg/help_pt.txt
/usr/share/share/grnupg/help_id.txt
/usr/share/share/grnupg/help_ar.txt
/usr/share/share/grnupg/help_el.txt
/usr/share/share/grnupg/help_es.txt
/usr/share/share/grnupg/help_sv.txt
/usr/share/share/grnupg/help_ro.txt
/usr/share/share/grnupg/help_da.txt
/usr/share/share/grnupg/help_fr.txt
/usr/share/share/grnupg/help_it.txt
/usr/share/share/grnupg/help_fi.txt
/usr/share/share/grnupg/help_pt.txt
/usr/share/share/grnupg/help_tr.txt
```

**Paso 3:** Ya teniendo todos los archivos vemos que hay muchos llamados flags.txt estuve haciendo un cat por cada uno pero me aparecían textos y no lo que estaba buscando que era una serie de números, para ello decidí de nuevo buscar otro comando que me ayudara a correr todos de nuevo sin ir uno por uno, entonces encontré un comando super magico el cual me lee todos los archivos de una vez y me los muestra todos juntos el comando que corri es el siguiente:

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

este comando me ayuda a leer y mostrar todos los archivos llamados flag.txt y lo que me gusta de este comando mágico es que le agrega haz cuando empieza otro archivo es decir en cada has que encuentres significa que es un archivo flag.txt el que está leyendo, al correrlo aparece de la siguiente manera:



```
luffy@6962099a084b:~/ONEPIECE$ find /home/luffy/ONEPIECE/ -type f -name "flag.txt" -exec cat {} + 2>/dev/null
Has encontrado un lugar peligrosoHas encontrado un enemigo y ha tenido que lucharHas encontrado un lugar lleno de secretos y misterios que apuntan a luffyHas encontrado un amigo y han decidido viajar juntosHas encontrado un enemigo y ha tenido que huirHas encontrado un lugar misterioso747d737f6a0308030a08010d0d0503510c05700575e020007505b06a55Has encontrado un lugar misteriosoHas encontrado un lugar seguroHas encontrado un lugar seguroHas encontrado un amig...Has encontrado un lugar seguroHas encontrado un amigo y han decidido viajar jun...osluffy@6962099a084b:~/ONEPIECE$ cd Dressrosa/
```

vemos todos los archivos flag.txt leídos y mostrados, pero hay algo raro vemos en el texto un montón de números raros lo cual eso era lo que estabamos buscando para encontrar la flag

→747d737f6a0308030a0706045400030a08010d0d0503510c065700575e020007505b060a55

**Paso 4:** Ya que encontramos ese número raro ahora debemos de ver la manera de como obtener algo más o menos legible ya que ahí estaba encriptado, entonces para este reto debíamos de desencriptar ese número extraño usando nuestro carnet que el mio es 21285 para ello me ayudé de un programa en python el cual me ayuda a hacer el XOR de ese número raro con mi carnet, a continuación les muestro una imagen del programa con su resultado.

```

Xor.py > ...
Xor.py > ...
1 def xor_decrypt(hex_string, key):
2     # Convertir La cadena hexadecimal en bytes
3     encrypted_bytes = bytes.fromhex(hex_string)
4
5     # Convertir el número de carné en bytes (repetir si es necesario)
6     key_bytes = str(key).encode()
7     key_length = len(key_bytes)
8
9     # Aplicar XOR
10    decrypted_bytes = bytes(b ^ key_bytes[i % key_length] for i, b in enumerate(encrypted_bytes))
11
12    # Convertir el resultado a una cadena ASCII
13    return decrypted_bytes.decode('errors', 'ignore')
14
15 hex_string = "747d737f6a0308030a0706045400030a08010d0d0503510c065700575e020007505b000a55"
16 carnet = 21285
17
18 decrypted_text = xor_decrypt(hex_string, carnet)
19 print("Texto desencriptado:", decrypted_text)

```

PROBLEMS OUTPUT PORTS TERMINAL GITLENS COMMENTS

PS C:\Users\Personal\Documents\Universidad\9 Semestre\DisenoEI> & "C:/Program Files/Python311/python.exe" "c:/Users/Personal/Documents/Universidad/9 Semestre/DisenoEI/xor.py"

Texto desencriptado: FLAG\_1912245f868935872c43e1ef726bc38d

PS C:\Users\Personal\Documents\Universidad\9 Semestre\DisenoEI>

Live Share ▾ 0 Live Share ▾

Ln 19, Col 46 Spaces: 4 UTF-8 CRLF Python 3.11.0 Background 10:32 31/03/2025

### Implementación:

El programa lo que hago es que creo una función llamada xor\_decrypt el cual lo que hace es convertir hexadecimal en bytes es decir ese número raro que encontramos, luego el número de carné igual lo convertimos a bytes y repetirlo para que tenga la misma longitud a la cadena hexadecimal es decir el número raro, luego aplico un Xor entre estas dos cadenas de bytes y luego el resultado lo convierto a una cadena ASCII e imprimo el resultado mostrado anteriormente la cual ya es la flag que utilizare para el siguiente reto.

→Mensaje descifrado: FLAG\_1912245f868935872c43e1ef726bc38d

**Paso 5:** Ahora el próximo objetivo es encontrar la imagen, pero ya sabiendo el comando de como buscar archivos en las carpetas ya es mucho más fácil ya que ahora busque por extensión png, jpg, zip, etc para poder ver dónde es que estaba la imagen, para ello utilice el siguiente comando:

→**find / -type f \(-name "\*.jpg" -o -name "\*.png" -o -name "\*.jpeg" -o -name "\*.gif" -o -name "\*.bmp" -o -name "\*.zip" \) 2>/dev/null**

Al correr el comando me aparece lo siguiente:

```

luffy@6962099a084b:~/ONEPIECE$ find / -type f \(-name "*.jpg" -o -name "*.png" -o -name "*.jpeg" -o -name "*.gif" -o -name "*.bmp" -o -name "*.zip" \) 2>/dev/null
/home/luffy/ONEPIECE/Alabasta/Rainbase/Casa_de_Vivi/poneglyph.zip
/usr/share/info/gnupg-module-overview.png
/usr/share/info/gnupg-cand-architecture.png
/usr/share/pixmaps/debian-logo.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@6962099a084b:~/ONEPIECE$

```

Vemos que hay un zip el cual tiene el mismo nombre como el requerimiento que nos piden, entonces ahí está la imagen escondida, debo de ver como descomprimir el zip.

**Paso 6:** Ahora toca descomprimir el zip, probe con unzip pero me causaba error ya que yo estaba en windows lo cual no me dejaba usar esa, en la imagen verán el error, pero encontre otra la cual me ayudo para descomprimir el zip mostrandome ya la imagen, el comando que utilice fue:

→**sudo apt-get install p7zip-full**

→**7z x poneglyph.zip**

Al correr estos 2 comandos me salio lo siguiente:

```
poneglyph.zip
luffy@6962099a084b:~/ONEPIECE/Alabasta/Rainbase/Casa_de_Vivi$ unzip poneglyph.zip
Archive: poneglyph.zip
  skipping: poneglyph.jpeg      need PK compat. v6.3 (can do v4.6)
luffy@6962099a084b:~/ONEPIECE/Alabasta/Rainbase/Casa_de_Vivi$ ls
poneglyph.zip
luffy@6962099a084b:~/ONEPIECE/Alabasta/Rainbase/Casa_de_Vivi$ cd poneglyph.zip
bash: cd: poneglyph.zip: Not a directory
luffy@6962099a084b:~/ONEPIECE/Alabasta/Rainbase/Casa_de_Vivi$ zipinfo poneglyph.zip
Archive: poneglyph.zip
Zip file size: 68509 bytes, number of entries: 1
-rw-rw-rw- 6.3 fat   69969 Bx u099 25-Mar-31 09:29 poneglyph.jpeg
1 file, 69969 bytes uncompressed, 68349 bytes compressed:  2.3%
luffy@6962099a084b:~/ONEPIECE/Alabasta/Rainbase/Casa_de_Vivi$ sudo apt-get install p7zip-full
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  p7zip
Suggested packages:
  p7zip-rar
The following NEW packages will be installed:
  p7zip p7zip-full
0 upgraded, 2 newly installed, 0 to remove and 10 not upgraded.
Need to get 1549 kB of archives.
After this operation, 5847 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 p7zip amd64 16.02+dfsg-8 [363 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 p7zip-full amd64 16.02+dfsg-8 [1186 kB]
Fetched 1549 kB in 2s (878 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package p7zip.
(Reading database ... 20077 files and directories currently installed.)
Preparing to unpack .../p7zip_16.02+dfsg-8_amd64.deb ...
Unpacking p7zip (16.02+dfsg-8) ...
Selecting previously unselected package p7zip-full.
Preparing to unpack .../p7zip-full_16.02+dfsg-8_amd64.deb ...
Unpacking p7zip-full (16.02+dfsg-8) ...
Setting up p7zip (16.02+dfsg-8) ...
Setting up p7zip-full (16.02+dfsg-8) ...
Selecting previously unselected package p7zip.
(Reading database ... 20077 files and directories currently installed.)
Preparing to unpack .../p7zip_16.02+dfsg-8_amd64.deb ...
Unpacking p7zip (16.02+dfsg-8) ...
Selecting previously unselected package p7zip-full.
Preparing to unpack .../p7zip-full_16.02+dfsg-8_amd64.deb ...
Unpacking p7zip-full (16.02+dfsg-8) ...
Setting up p7zip (16.02+dfsg-8) ...
Setting up p7zip-full (16.02+dfsg-8) ...
luffy@6962099a084b:~/ONEPIECE/Alabasta/Rainbase/Casa_de_Vivi$ 7z x poneglyph.zip

7-zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=C,Utf16=off,HugeFiles=on,64 bits,8 CPUs Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz (706E5),ASM,AES-NI)

Scanning the drive for archives:
1 file, 68509 bytes (67 KiB)

Extracting archive: poneglyph.zip
--
Path = poneglyph.zip
Type = zip
Physical Size = 68509

Enter password (will not be echoed):
Everything is Ok

Size:       69969
Compressed: 68509
luffy@6962099a084b:~/ONEPIECE/Alabasta/Rainbase/Casa_de_Vivi$ ls
poneglyph.jpeg  poneglyph.zip
luffy@6962099a084b:~/ONEPIECE/Alabasta/Rainbase/Casa_de_Vivi$
```

Como podemos ver ya se extrajo el zip el cual contiene la imagen que estábamos buscando para este reto.

**Paso 7:** Ahora nos toca obtener el texto de la imagen, primero lo que hice fue guardar la imagen en mi computadora lo cual la descargue desde docker para poder pasarla al programa que nos brindó Ludwing el cual extrae el texto de una imagen el cual hace un XOR con mi número de carné, los archivos que utilice son luffy\_xor.py y extract\_text\_from\_image.py estos archivos son los que nos brindó Ludwing al correr el programa me mostró lo siguiente:

```

6 def extraer_texto_metadata(imagen_path):
7     img = Image.open(imagen_path)
8
9     # Obtener los metadatos EXIF
10    exif_dict = piexif.load(img.info.get('exif', b''))
11
12    # Obtener el texto almacenado en 'Artist' (o en el campo que elegimos)
13    texto = exif_dict['0th'].get(piexif.ImageIFD.Artist)
14
15    if texto:
16        return texto.decode('utf-8')
17    return None
18
19
20 # Uso del código para descifrar la imagen
21 # Ejemplo de uso
22 image_path = "C:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf_onepice_symmetric_cipher/challenges/luffy/"
23 student_id = input("Introduce tu carné para descifrar el mensaje: ")
24 texto_cifrado = extraer_texto_metadata(image_path)
25 decrypted_text = xor_cipher(texto_cifrado, student_id)
26 print(decrypted_text)
27
28

```

PROBLEMS OUTPUT PORTS TERMINAL GITLENS COMMENTS

PS C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf\_onepice\_symmetric\_cipher\utils> & "C:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf\_onepice\_symmetric\_cipher/env/Scripts/python.exe" "c:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf\_onepice\_symmetric\_cipher/utils/extract\_text\_from\_image.py"

Introduce tu carné para descifrar el mensaje: 21285

b'By the time the Straw Hat Pirates arrived at Fish-Man Island, its Road Poneglyph had been removed.'

PS C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf\_onepice\_symmetric\_cipher\utils> []

Live Share You, 40 seconds ago Not Committed Yet Ln 22, Col 174 Spaces: 4 UTF-8 CRLF Python 3.11.0 (env) Background 11:35 31/03/2025

Y listo logramos terminar el primer reto del XOR ya que encontramos la flag y el texto de la imagen en base a mi número de carné 21285.

## Reto # 2 Zoro

### ✓ Introducción del reto

En este reto se trabajará con el cifrado RC4, un algoritmo de flujo que fue ampliamente utilizado en protocolos de seguridad como SSL y WEP. Aunque fue popular por su velocidad y simplicidad, RC4 presenta vulnerabilidades bien conocidas que lo hacen inseguro para aplicaciones modernas. Aprovechando estas debilidades, se deberá analizar un mensaje cifrado utilizando RC4, comprender cómo se genera el flujo de claves y utilizar esa información para intentar descifrar el contenido original del mensaje.

### ✓ Objetivos del reto

- Comprender cómo funciona el algoritmo de cifrado de flujo RC4 y cómo utiliza una clave para generar un flujo pseudoaleatorio.
- Analizar un flujo de datos cifrado con RC4.
- Analizar las debilidades de RC4, como los sesgos en la generación de claves y la vulnerabilidad a ataques por correlación.

### ✓ Flag Encontrada

→ **FLAG\_023d3aaaf90363ca35ee54bf8f894dc81**

## ✓ Párrafo del poneglyphs

Imagen encontrada



Párrafo encontrado

→ b'After arriving at the Sea Forest, Robin read the lone Poneglyph there and'

## ✓ Documentación del proceso, Presentación y claridad

**Paso 1:** ingresamos al segundo challenge, ya entrando al segundo challenge debemos hacer lo mismo que el reto anterior que es ingresar el usuario y la password que recordemos que la password ahora es la flag que encontramos en el reto anterior para ello corrí los siguientes comandos:

→ docker exec -it zoro\_challenge bash

→ su zore con el password del reto anterior FLAG\_1912245f868935872c43e1ef726bc38d

Al correrlo aparece lo siguiente:

```
(env) C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepiece_symmetric_cipher>docker exec -it zoro_challenge bash
nobody@e17a74ee3ae4:/home/zoro$ su zoro
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

zoro@e17a74ee3ae4:~$ -
```

**Paso 2:** Ahora era momento de encontrar la flag, para ello hice y utilice el mismo comando en el reto anterior que me busca y me muestre todos los archivos txt o los que tengan flag, el comando es el siguiente:

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

```

zoro@e17a74ee3ae4:~/ONEPIECE$ find / -type f \(-name \"*.txt\" -o -name \"*.flag\" -o -name \"*.hidden\" -o -name \"*.enc\" \) 2>/dev/null
/home/zoro/ONEPIECE/Dressrosa/Royal_Palace/Casa_de_Rebecca/flag.txt
/home/zoro/ONEPIECE/Dressrosa/Flower_Hill/Casa_de_Rebecca/flag.txt
/home/zoro/ONEPIECE/Sabaoody_Archipelago/Grove_88/Casa_de_Shaky/flag.txt
/home/zoro/ONEPIECE/Sabaoody_Archipelago/Grove_109/Casa_de_Keimi/flag.txt
/home/zoro/ONEPIECE/Sabaoody_Archipelago/Grove_109/Casa_de_Rayleigh/flag.txt
/home/zoro/ONEPIECE/Sabaoody_Archipelago/Grove_24/Casa_de_Shaky/flag.txt
/home/zoro/ONEPIECE/Skypiea/Upper_Yard/Casa_de_Alsa/flag.txt
/home/zoro/ONEPIECE/Zou/Left_Hin_Leg/Casa_de_Nekomamushi/flag.txt
/home/zoro/ONEPIECE/Alabasta/Spiders_Cafe/Casa_de_Miss_All_Sunday/flag.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/unicore/Blocks.txt
/usr/share/perl/5.34.0/Unicode/Collate/allkeys.txt
/usr/share/perl/5.34.0/Unicode/Collate/keys.txt
/usr/share/gnupg/help.nb.txt
/usr/share/gnupg/help.sk.txt
/usr/share/gnupg/help.de.txt
/usr/share/gnupg/help.it.txt
/usr/share/gnupg/help.pl.txt
/usr/share/gnupg/help.ca.txt
/usr/share/gnupg/help.cs.txt
/usr/share/gnupg/help.be.txt
/usr/share/gnupg/help.ru.txt
/usr/share/gnupg/help.zh_TW.txt
/usr/share/gnupg/help.gl.txt
/usr/share/gnupg/help.ja.txt
/usr/share/gnupg/help_zh_CN.txt
/usr/share/gnupg/help_pt.txt
/usr/share/gnupg/help_id.txt
/usr/share/gnupg/help_eo.txt
/usr/share/gnupg/help_et.txt
/usr/share/gnupg/help_es.txt
/usr/share/gnupg/help_sv.txt
/usr/share/gnupg/help_ro.txt
/usr/share/gnupg/help_da.txt
/usr/share/gnupg/help_fr.txt
/usr/share/gnupg/help_el.txt
/usr/share/gnupg/help_fi.txt
/usr/share/gnupg/help_tr.txt
/usr/share/gnupg/help_hu.txt

```

**Paso 3:** Hice lo mismo que el challenge anterior que es leer todos los archivos flag.txt y mostrarlos para encontrar el número raro que ando buscando para esto el comando que corres el siguiente:

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

Al correr el comando me aparece lo siguiente:

```

zoro@e17a74ee3ae4:~/ONEPIECE$ find /home/zoro/ONEPIECE/ -type f -name "flag.txt" -exec cat {} + 2>/dev/null
Has encontrado un lugar abandonadoHas encontrado un lugar peligrosoHas encontrado un enemigo y ha tenido que huir6d4080499d48a3ab0ac7e965ca7eb90510b84006d09a0c5f75b491c
5c5245a4a2b06b36337dHas encontrado un objeto misterioso y no sabes qud esHas encontrado un objeto misterioso y no sabes qud esHas encontrado un lugar abandonadoHas encontrado un mapa que apunta a zoroHas encontrado un mapa que apunta a zoroHas encontrado un enemigo y ha tenido que lucharzoro@e1/a/4ee3ae4:~/ONEPIECE$ 

```

Vemos que ahora está el mismo número raro ese número raro, debó de aplicarle algo para que me muestre mi flag, aquí es donde está la diferencia del challenge anterior.

→ 6d4080499d48a3ab0ac7e965ca7eb90510b84006d09a0c5f75b491c5c5245a42b06b36337

**Paso 4:** Ahora ese número extraño debo crear un script el cual me ayude a desencriptar utilizando como llave mi número de carné, para esto cree un script en python. Este programa toma un texto cifrado en formato hexadecimal en mi caso es ese número extraño que encontré y lo descifra utilizando el algoritmo RC4 con una clave específica, en este caso, es mi número de carné 21285. Convierte la clave y el texto cifrado a bytes, aplica el descifrado y muestra el mensaje en texto legible si es posible. Aquí les muestro como quedó el resultado

The screenshot shows a terminal window with the following content:

```
RC4.py > ...
from Crypto.Cipher import ARC4
import binascii
# Datos
hex_cifrado = "6d4080499d48a3ab0ac7e965ca7eb90510b84006d09a0c5f75b491c5c5245a42b06b36337d"
clave = "21285" # Número de carné
# Convertir la clave a bytes
clave_bytes = clave.encode()
# Convertir el hexadecimal a bytes
cifrado_bytes = binascii.unhexlify(hex_cifrado)
# Crear el descifrador RC4
rc4 = ARC4.new(clave_bytes)
# Descifrar
mensaje_descifrado = rc4.decrypt(cifrado_bytes)
# Mostrar el resultado
def es_texto_legible(texto):
    try:
        texto.decode('utf-8')
    return True
except UnicodeDecodeError:
PROBLEMS OUTPUT PORTS TERMINAL GITLENS COMMENTS
PS C:\Users\Personal\Documents\Universidad\9 Semestre\DiseñoEI> & "C:/Program Files/Python311/python.exe" "c:/Users/Personal/Documents/Universidad/9 Semestre\RC4.py"
Mensaje descifrado: FLAG_023d3aaaf90363ca35ee54bf8f894dc81
PS C:\Users\Personal\Documents\Universidad\9 Semestre\DiseñoEI>
```

## **Implementación:**

Este programa descifra un mensaje cifrado con RC4 utilizando la biblioteca Crypto. Primero, convierte la clave (en este caso, un número de carné) y el mensaje cifrado (en formato hexadecimal) a bytes. Luego, crea un objeto RC4 con la clave y lo usa para descifrar el contenido. Finalmente, verifica si el resultado puede ser interpretado como texto legible en UTF-8. Si es así, lo muestra como texto; de lo contrario, imprime la salida en formato de bytes.

→Mensaje descifrado: FLAG 023d3aaf90363ca35ee54bf8f894dc81

**Paso 5:** Ahora el próximo objetivo es encontrar la imagen, pero ya sabiendo el comando de como buscar archivos en las carpetas como lo hicimos anteriormente en el reto pasado ya es mucho más fácil ya que ahora busque por extensión png, jpg, zip, etc para poder ver dónde es que estaba la imagen en este caso estamos buscando el zip poneglyphs , para ello utilice el siguiente comando:

```
→find / -type f \(-name "*.jpg" -o -name "*.png" -o -name "*.jpeg" -o -name "*.gif" -o  
-name "*.bmp" -o -name "*.zip"\) 2>/dev/null
```

Al correr el comando me aparece lo siguiente:

```
trado un mapa que apunta a zoroHas encontrado un mapa que apunta a zoroHas encontrado un enemigo y ha tenido que lucharzoro@e17a74ee3ae4:~/ONEPIECE$ find / -type f \(-name *.jpg" -o -name "*.png" -o -name "*.jpeg" -o -name "*.gif" -o -name "*.bmp" -o -name "*.zip" \) 2>/dev/null
/home/zoro/ONEPIECE/Alabasta/Nanohana/Casa_de_Toto/poneglyph.zip
/usr/share/info/gnupg-module-overview.png
/usr/share/info/gnupg-card-architecture.png
/usr/share/pixmaps/debian-logo.png
/usr/share/icons/hicolor/48x48/apps/gvim.png
/usr/share/icons/locoolar/32x32/apps/gvim.png
/usr/share/icons/locoolar/16x16/apps/gvim.png
/usr/share/gitweb/static/git-logo.png
/usr/share/gitweb/static/git-favicon.png
zoro@e17a74ee3ae4:~/ONEPIECE$
```

Ya encontramos donde está el zip es hora de extraerlo.

**Paso 6:** Ahora tenemos que extraer el zip para ello lo haremos como el reto anterior sin el unzip ya que este me da problemas entonces naveguemos hasta donde se encuentra el zip y ya cuando estemos ahí lo extraemos y mostramos que archivos contiene para ello corramos los siguientes comandos.

```
→ sudo apt update  
→ sudo apt-get install p7zip-full  
→ 7z x poneglyph.zip
```

A continuación les muestro unas imagenes:

```
zoro@e17a74ee3ae4:~/ONEPIECE$ cd Alabasta/  
zoro@e17a74ee3ae4:~/ONEPIECE/Alabasta$ cd Nanohana/  
zoro@e17a74ee3ae4:~/ONEPIECE/Alabasta/Nanohana$ cd Casa_de_Toto/  
zoro@e17a74ee3ae4:~/ONEPIECE/Alabasta/Nanohana/Casa_de_Toto$ ls  
poneglyph.zip
```

```
zoro@e17a74ee3ae4:~/ONEPIECE/Alabasta/Nanohana/Casa_de_Toto$ sudo apt update  
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]  
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]  
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]  
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]  
Get:5 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]  
Get:6 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]  
Get:7 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [47.7 kB]  
Get:8 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]  
Get:9 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]  
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1241 kB]  
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2773 kB]  
Get:12 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [3972 kB]  
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [4140 kB]  
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1540 kB]  
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [3087 kB]  
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [55.7 kB]  
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [35.2 kB]  
Get:18 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [82.7 kB]  
Fetched 37.3 MB in 13s (2943 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
24 packages can be upgraded. Run 'apt list --upgradable' to see them.  
zoro@e17a74ee3ae4:~/ONEPIECE/Alabasta/Nanohana/Casa_de_Toto$ sudo apt install p7zip-full  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  p7zip  
Suggested packages:  
  p7zip-rar  
The following NEW packages will be installed:  
  p7zip p7zip-full  
0 upgraded, 2 newly installed, 0 to remove and 24 not upgraded.  
Need to get 1549 kB of archives.  
After this operation, 5847 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 p7zip amd64 16.02+dfsg-8 [363 kB]  
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 p7zip-full amd64 16.02+dfsg-8 [1186 kB]  
Fetched 1549 kB in 1s (1832 kB/s)  
debconf: delaying package configuration, since apt-utils is not installed  
Selecting previously unselected package p7zip.  
(Reading database ... 20059 files and directories currently installed.)
```

```

zoro@e17a74ee3ae4:~/ONEPIECE/Alabasta/Nanohana/Casa_de_Toto$ 7z x poneglyph.zip
7-zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=C,Utf16=off,HugeFiles=on,64 bits,8 CPUs Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz (706E5),ASM,AES-NI)

Scanning the drive for archives:
1 file, 53247 bytes (52 KiB)

Extracting archive: poneglyph.zip
--
Path = poneglyph.zip
Type = zip
Physical Size = 53247

Enter password (will not be echoed):
Everything is Ok

Size:      52869
Compressed: 53247
zoro@e17a74ee3ae4:~/ONEPIECE/Alabasta/Nanohana/Casa_de_Toto$ ls
poneglyph.jpeg  poneglyph.zip
zoro@e17a74ee3ae4:~/ONEPIECE/Alabasta/Nanohana/Casa_de_Toto$ 

```

Ya trajimos la imagen, ahora debemos de copiar a nuestra computadora en el reto anterior la descargue desde docker pero este no me deja entonces me puse a investigar y encontré un comando que me extrae la imagen directamente a mi compu, para ello correr el siguiente comando:

→**docker cp**

**zoro\_challenge:/home/zoro/ONEPIECE/Alabasta/Nanohana/Casa\_de\_Toto/poneglyph.jpeg ./poneglyph\_zoro.jpg.**

A continuación una imagen de cómo se vería:

```

C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>docker cp zoro_challenge:/home/zoro/ONEPIECE/Alabasta/Nanohana/Casa_de_Toto/poneglyph.jpeg ./poneglyph_zoro.jpg
Successfully copied 54.8KB to C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher\poneglyph_zoro.jpg
C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>

```

Listo ya tenemos la imagen en la computadora



**Paso 7:** Ahora nos toca obtener el texto de la imagen, primero lo que hice fue guardar la imagen en mi computadora lo cual utilice el comando del paso 6 para poder pasarlal al programa que nos brindó Ludwing el cual extrae el texto de una imagen el cual hace un XOR con mi número de carné, los archivos que utilice son luffy\_xor.py y extract\_text\_from\_image.py estos archivos son los que nos brindó Ludwing al correr el programa me mostró lo siguiente:

```

extract_text_from_image.py M x zoro_rc4.py
utils > extract_text_from_image.py ...
4   from patty_xor import xor_cipher
5
6 def extraer_texto_metadata(imagen_path):
7     # Abrir la imagen
8     img = Image.open(imagen_path)
9
10    # Obtener los metadatos EXIF
11    exif_dict = piexif.load(img.info.get("exif", b''))
12
13    # Obtener el texto almacenado en 'Artist' (o en el campo que elegimos)
14    texto = exif_dict['0th'].get(piexif.ImageIFD.Artist)
15    if texto:
16        return texto.decode('utf-8')
17    return None
18
19
20    # Uso del código para descifrar la imagen
21    # Ejemplo de uso
22    image_path = "C:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf_onepice_symmetric_cipher/poneglyph_zoro.jpg"
23    student_id = input("Introduce tu carné para descifrar el mensaje: ")
24    texto_cifrado = extraer_texto_metadata(image_path)
25    decrypted_text = xor_cipher(texto_cifrado, student_id)
26    print(decrypted_text)
27
28

```

PROBLEMS OUTPUT PORTS TERMINAL GITLENS COMMENTS

PS C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf\_onepice\_symmetric\_cipher> & "C:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf\_onepice\_symmetric\_cipher/env/Scripts/python.exe" "c:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf\_onepice\_symmetric\_cipher/utils/extract\_text\_from\_image.py"
Introduce tu carné para descifrar el mensaje: 21285
b'After arriving at the Sea Forest, Robin read the lone Poneglyph there and'
PS C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf\_onepice\_symmetric\_cipher>

0 0 Live Share Reactivating terminals... Not Committed Yet Ln 22, Col 120 Spaces: 4 UTF-8 CRLF Python 3.11.0 (env) Background 13:50 ESP 3/04/2025

**Texto de la imagen encontrada: b'After arriving at the Sea Forest, Robin read the lone Poneglyph there and'**

## Reto # 3 Usopp

### ✓ Introducción del reto

En este reto nos enfrentaremos a un mensaje oculto dentro de archivos de texto, cifrado mediante una técnica de flujo que utiliza XOR con claves derivadas de un número de carné. Este tipo de cifrado genera una secuencia pseudoaleatoria (keystream) que se combina con el mensaje original para ocultar su contenido. Sin embargo, cuando el generador de claves es débil o predecible, es posible descubrir vulnerabilidades que permiten recuperar el mensaje original. nuestro desafío consistirá en analizar cómo se genera la clave, identificar fallos en la implementación y aplicar técnicas de descifrado para revelar el mensaje oculto.

### ✓ Objetivos del reto

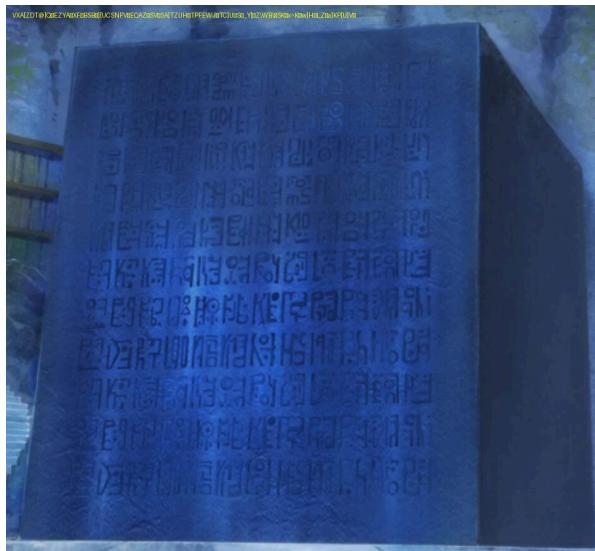
- Comprender el funcionamiento de los cifrados de flujo personalizados y su dependencia de claves pseudoaleatorias.
- Analizar la implementación de un cifrado de flujo personalizado y encontrar fallos en la generación de claves para descifrar el mensaje.
- Identificar vulnerabilidades comunes, como la reutilización de claves o el uso de generadores predecibles.
- Fortalecer habilidades de análisis criptográfico en entornos reales con implementaciones no estándar.

## ✓ Flag Encontrada

→ **FLAG\_5c0cf71f5e6f6f90b682d3f31238b906**

## ✓ Párrafo del poneglyphs

Imagen encontrada



Párrafo encontrado

→ b'discovered that it was engraved with an apology letter from a man known as Joy Boy to Poseidon.'

## ✓ Documentación del proceso, Presentación y claridad

**Paso 1:** ingresamos al tercer challenge, ya entrando al tercer challenge debemos hacer lo mismo que los retos anteriores que es ingresar el usuario y la password que recordemos que la password ahora es la flag que encontramos en el reto 2 anterior para ello corrí los siguientes comandos:

→ docker exec -it usopp\_challenge bash

→ su usopp con el password del reto 2 FLAG\_023d3aa90363ca35ee54bf8f894dc81

Al correrlo aparece lo siguiente:

```
C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>docker exec -it usopp_challenge bash
nobody@c51f3e9bedba:/home/usopp$ su usopp
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

usopp@c51f3e9bedba:~$
```

**Paso 2:** Ahora era momento de encontrar la flag, para ello hice y utilice el mismo comando en los retos anteriores que me busca y me muestre todos los archivos txt o los que tengan flag, el comando es el siguiente:

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

Al correrlo mostrará todos los archivos encontrados como a continuación se muestra:

```
usopp@c51f3e9bedba:~$ find / -type f \(\ -name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null
/home/usopp/ONEPIECE/Dressrosa/Toy_House/Casa_de_Sugar/flag.txt
/home/usopp/ONEPIECE/Dressrosa/Corrida_Colosseum/Casa_de_Viola/flag.txt
/home/usopp/ONEPIECE/Sabaody_Archipelago/Grove_80/Casa_de_Rayleigh/flag.txt
/home/usopp/ONEPIECE/Skypiea/Angel_Beach/Casa_de_Pagaya/flag.txt
/home/usopp/ONEPIECE/Skypiea/Angel_Beach/Casa_de_Gan_Fall/flag.txt
/home/usopp/ONEPIECE/Skypiea/Shandora/Casa_de_Robin/flag.txt
/home/usopp/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Inuarashi/flag.txt
/home/usopp/ONEPIECE/Zou/Right_Belly_Fortress/Casa_de_Nekomamushi/flag.txt
/home/usopp/ONEPIECE/Alabasta/Nanohana/Casa_de_Toto/flag.txt
/home/usopp/ONEPIECE/Alabasta/Alubarna/Casa_de_Vivi/flag.txt
/home/usopp/ONEPIECE/Alabasta/Katorea/Casa_de_Kohza/flag.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/unicore/Blocks.txt
/usr/share/perl/5.34.0/unicore/Collate/allkeys.txt
/usr/share/perl/5.34.0/Unicode/Collate/keys.txt
/usr/share/gnupg/help.nb.txt
/usr/share/gnupg/help.sk.txt
```

**Paso 3:** Hice lo mismo que el challenge 2 que es leer todos los archivos flag.txt y mostrarlos para encontrar el número raro que ando buscando para esto el comando que corres es el siguiente:

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

Al correr el comando me aparece lo siguiente:

```
usopp@c51f3e9bedba:~$ find /home/usopp/ONEPIECE/ -type f -name "flag.txt" -exec cat {} + 2>/dev/null
Has encontrado un mapa que apunta a usoppHas encontrado un amigo y han decidido viajar juntasHas encontrado un lugar lleno de secretos y misterios que apuntan a usoppHa
s encontrado una pista que apunta a usoppHas encontrado un mapa que apunta a usoppHas encontrado un enemigo y ha tenido que huirHas encontrado un obstaculo y hay que su
perarloHas encontrado un lugar misteriosoHas encontrado un mapa que apunta a usoppHas encontrado una pista que apunta a usoppa77742694e1f51851a6e3839d7c2887a486b5c3ec91
d17d6114924ec13abc486b01d5fd83fusopp@c51f3e9bedba:~$
```

Nos volvemos a encontrar con este número grande raro el cual es el que nos va servir para poder encontrar la flag aplicando lo que el reto nos pida.

```
→ a77742694e1f51851a6e3839d7c2887a486b5c3ec91d17d6114924ec13abc486b01d5fd83f
```

**Paso 4:** Al igual que los retos anteriores este número raro se debe de desencriptar para poder obtener la flag para el próximo reto, para ello y viendo podemos aprovechar la función usopp\_cipher.py que nos brindo Ludwing en el proyecto, vamos a utilizar en específico la función decrypt para poder encontrar nuestra flag. A continuación una imagen de cual es la flag encontrada y el script utilizado.

```

utils > 🐍 usopp.py > ...
1  from usopp_cipher import decrypt
2
3  # La cadena cifrada que deseas descifrar (en formato hexadecimal)
4  ciphertext = "a77742694e1f51851a6e389d7c2887a486b5c3ec91d17d6114924ec13abc486b01d5fd83f"
5
6  # Llamar a la función decrypt pasándole la cadena cifrada
7  decrypt(ciphertext)
8

```

PROBLEMS OUTPUT PORTS TERMINAL GITLENS COMMENTS

```

PS C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher> & "C:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf_onepice_symmetric_cipher/env/Scripts/python.exe" "c:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf_onepice_symmetric_cipher/utils/usopp.py"
Semilla encontrada: 1234
Texto descifrado: FLAG_5c0cf71f5e6f6f90b682d3f31238b906
PS C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher> []

```

→ **FLAG\_5c0cf71f5e6f6f90b682d3f31238b906**

### Implementación:

El código implementa un cifrado basado en un "generador de secuencias de claves" (keystream) que utiliza un generador de números aleatorios (PRNG) con una semilla dada. En el cifrado, se realiza una operación XOR entre cada byte del texto plano y el keystream generado. En la descifrado, la función intenta encontrar la semilla correcta para descifrar el texto cifrado, verificando si comienza con "FLAG\_". La principal diferencia entre un XOR "normal" y este cifrado es que aquí se usa una secuencia de claves generada de forma pseudo-aleatoria para hacer la operación XOR, lo que lo hace dependiente de la semilla para generar el keystream y producir el texto cifrado.

**Paso 5:** Ahora el próximo objetivo es encontrar la imagen, pero ya sabiendo el comando de como buscar archivos en las carpetas como lo hicimos en los retos pasados ya es mucho más fácil ya que ahora busque por extensión png, jpg, zip, etc para poder ver dónde es que estaba la imagen en este caso estamos buscando el zip poneglyphs , para ello utilice el siguiente comando:

```
→ find / -type f \(-name "*.jpg" -o -name "*.png" -o -name "*.jpeg" -o -name "*.gif" -o -name "*.bmp" -o -name "*.zip"\) 2>/dev/null
```

Al correr el comando me aparece lo siguiente:

```

usopp@c51f3e9bedba:~/ONEPIECE$ find / -type f \(-name "*.jpg" -o -name "*.png" -o -name "*.jpeg" -o -name "*.gif" -o -name "*.bmp" -o -name "*.zip"\) 2>/dev/null
/home/usopp/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Kawamatsu/poneglyph.zip
/usr/share/info/gnupg-module-overview.png
/usr/share/info/gnupg-card-architecture.png
/usr/share/pixmaps/debian-logo.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
usopp@c51f3e9bedba:~/ONEPIECE$
```

Y de nuevo nos volvemos a topar al .zip que es el que debemos de extraer para la imagen.

**Paso 6:** Ahora tenemos que extraer el zip para ello lo haremos como los retos anteriores con p7zip-full, entonces naveguemos hasta donde se encuentra el zip y ya cuando estemos ahí lo extraemos y mostramos que archivos contiene para ello corramos los siguientes comandos.

→ **sudo apt update**

→ **sudo apt-get install p7zip-full**

→ **7z x poneglyph.zip**

A continuación les muestro unas imágenes:

```
usopp@c51f3e9bedba:~/ONEPIECE$ cd Zou/
usopp@c51f3e9bedba:~/ONEPIECE/Zou$ cd Right_Hind_Leg/
usopp@c51f3e9bedba:~/ONEPIECE/Zou/Right_Hind_Leg$ cd Casa_de_Kawamatsu/
usopp@c51f3e9bedba:~/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Kawamatsu$ ls
poneglyph.zip

p7zip
Suggested packages:
  p7zip-rar
The following NEW packages will be installed:
  p7zip p7zip-full
0 upgraded, 2 newly installed, 0 to remove and 24 not upgraded.
Need to get 1549 kB of archives.
After this operation, 5847 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 p7zip amd64 16.02+dfsg-8 [363 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 p7zip-full amd64 16.02+dfsg-8 [1186 kB]
Fetched 1549 kB in 0s (1691 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package p7zip.
(Reading database ... 28059 files and directories currently installed.)
Preparing to unpack .../p7zip_16.02+dfsg-8_amd64.deb ...
Unpacking p7zip (16.02+dfsg-8) ...
Selecting previously unselected package p7zip-full.
Preparing to unpack .../p7zip-full_16.02+dfsg-8_amd64.deb ...
Unpacking p7zip-full (16.02+dfsg-8) ...
Setting up p7zip (16.02+dfsg-8) ...
Setting up p7zip-full (16.02+dfsg-8) ...
usopp@c51f3e9bedba:~/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Kawamatsu$ 7z x poneglyph.zip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=C,Utf16=off,HugeFiles=on,64 bits,8 CPUs Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz (70E5),ASM,AES-NI)

Scanning the drive for archives:
1 file, 64425 bytes (63 kB)

Extracting archive: poneglyph.zip
--
Path = poneglyph.zip
Type = zip
Physical Size = 64425

Enter password (will not be echoed):
Everything is OK

Size:       64581
Compressed: 64425
usopp@c51f3e9bedba:~/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Kawamatsu$ ls
poneglyph.jpeg  poneglyph.zip
```

Ya trajimos la imagen ahora es momento de cargarla a nuestra computadora local, para ello correr el siguiente comando:

→ **docker cp**

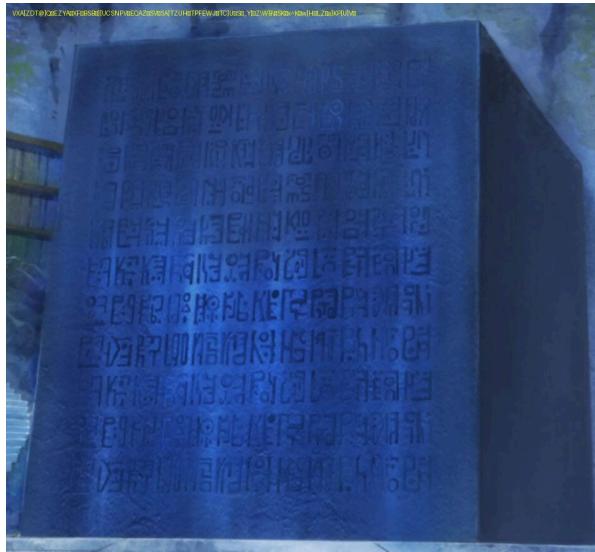
**usopp\_challenge:/home/usopp/ONEPIECE/Zou/Right\_Hind\_Leg/Casa\_de\_Kawamatsu/**  
**poneglyph.jpeg ./poneglyph\_usopp.jpg.**

se veria algo así:

```
C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>docker cp usopp_challenge:/home/usopp/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Kawamatsu/poneglyph.jpeg ./poneglyph_usopp.jpg.
Successfully copied 66.6kB to C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher\poneglyph_usopp.jpg

C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>
```

Ya esta la imagen en mi computadora



**Paso 7:** Ahora nos toca obtener el texto de la imagen, primero lo que hice fue guardar la imagen en mi computadora lo cual utilice el comando del paso 6 para poder pasarlal al programa que nos brindó Ludwing el cual extrae el texto de una imagen el cual hace un XOR con mi número de carné, los archivos que utilice son luffy\_xor.py y extract\_text\_from\_image.py estos archivos son los que nos brindó Ludwing al correr el programa me mostró lo siguiente:

```
utils > extract_text_from_image.py > ...
 6 def extraer_texto_metadata(image_path):
10     # Obtener los metadatos EXIF
11     exif_dict = piexif.load(img.info.get('exif', b''))
12
13     # Obtener el texto almacenado en 'Artist' (o en el campo que elegimos)
14     texto = exif_dict['0th'].get(piexif.ImageIFD.Artist)
15     if texto:
16         return texto.decode('utf-8')
17     return None
18
19
20 # Uso del código para descifrar la imagen
21 # Ejemplo de uso
22 image_path = "C:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf_onepice_symmetric_cipher/poneglyph_usopp.j
23 student_id = input("Introduce tu carné para descifrar el mensaje: ")
24 texto_cifrado = extraer_texto_metadata(image_path)
25 decrypted_text = xor_cipher(texto_cifrado, student_id)
26 print(decrypted_text)
27
```

PROBLEMS OUTPUT PORTS TERMINAL GITLENS COMMENTS

```
PS C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher> & "C:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf_onepice_symmetric_cipher/env/Scripts/python.exe" "c:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf_onepice_symmetric_cipher/utils/extract_text_from_image.py"
Semilla encontrada: 1234
Texto descifrado: FLAG_5c0cf71f5e6f6f90b682d3f31238b906
PS C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher> & "C:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf_onepice_symmetric_cipher/env/Scripts/python.exe" "c:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf_onepice_symmetric_cipher/utils/extract_text_from_image.py"
Introduce tu carné para descifrar el mensaje: 21285
b'discovered that it was engraved with an apology letter from a man known as Joy Boy to Poseidon.'
PS C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>
```

**Texto encontrado:** b'discovered that it was engraved with an apology letter from a man known as Joy Boy to Poseidon.'

## Reto # 4 Nami

### ✓ Introducción del reto

En este reto explorarán el cifrado ChaCha20, un algoritmo de flujo moderno ampliamente utilizado por su velocidad y alta seguridad. Este tipo de cifrado se basa en una clave secreta y un valor llamado nonce, que debe ser único para cada mensaje cifrado. A través de este ejercicio, analizaremos cómo el uso (o mal uso) del nonce y la clave influye directamente en la seguridad del mensaje. Al experimentar con diferentes combinaciones, comprenderás por qué incluso un cifrado robusto como ChaCha20 puede volverse vulnerable si se configura incorrectamente.

### ✓ Objetivos del reto

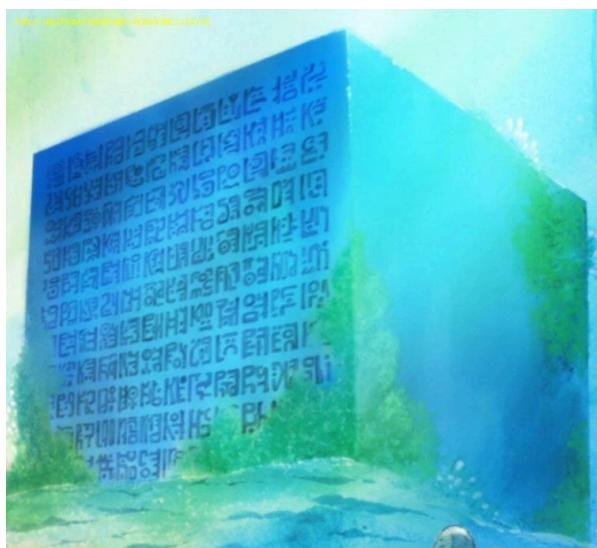
- Comprender el funcionamiento del cifrado de flujo ChaCha20 y su importancia en la criptografía moderna.
- Analizar el rol del nonce y su impacto en la seguridad del mensaje cifrado.
- Experimentar con la encriptación y desencriptación de datos utilizando claves derivadas del usuario.
- Fortalecer habilidades prácticas en el uso de cifrados seguros y buenas prácticas criptográficas.

### ✓ Flag Encontrada

→ **FLAG\_a6c5ee7df3b480e8b8754e112c5d5d78**

### ✓ Párrafo del poneglyphs

Imagen encontrada



Párrafo encontrado

→ **b'he told Robin the story handed down through '**

## ✓ Documentación del proceso, Presentación y claridad

**Paso 1:** ingresamos al tercer challenge, ya entrando al tercer challenge debemos hacer lo mismo que los retos anteriores que es ingresar el usuario y la password que recordemos que la password ahora es la flag que encontramos en el reto 3 para ello corrí los siguientes comandos:

→ docker exec -it nami\_challenge bash

→ su nami con el password del reto 3 FLAG\_5c0cf71f5e6f6f90b682d3f31238b906

Al correrlo aparece lo siguiente:

```
C:\Users\Personal\Documents\Universidad\9 Semestre\CTF\OnePiece_Symmetric_Cipher>docker exec -it nami_challenge bash
nobody@02f7fc48e8e:/home/nami$ su nami
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

nami@02f7fc48e8e:~$
```

**Paso 2:** Es hora de encontrar la flag, para ello hice y utilice el mismo comando en los retos anteriores que me busca y me muestre todos los archivos txt o los que tengan flag, el comando es el siguiente:

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

Al correrlo mostrará todos los archivos encontrados como a continuación se muestra:

```
nami@02f7fc48e8e:~$ find / -type f \(-name "*.txt" -o -name "*.flag" -o -name "*.hidden" -o -name "*.enc" \) 2>/dev/null
/home/nami/ONEPIECE/Dressrosa/Flower_Hill/Casa_de_Riku_Dold_III/flag.txt
/home/nami/ONEPIECE/Sabadoy_Archipelago/Grove_80/Casa_de_Keimi/Flag.txt
/home/nami/ONEPIECE/Sabadoy_Archipelago/Grove_80/Casa_de_Rayleigh/flag.txt
/home/nami/ONEPIECE/Sabadoy_Archipelago/Grove_66/Casa_de_Rayleigh/flag.txt
/home/nami/ONEPIECE/Skypiea/Shandora/Casa_de_Montblanc_Norland/flag.txt
/home/nami/ONEPIECE/Skypiea/Heavens_Gate/Casa_de_Enel/flag.txt
/home/nami/ONEPIECE/Zou/Right_Hind_Leg/Casa_de_Nekomamushi/Flag.txt
/home/nami/ONEPIECE/Zou/Right_Belly_Fortress/Casa_de_Nekomamushi/Flag.txt
/home/nami/ONEPIECE/Alabasta/Rainbase/Casa_de_Vivi/flag.txt
/home/nami/ONEPIECE/Alabasta/Yuba/Casa_de_Toto/Flag.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/Blocks.txt
/usr/share/perl/5.34.0/unicode/collate/allkeys.txt
/usr/share/perl/5.34.0/unicode/collate/keys.txt
/usr/share/gnupg/help_nb.txt
/usr/share/gnupg/help_sk.txt
/usr/share/gnupg/help_de.txt
/usr/share/gnupg/help_it.txt
/usr/share/gnupg/help_pl.txt
/usr/share/gnupg/help_ca.txt
/usr/share/gnupg/help_cs.txt
/usr/share/gnupg/help_be.txt
/usr/share/gnupg/help_ru.txt
/usr/share/gnupg/help_zh_TW.txt
/usr/share/gnupg/help_gl.txt
/usr/share/gnupg/help_ja.txt
/usr/share/gnupg/help_zh_CN.txt
/usr/share/gnupg/help_pt_PT.txt
/usr/share/gnupg/help_id.txt
/usr/share/gnupg/help_eo.txt
/usr/share/gnupg/help_et.txt
/usr/share/gnupg/help_es.txt
/usr/share/gnupg/help_sv.txt
/usr/share/gnupg/help_ro.txt
/usr/share/gnupg/help_da.txt
/usr/share/gnupg/help_fr.txt
/usr/share/gnupg/help_el.txt
/usr/share/gnupg/help_fi.txt
/usr/share/gnupg/help.txt
```

**Paso 3:** Hice lo mismo que el challenge 2 que es leer todos los archivos flag.txt y mostrarlos para encontrar el número raro que ando buscando para esto el comando que corri es el siguiente:

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

Al correr el comando me aparece lo siguiente:

```
nami@02f7fc48e8e:~$ find /home/nami/ONEPIECE/ -type f -name "flag.txt" -exec cat {} + 2>/dev/null
Has encontrado un lugar misterioso3fc06ae08a1fadefbd28e6fd903756702673de26a2bd05affcf4ba3c0b630d438336db15436Has encontrado una pista que apunta a namiHas encontrado un lugar lleno de secretos y misterios que apuntan a namiHas encontrado una pista que apunta a namiHas encontrado un mapa que apunta a namiHas encontrado un enemigo y ha tenido que huirHas encontrado un mapa que apunta a namiHas encontrado un objeto misterioso y no sabes qu@ est@Has encontrado un tesoro que apunta a naminami@02f7fc48e8e:~$
```

Como ya es costumbre vemos de nuevo los números raros el cual es el que le vamos a tener que aplicar ChaCha20 para poder encontrar la última flag.

→ 3fc06ae08a1fadefbd28e6fd903756702673de26a2bd05affcf4ba3c0b630d438336db15436

**Paso 4:** Al igual que los retos anteriores este número raro se debe de desencriptar para poder obtener la última flag, para ello y viendo podemos aprovechar la función nami\_chacha.py que nos brindo Ludwing en el proyecto, vamos a utilizar en específico la función chacha20\_decrypt para poder encontrar nuestra flag con nuestro número de carné como user\_id que es 21285. A continuación una imagen de cual es la flag encontrada y el script utilizado.

```

extract_text_from_image.py M Chacha.py U nami_chacha.py
utils > Chacha.py > ...
1  from nami_chacha import chacha20_decrypt
2  from binascii import unhexlify
3
4  # La cadena hexadecimal cifrada
5  hex_ciphertext = "3fc06ae08a1fadefbd28e6fd903756702673de26a2bd05affcf4ba3c0b630d438336db15436"
6
7  try:
8      # Convertir la cadena hexadecimal a bytes
9      ciphertext_bytes = unhexlify(hex_ciphertext)
10
11     # Tu user_id (en este caso usamos "21285" como en la función original)
12     user_id = "21285"
13
14     # Desencriptar
15     plaintext = chacha20_decrypt(ciphertext_bytes, user_id)
16     print("Mensaje desencriptado:", plaintext)
17 except Exception as e:
18     print(f"Error al desencriptar: {str(e)}")

```

PROBLEMS OUTPUT PORTS TERMINAL GITLENS

anal\Documents\Universidad\9 Semestre\Cifrado\ctf\_onepice\_symmetric\_cipher\utils\Chacha.py"

Mensaje desencriptado: FLAG\_a6c5ee7df3b480e8b8754e112c5d5d78

PS C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf\_onepice\_symmetric\_cipher>

→FLAG\_a6c5ee7df3b480e8b8754e112c5d5d78

### Implementación:

Este código utiliza la función chacha20\_decrypt del módulo nami\_chacha para descifrar un mensaje cifrado en hexadecimal con el algoritmo ChaCha20. Primero convierte la cadena cifrada a bytes, luego utiliza un user\_id como base para derivar la clave y posiblemente el nonce. Finalmente, intenta descifrar el mensaje y muestra el resultado. Si ocurre algún error durante el proceso, lo captura e imprime, facilitando la experimentación con distintos valores y entendiendo cómo afectan al descifrado.

**Paso 5:** Ahora el próximo objetivo es encontrar la imagen, pero ya sabiendo el comando de como buscar archivos en las carpetas como lo hicimos en los retos pasados ya es mucho más fácil ya que ahora busque por extensión png, jpg, zip, etc para poder ver dónde es que estaba la imagen en este caso estamos buscando el zip poneglyphs , para ello utilice el siguiente comando:

→find / -type f \(-name "\*.jpg" -o -name "\*.png" -o -name "\*.jpeg" -o -name "\*.gif" -o -name "\*.bmp" -o -name "\*.zip"\) 2>/dev/null

Al correr el comando me aparece lo siguiente:

```
nami@02f7fc48e8e:~/ONEPIECE$ find / -type f \(-name "*.jpg" -o -name "*.png" -o -name "*.jpeg" -o -name "*.gif" -o -name "*.bmp" -o -name "*.zip" \) 2>/dev/null
/home/nami/ONEPIECE/Zou/Left_Hind_Leg/Casa_de_Nekomamushi/poneglyph.zip
/usr/share/info/gnupg-module-overview.png
/usr/share/info/gnupg-card-architecture.png
/usr/share/pixmaps/debian-logo.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
nami@02f7fc48e8e:~/ONEPIECE$
```

Ya es nuestro último poneglyph.zip que nos volvemos a topar, es hora de que lo descomprimimos.

**Paso 6:** Ahora tenemos que extraer el zip para ello lo haremos como los retos anteriores con p7zip-full, entonces naveguemos hasta donde se encuentra el zip y ya cuando estemos ahí lo extraemos y mostramos que archivos contiene para ello corramos los siguientes comandos.

→ **sudo apt update**

→ **sudo apt-get install p7zip-full**

→ **7z x poneglyph.zip**

A continuación les muestro unas imagenes:

```
nami@02f7fc48e8e:~/ONEPIECE$ cd Zou/
nami@02f7fc48e8e:~/ONEPIECE/Zou$ cd Left_Hind_Leg/
nami@02f7fc48e8e:~/ONEPIECE/Zou/Left_Hind_Leg$ cd Casa_de_Nekomamushi/
nami@02f7fc48e8e:~/ONEPIECE/Zou/Left_Hind_Leg/Casa_de_Nekomamushi$ ls
poneglyph.zip
nami@02f7fc48e8e:~/ONEPIECE/Zou/Left_Hind_Leg/Casa_de_Nekomamushi$ sudo apt-get install p7zip-full
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  p7zip
Suggested packages:
  p7zip-rar
The following NEW packages will be installed:
  p7zip p7zip-full
0 upgraded, 2 newly installed, 0 to remove and 24 not upgraded.
Need to get 1549 kB of archives.
After this operation, 5847 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 p7zip amd64 16.02+dfsg-8 [363 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 p7zip-full amd64 16.02+dfsg-8 [1186 kB]
Fetched 1549 kB in 2s (783 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package p7zip.
(Reading database ... 20059 files and directories currently installed.)
Preparing to unpack .../p7zip_16.02+dfsg-8_amd64.deb ...
Unpacking p7zip (16.02+dfsg-8) ...
Selecting previously unselected package p7zip-full.
Preparing to unpack .../p7zip-full_16.02+dfsg-8_amd64.deb ...
Unpacking p7zip-full (16.02+dfsg-8) ...
Setting up p7zip (16.02+dfsg-8) ...
Setting up p7zip-full (16.02+dfsg-8) ...
nami@02f7fc48e8e:~/ONEPIECE/Zou/Left_Hind_Leg/Casa_de_Nekomamushi$ .
```

```

nami@02f7fc48e8e:~/ONEPIECE/Zou/Left_Hind_Leg/Casa_de_Nekomamushi$ 7z x poneglyph.zip
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (Locale=C.UTF-8, HugeFiles=on, 64 bits, 8 CPUs Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz (706E5), ASM, AES-NI)

Scanning the drive for archives:
1 file, 51957 bytes (51 KiB)

Extracting archive: poneglyph.zip
--
Path = poneglyph.zip
Type = zip
Physical Size = 51957

Enter password (will not be echoed):
Everything is Ok

Size: 51994
Compressed: 51957
nami@02f7fc48e8e:~/ONEPIECE/Zou/Left_Hind_Leg/Casa_de_Nekomamushi$ ls
poneglyph.jpeg  poneglyph.zip
nami@02f7fc48e8e:~/ONEPIECE/Zou/Left_Hind_Leg/Casa_de_Nekomamushi$ 

```

Ya logramos extraer nuestra imagen de nuestro último reto ahora es momento de cargarla a nuestra computadora local, para ello correr el siguiente comando:

→ **docker cp**

**nami\_challenge:/home/nami/ONEPIECE/Zou/Left\_Hind\_Leg/Casa\_de\_Nekomamushi/**  
**poneglyph.jpeg ./poneglyph\_nami.jpg.**

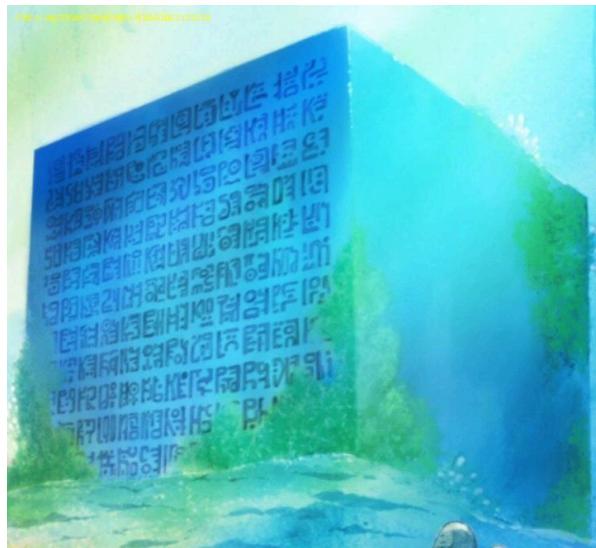
se veria algo así:

```

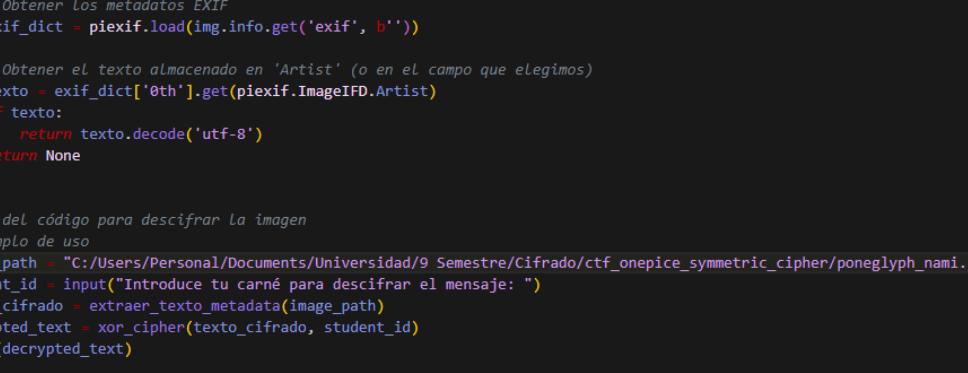
C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>docker cp nami_challenge:/home/nami/ONEPIECE/Zou/Left_Hind_Leg/Casa_de_Nekomamushi/poneglyph.jpeg ./poneglyph_nami.jpg.
Successfully copied 53.8kB to C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher\poneglyph_nami.jpg
C:\Users\Personal\Documents\Universidad\9 Semestre\Cifrado\ctf_onepice_symmetric_cipher>

```

Listo ya tenemos la imagen para nuestro último paso, esta es la imagen obtenida:



**Paso 7:** Ahora nos toca obtener el texto de la imagen, primero lo que hice fue guardar la imagen en mi computadora lo cual utilice el comando del paso 6 para poder pasarlal al programa que nos brindó Ludwing el cual extrae el texto de una imagen el cual hace un XOR con mi número de carné, los archivos que utilice son luffy\_xor.py y extract\_text\_from\_image.py estos archivos son los que nos brindó Ludwing al correr el programa me mostró lo siguiente:



```
re/Cifrado/ctf_onepice_symmetric_cipher/env/Scripts/python.exe" "c:/Users/Personal/Documents/Universidad/9 Semestre/Cifrado/ctf_onepice_symmetric_cipher/utils/extract_text_from_image.py"
Introduce tu carné para descifrar el mensaje: 21285
b'he told Robin the story handed down through '
```

**Texto encontrado: b'he told Robin the story handed down through '**

## **Reflexión final sobre lo aprendido**

Al terminar los cuatro retos aprendí nuevas cosas una de ellas y la principal fue el manejo de docker, ya que actualmente no había tenido acercamiento a docker por lo cual aprendí a cómo levantar un proyecto en contenedores. Además, algunos comandos de búsqueda en la consola de docker, aparte amplié un poco mis conocimientos sobre algunos algoritmos de cifrado y como es que se manejan, los algoritmos son: Xor, Stream cipher custom, RC4 y Chacha20. Estos algoritmos son super interesantes como es que logras encriptar y desencriptar los mensajes cada uno maneja pasos diferentes y cada uno tiene sus fortalezas y debilidades. Este proyecto fue súper interesante y entretenido dado que cada reto tenía su propio nivel de dificultad en cuanto a los algoritmos mencionados anteriormente. Para futuros estudiantes esta genial este proyecto ya que al igual que yo podría aprender sobre estos algoritmos que son súper interesantes e importantes para un mensaje seguro, además no dejemos a un lado el uso de docker ya que actualmente es muy utilizado y este proyecto puede dar conocimientos en docker para poder levantar un proyecto. Como recomendaciones estaría genial que este proyecto se enfocará más en una caricatura como por ejemplo bob esponja ya que esta tiene una historia y en base a esto podrían salir nuevos retos con nuevos algoritmos de encriptación como el sha256, etc.

## **Retos encontrados:**

- Dificultad en poder obtener una imagen de docker a mi computadora local.
  - Dificultad al principio para poder obtener el texto de la imagen.
  - Dificultad en cómo poder encontrar los archivos más rápidos.
  - Dificultad en extraer un zip de docker usando unzip.
  - Dificultad en extraer la flag del texto encontrado en el flag.txt

## Referencias

- Cilleruelo, C. (2024a, abril 18). ¿Qué es el cifrado XOR? | KeepCoding Bootcamps. KeepCoding Bootcamps. <https://keepcoding.io/blog/que-es-el-cifrado-xor/>.
- Martí, G. (2022, 4 junio). El cifrado XOR - Gabriel Martí - Medium. Medium. <https://gabimarti.medium.com/el-cifrado-xor-f6f7367c8b9d>.
- Ballejos, L. (2025, 18 febrero). ¿Qué es el cifrado RC4? - NinjaOne. NinjaOne. <https://www.ninjaone.com/es/it-hub/endpoint-security/cifrado-rc4/>.
- Awati, R. (2024, 10 diciembre). What is a stream cipher? Search Security. <https://www.techtarget.com/searchsecurity/definition/stream-cipher>.
- ParejAbertzale. (2023, 26 noviembre). Algoritmo de cifrado ChaCha20 - ParejAbertzale - Medium. Medium. <https://medium.com/@parejaemi/algoritmo-de-cifrado-chacha20-119a6d7c19a7>.