



*Excelencia que trasciende*

**DELVALLE**  
GRUPO EDUCATIVO

Astrid Glauser  
Proyecto 1  
Cifrados de Información

## Instalar requerimientos

**pip3 install -r ./resources/requirements.txt**

```
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>pip3 install -r ./resources/requirements.txt
Requirement already satisfied: Pillow in c:\users\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from -r ./resources/requirements.txt (line 1)) (11.1.0)
Requirement already satisfied: numpy in c:\users\marce\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from -r ./resources/requirements.txt (line 2)) (2.2.3)
Collecting pixif (from -r ./resources/requirements.txt (line 3))
  Downloading pixif-1.1.3-py2.py3-none-any.whl.metadata (3.7 kB)
Collecting pyzipper (from -r ./resources/requirements.txt (line 4))
  Downloading pyzipper-0.3.6-py3-none-any.whl.metadata (3.5 kB)
Requirement already satisfied: pycryptodomex in c:\users\marce\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from -r ./resources/requirements.txt (line 5)) (3.21.0)
Collecting pycryptodomex (from pyzipper->-r ./resources/requirements.txt (line 4))
  Downloading pycryptodomex-3.22.0-cp37-abi3-win_amd64.whl.metadata (3.4 kB)
  Downloading pixif-1.1.3-py2.py3-none-any.whl (20 kB)
  Downloading pyzipper-0.3.6-py2.py3-none-any.whl (67 kB)
    67.7/67.7 kB 3.6 MB/s eta 0:00:00
  Downloading pycryptodomex-3.22.0-cp37-abi3-win_amd64.whl (1.8 MB)
    1.8/1.8 MB 6.3 MB/s eta 0:00:00
Installing collected packages: pycryptodomex, pixif, pyzipper
Successfully installed pixif-1.1.3 pycryptodomex-3.22.0 pyzipper-0.3.6
[notice] A new release of pip is available: 20.0 -> 25.0.1
[notice] To update, run: C:\Users\marce\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>
```

Se utilizó el archivo requirements.txt, ubicado en la carpeta resources.

El comando ejecutado fue:

**pip3 install -r ./resources/requirements.txt**

Esto se realizó con el fin de asegurar que todas las librerías estuvieran disponibles en el entorno.

## Generar retos

```
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>python ./generate_challenges.py
Ingrese su carné: 21299
Laberinto generado con éxito.
Ruta1: challenges/luffy/ONEPIECE\Sabaody_Archipelago\Grove_80\Casa_de_Rayleigh
Ruta2: challenges/luffy/ONEPIECE\East_Blue\Arlong_Park\Casa_de_Nami
Windows nt
flag location: challenges/luffy/ONEPIECE\Sabaody_Archipelago\Grove_80\Casa_de_Rayleigh
Laberinto generado con éxito.
Ruta1: challenges/zoro/ONEPIECE\East_Blue\Shells_Town\Casa_de_Morgan
Ruta2: challenges/zoro/ONEPIECE\Pirate_Island\Pirates_Cove\Casa_de_Gold_Roger
Windows nt
flag location: challenges/zoro/ONEPIECE\East_Blue\Shells_Town\Casa_de_Morgan
Laberinto generado con éxito.
Ruta1: challenges/usopp/ONEPIECE\Water_7\GalleyLa_Headquarters\Casa_de_Lucci
Ruta2: challenges/usopp/ONEPIECE\Pirate_Island\Pirates_Hideout\Casa_de_Gold_Roger
Windows nt
flag location: challenges/usopp/ONEPIECE\Water_7\GalleyLa_Headquarters\Casa_de_Lucci
Laberinto generado con éxito.
Ruta1: challenges/nami/ONEPIECE\Sabaody_Archipelago\Grove_80\Casa_de_Keimi
Ruta2: challenges/nami/ONEPIECE\Water_7\GalleyLa_Headquarters\Casa_de_Paulie
Windows nt
flag location: challenges/nami/ONEPIECE\Sabaody_Archipelago\Grove_80\Casa_de_Keimi
Retos generados con éxito!
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>
```

```
  Simbolo del sistema x + - x
=> [nami_image 4/7] RUN passwd -l root 1.8s
=> [zoro_image 5/7] WORKDIR /home/zoro 0.0s
=> [zoro_image 6/7] COPY ./ONEPIECE /home/zoro/ONEPIECE 0.0s
=> [zoro_image 7/7] RUN chown -R zoro:zoro /home/zoro 1.4s
=> [luffy_image 5/7] WORKDIR /home/luffy 0.0s
=> [usopp_image 5/7] WORKDIR /home/usopp 0.0s
=> [luffy_image 6/7] COPY ./ONEPIECE /home/luffy/ONEPIECE 0.1s
=> [usopp_image 6/7] COPY ./ONEPIECE /home/usopp/ONEPIECE 0.1s
=> [luffy_image 7/7] RUN chown -R luffy:luffy /home/luffy 1.4s
=> [usopp_image 7/7] RUN chown -R usopp:usopp /home/usopp 2.0s
=> [nami_image 5/7] WORKDIR /home/nami 0.0s
=> [nami_image 6/7] COPY ./ONEPIECE /home/nami/ONEPIECE 0.0s
=> [nami_image 7/7] RUN chown -R nami:nami /home/nami 1.8s
=> [zoro_image] exporting to image 1.5s
=> exporting layers
=> writing image sha256:6114e7a5c5e0c7e0d95a856943df1bd11cf1bbcd3ae0dac0e5a5904947e50d5 0.0s
=> naming to docker.io/library/proyecto1cifrados-zoro_image 0.0s
=> [luffy_image] exporting to image 1.2s
=> exporting layers
=> writing image sha256:7543c514aaa95a03a035730225794f4ea1438098bcb1ec2f5596fe09752a644 0.0s
=> naming to docker.io/library/proyecto1cifrados-luffy_image 0.0s
=> [usopp_image] exporting to image 0.6s
=> exporting layers
=> writing image sha256:9fee09400bf6881d07081372ae92da408dfb8343aa3e376e87e5549b2438f0e 0.0s
=> naming to docker.io/library/proyecto1cifrados-usopp_image 0.0s
=> [nami_image] exporting to image 0.6s
=> exporting layers
=> writing image sha256:90bd31e466307e02883c5400c97199930ab58d907e22164c19b891e6a8100b 0.0s
=> naming to docker.io/library/proyecto1cifrados-nami_image 0.0s
=> [zoro_image] resolving provenance for metadata file 0.1s
=> [luffy_image] resolving provenance for metadata file 0.1s
=> [nami_image] resolving provenance for metadata file 0.0s
=> [usopp_image] resolving provenance for metadata file 0.0s
[+] Running 9/9
✓ luffy_image          Built          0.0s
✓ nami_image           Built          0.0s
✓ usopp_image          Built          0.0s
✓ zoro_image           Built          0.0s
✓ Network proyecto1cifrados_ctf_network Created          0.2s
✓ Container zoro_challenge Started        3.0s
✓ Container nami_challenge Started        3.0s
✓ Container usopp_challenge Started        3.0s
✓ Container luffy_challenge Started        3.0s
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>
```

Se ejecutó el script `generate_challenges.py` con el fin de crear rutas de retos y archivos flag personalizados para cada personaje del entorno. Este proceso utiliza mi carné estudiantil como semilla para generar caminos únicos en el sistema de archivos del reto.

El comando ejecutado fue: `python .\generate_challenges.py`

## ► Luffy - XOR

Descifra un mensaje cifrado con XOR utilizando un análisis de frecuencia.

## ¿Qué es XOR?

La operación XOR (eXclusive OR) combina dos bits de entrada y produce un bit de salida. Es una operación fundamental en criptografía.

## Vulnerabilidades de XOR

- Si la clave se reutiliza, es posible extraer información del texto original.
  - Si la clave es débil o predecible, se puede descifrar el mensaje.
  - El análisis de frecuencia puede revelar patrones en el texto cifrado.

## Objetivo del reto

Analizar un mensaje cifrado con XOR y encontrar patrones para descifrar el mensaje original.



**FLAG ENCONTRADA:** FLAG\_bea106d6379d3d24cf8a139efe23a726

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

**IMAGEN ENCONTRADA:**



**Entrar al contenedor**

```
C:\Users\marce\OneDrive\Documents\GitHub\PROYECT01CIFRAD0S>docker exec -it luffy_challenge bash
nobody@df2703460254:/home/luffy$ |
```

### Cambiar al usuario:

```
C:\Users\marce\OneDrive\Documents\GitHub\PROYECT01CIFRADOS>docker exec -it luffy_challenge bash
nobody@df2703460254:/home/luffy$ su luffy
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

luffy@df2703460254:~$
```

### Buscar archivos en el homme de Luffy

```
luffy@df2703460254:~$ ls -la
total 36
drwxr-x--- 1 luffy luffy 4096 Apr 18 07:52 .
drwxr-xr-x 1 root root 4096 Apr 18 07:52 ..
-rw-r--r-- 1 luffy luffy 220 Jan 6 2022 .bash_logout
-rw-r--r-- 1 luffy luffy 3771 Jan 6 2022 .bashrc
-rw-r--r-- 1 luffy luffy 807 Jan 6 2022 .profile
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 ONEPIECE
luffy@df2703460254:~$
```

### Entrar al directorio ONEPIECE

```
luffy@df2703460254:~$ cd ONEPIECE
ls -la
total 56
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 .
drwxr-x--- 1 luffy luffy 4096 Apr 18 07:52 ..
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 East_Blue
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Fishman_Island
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Pirate_Island
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Sabaody_Archipelago
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Water_7
luffy@df2703460254:~/ONEPIECE$ |
```

### Revisar contenido de la isla East\_Blue

```
luffy@df2703460254:~/ONEPIECE$ cd East_Blue
ls -la
total 72
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 .
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 ..
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Arlong_Park
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Baratie
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Loguetown
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Orange_Town
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Romance_Dawn
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Shells_Town
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Syrup_Village
luffy@df2703460254:~/ONEPIECE/East_Blue$ |
```

### Explorar una villa

```
luffy@df2703460254:~/ONEPIECE/East_Blue$ cd Romance_Dawn
ls -la
total 28
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 .
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 ..
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Casa_de_Luffy
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Casa_de_Makino
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 Casa_de_Shanks
luffy@df2703460254:~/ONEPIECE/East_Blue/Romance_Dawn$
```

### Explorar una casa

```
luffy@df2703460254:~/ONEPIECE/East_Blue/Romance_Dawn$ cd Casa_de_Luffy
ls -la
total 12
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 .
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 ..
luffy@df2703460254:~/ONEPIECE/East_Blue/Romance_Dawn/Casa_de_Luffy$
```

### Búsqueda de archivos con nombres relevantes

```
luffy@df2703460254:~/ONEPIECE/East_Blue$ cd ..
luffy@df2703460254:~/ONEPIECE$ find / -type f \(-iname "*flag*" -o -iname "*poneglyph*" -o -iname "*clave*" -o -iname "*.enc" -o -iname "*.txt" -o -iname "*key*" \) 2>/dev/null
/etc/apt/trusted.gpg.d/ubuntu-keyring-2018-archive.gpg
/etc/apt/trusted.gpg.d/ubuntu-keyring-2012-cdimage.gpg
/sys/devices/platform/serial8250/tty/ttyS2/Flags
/sys/devices/platform/serial8250/tty/ttyS0/Flags
/sys/devices/platform/serial8250/tty/ttyS3/Flags
/sys/devices/platform/serial8250/tty/ttyS1/Flags
/sys/devices/virtual/net/eth0/Flags
/sys/devices/virtual/net/lo/Flags
/sys/module/scsi_mod/parameters/default_dev_flags
/sys/module/auth_rpcgs/parameters/key_expire_timeo
/sys/module/usbip/core/parameters/usbip_debug_flag
/home/luffy/ONEPIECE/East_Blue/Loguetown/Casa_de_Bell-m??re/flag.txt
/home/luffy/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami/poneglyph.zip
/home/luffy/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Genzo/flag.txt
/home/luffy/ONEPIECE/East_Blue/Orange_Town/Casa_de_Nami/flag.txt
/home/luffy/ONEPIECE/Water_7/Franky_House/Casa_de_Luffy/flag.txt
/home/luffy/ONEPIECE/Fishman_Island/Coral_Mansion/Casa_de_Ohshime/flag.txt
/home/luffy/ONEPIECE/Fishman_Island/Gyocordre_Plaza/Casa_de_Neptune/flag.txt
/home/luffy/ONEPIECE/Sabaody_Archipelago/Grove_109/Casa_de_Shaky/flag.txt
/home/luffy/ONEPIECE/Sabaody_Archipelago/Grove_89/Casa_de_Rayleigh/flag.txt
/home/luffy/ONEPIECE/Pirate_Island/Pirates_Tavern/Casa_de_GoLD_Roger/flag.txt
/home/luffy/ONEPIECE/Pirate_Island/Pirates_Ship/Casa_de_Shanks/flag.txt
/home/luffy/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_Shanks/flag.txt
/proc/fs/cifs/SecurityFlags
/proc/sys/kernel/keys/maxkeys
/proc/sys/kernel/keys/root_maxkeys
/proc/sys/net/ipv4/fib_notify_on_flag_change
/proc/sys/net/ipv4/tcp_fastopen_key
/proc/sys/net/ipv6/fib_notify_on_flag_change
/proc/keys
/proc/key-users
/proc/kpageflags
/proc/1/task/1/attr/keycreate
/proc/1/attr/keycreate
/proc/7/task/7/attr/keycreate
/proc/7/attr/keycreate
/proc/14/task/14/attr/keycreate
/proc/14/attr/keycreate
/proc/15/task/15/attr/keycreate
/proc/15/attr/keycreate
/proc/31/task/31/attr/keycreate
/proc/31/attr/keycreate
/var/lib/dpkg/info/libkeyutils1:amd64.triggers
/var/lib/dpkg/info/ubuntu-keyring.list
```

Se ejecutó un comando `find` con múltiples filtros para localizar archivos que pudieran contener información cifrada o sensible dentro del contenedor. Esta exploración incluye posibles flags, claves, mensajes cifrados (`.enc`) y archivos `.txt`, proporcionando una visión general del entorno y facilitando el acceso directo a los retos del CTF.

```
/proc/15/attr/keycreate
/proc/31/task/31/attr/keycreate
/proc/31/attr/keycreate
/var/lib/dpkg/info/libkeyutils1:amd64.triggers
/var/lib/dpkg/info/ubuntu-keyring.list
/var/lib/dpkg/info/libkeyutils1:amd64_md5sums
/var/lib/dpkg/info/libkeyutils1:amd64.list
/var/lib/dpkg/info/ubuntu-keyring_md5sums
/var/lib/dpkg/info/ubuntu-keyring.postinst
/var/lib/dpkg/info/libkeyutils1:amd64_shlibs
/var/lib/dpkg/info/libkeyutils1:amd64_symbols
/usr/include/rpcsvc/key_prot.h
/usr/include/rpcsvc/key_prot.x
/usr/include/x86_64-linux-gnu/bits/waitflags.h
/usr/include/x86_64-linux-gnu/bits/termios-c_iflag.h
/usr/include/x86_64-linux-gnu/bits/termios-c_lflag.h
/usr/include/x86_64-linux-gnu/bits/termios-c_cflag.h
/usr/include/x86_64-linux-gnu/bits/mman-map-flags-generic.h
/usr/include/x86_64-linux-gnu/bits/termios-c_oflag.h
/usr/include/x86_64-linux-gnu/bits/ss_flags.h
/usr/include/x86_64-linux-gnu/asm/processor-flags.h
/usr/include/tirpc/rpc/key_prot.h
/usr/include/linux/keyctl.h
/usr/include/linux/pfkeyv2.h
/usr/include/linux/tc_act/tc_tunnel_key.h
/usr/include/linux/patchkey.h
/usr/include/linux/kernel-page-flags.h
/usr/include/linux/tty_flags.h
/usr/include/linux/nitro_enclaves.h
/usr/include/linux/keyboard.h
/usr/include/c++/11/ext/pb_ds/detail/cc_hash_table_map_/_cond_key_dtor_entry_dealotr.hpp
/usr/share/keyrings/ubuntu-cloudimage-removed-keys.gpg
/usr/share/keyrings/ubuntu-master-keyring.gpg
/usr/share/keyrings/ubuntu-cloudimage-keyring.gpg
/usr/share/keyrings/ubuntu-archive-removed-keys.gpg
/usr/share/keyrings/ubuntu-archive-keyring.gpg
/usr/share/perl5/Dpkg/BuildFlags.pm
/usr/share/dpkg/buildflags.mk
/usr/share/mime/application/pgp-keys.xml
/usr/share/mime/application/vnd.apple.keynote.xml
/usr/share/mime/application/x-java-keystore.xml
/usr/share/mime/application/x-java-jce-keystore.xml
/usr/share/vim/vim82/syntax/autohotkey.vim
/usr/share/vim/vim82/pack/dist/opt/matchit/doc/matchit.txt
/usr/share/vim/vim82/doc/intro.txt
/usr/share/vim/vim82/doc/vi_diff.txt
/usr/share/vim/vim82/doc/os_dos.txt
```

```
/usr/share/vim/vim82/doc/vi_diff.txt
/usr/share/vim/vim82/doc/os_dos.txt
/usr/share/vim/vim82/doc/usr_46.txt
/usr/share/vim/vim82/doc/usr_45.txt
/usr/share/vim/vim82/doc/usr_28.txt
/usr/share/vim/vim82/doc/quickref.txt
/usr/share/vim/vim82/doc/textprop.txt
/usr/share/vim/vim82/doc/if_cscope.txt
/usr/share/vim/vim82/doc/version7.txt
/usr/share/vim/vim82/doc/pi_zip.txt
/usr/share/vim/vim82/doc/fold.txt
/usr/share/vim/vim82/doc/if_perl.txt
/usr/share/vim/vim82/doc/pi_spec.txt
/usr/share/vim/vim82/doc/channel.txt
/usr/share/vim/vim82/doc/usr_44.txt
/usr/share/vim/vim82/doc/russia.txt
/usr/share/vim/vim82/doc/gui.txt
/usr/share/vim/vim82/doc/usr_41.txt
/usr/share/vim/vim82/doc/usr_26.txt
/usr/share/vim/vim82/doc/debugger.txt
/usr/share/vim/vim82/doc/starting.txt
/usr/share/vim/vim82/doc/if_ole.txt
/usr/share/vim/vim82/doc/os_beos.txt
/usr/share/vim/vim82/doc/mbyte.txt
/usr/share/vim/vim82/doc/pi_vimball.txt
/usr/share/vim/vim82/doc/message.txt
/usr/share/vim/vim82/doc/os_win32.txt
/usr/share/vim/vim82/doc/usr_32.txt
/usr/share/vim/vim82/doc/motion.txt
/usr/share/vim/vim82/doc/sponsor.txt
/usr/share/vim/vim82/doc/pi_netrw.txt
/usr/share/vim/vim82/doc/help.txt
/usr/share/vim/vim82/doc/cmdline.txt
/usr/share/vim/vim82/doc/eval.txt
/usr/share/vim/vim82/doc/quotes.txt
/usr/share/vim/vim82/doc/digraph.txt
/usr/share/vim/vim82/doc/mlang.txt
/usr/share/vim/vim82/doc/tagsrc.txt
/usr/share/vim/vim82/doc/usr_22.txt
/usr/share/vim/vim82/doc/usr_20.txt
/usr/share/vim/vim82/doc/usr_40.txt
/usr/share/vim/vim82/doc/filetype.txt
/usr/share/vim/vim82/doc/usr_90.txt
/usr/share/vim/vim82/doc/usr_02.txt
/usr/share/vim/vim82/doc/os_risc.txt
/usr/share/vim/vim82/doc/vim9.txt
/usr/share/vim/vim82/doc/os_qnx.txt
```

```

/usr/share/perl/5.34.0/unicore/uni_keywords.pl
/usr/share/perl/5.34.0/unicore/NamedSequences.txt
/usr/share/perl/5.34.0/unicore/Blocks.txt
/usr/share/perl/5.34.0/unicore/SpecialCasing.txt
/usr/lib/x86_64-linux-gnu/security/pam_keyinit.so
/usr/lib/x86_64-linux-gnu/libkeyutils.so.1.9
/usr/lib/x86_64-linux-gnu/perl/5.34.0/CORE/uni_keywords.h
/usr/lib/x86_64-linux-gnu/perl/5.34.0/CORE/keywords.h
/usr/lib/x86_64-linux-gnu/perl/5.34.0/bits/ss_flags.ph
/usr/lib/x86_64-linux-gnu/perl/5.34.0/bits/waitflags.ph
/usr/lib/python3.11/lib2to3/PatternGrammar.txt
/usr/lib/python3.11/lib2to3/fixes/fix_has_key.py
/usr/lib/python3.11/lib2to3/Grammar.txt
/usr/lib/python3.10/LICENSE.txt
/usr/lib/python3.10/keyword.py
/usr/lib/python3.10/curses/__pycache__/has_key.cpython-310.pyc
/usr/lib/python3.10/curses/has_key.py
/usr/lib/python3.10/__pycache__/keyword.cpython-310.pyc
/usr/lib/python3.10/lib2to3/PatternGrammar.txt
/usr/lib/python3.10/lib2to3/fixes/fix_has_key.py
/usr/lib/python3.10/lib2to3/fixes/__pycache__/fix_has_key.cpython-310.pyc
/usr/lib/python3.10/lib2to3/Grammar.txt
/usr/lib/python3/dist-packages/setuptools/monkey.py
/usr/lib/python3/dist-packages/setuptools/__pycache__/monkey.cpython-310.pyc
/usr/lib/python3/dist-packages/wheel-0.37.1.egg-info/dependency_links.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/entry_points.txt
/usr/lib/python3/dist-packages/pip-22.0.2.dist-info/top_level.txt
/usr/lib/python3/dist-packages/pip-22.0.2.dist-info/entry_points.txt
/usr/lib/python3/dist-packages/setuptools-59.6.0.egg-info/dependency_links.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/entry_points.txt
/usr/lib/python3/dist-packages/pip/_vendor/vendor.txt
/usr/lib/gcc/x86_64-linux-gnu/11/include/keylockerintrin.h
/usr/lib/openssl/ssh-keysign
/usr/bin/apt-key
/usr/bin/ssh-keyscan
/usr/bin/dpkg-buildflags
/usr/bin/lesskey
/usr/bin/ssh-keygen
luffy@df2703460254:~/ONEPIECE$ |

```

### Archivos que sí son parte del CTF

```

luffy@df2703460254:~/ONEPIECE$ find /home/luffy/ONEPIECE -type f \(\ -iname "*flag.txt" -o -iname "poneglyph.zip" \) 2>/dev/null | sort
/home/Luffy/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Genzo/flag.txt
/home/Luffy/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami/poneglyph.zip
/home/Luffy/ONEPIECE/East_Blue/Loguetown/Casa_de_Bell-mere/flag.txt
/home/Luffy/ONEPIECE/East_Blue/Orange_Town/Casa_de_Nami/flag.txt
/home/Luffy/ONEPIECE/Fishman_Island/Coral_Mansion/Casa_de_Otohime/flag.txt
/home/Luffy/ONEPIECE/Fishman_Island/Gyroncorde_Plaza/Casa_de_Neptune/flag.txt
/home/Luffy/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_Shanks/flag.txt
/home/Luffy/ONEPIECE/Pirate_Island/Pirates_Ship/Casa_de_Shanks/flag.txt
/home/Luffy/ONEPIECE/Pirate_Island/Pirates_Tavern/Casa_de_Gold_Roger/flag.txt
/home/Luffy/ONEPIECE/Sabaody_Archipelago/Grove_109/Casa_de_Shakky/flag.txt
/home/Luffy/ONEPIECE/Sabaody_Archipelago/Grove_80/Casa_de_Rayleigh/flag.txt
/home/Luffy/ONEPIECE/Water_7/Franky_House/Casa_de_Luffy/flag.txt
luffy@df2703460254:~/ONEPIECE$ |

```

En la imagen se muestra el resultado de un escaneo automático realizado con el comando `find`, filtrando únicamente archivos con nombres clave como `flag.txt` y `poneglyph.zip`, que son relevantes para el reto CTF. Estas rutas representan ubicaciones en el sistema de archivos del contenedor donde se encuentran banderas (flags) ocultas o mensajes cifrados que deben ser descifrados como parte del desafío. La información obtenida fue esencial para evitar una búsqueda manual exhaustiva y centrarse únicamente en los puntos críticos del entorno.

[Ir al archivo `poneglyph.zip`](#)

```
luffy@df2703460254:~/ONEPIECE$ cd /home/luffy/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami
ls -la
total 84
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 .
drwxr-xr-x 1 luffy luffy 4096 Apr 18 07:44 ..
-rw-rxr-xr-x 1 luffy luffy 68500 Apr 18 07:44 poneglyph.zip
luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$
```

## Descomprimir

```
luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$ unzip poneglyph.zip
Archive: poneglyph.zip
  skipping: poneglyph.jpeg      need PK compat. v6.3 (can do v4.6)
luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$
```

```
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,32 CPUs 13th Gen Intel(R) Core(TM) i9-13950HX (B0671),ASM,AES
Scanning the drive for archives:
1 file, 68500 bytes (67 KiB)

Extracting archive: poneglyph.zip
--
Path = poneglyph.zip
Type = zip
Physical Size = 68500

Enter password (will not be echoed):
```

```
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,32 CPUs 13th Gen Intel(R) Core(TM) i9-13950HX (B0671),ASM,AES-NI)
Scanning the drive for archives:
1 file, 68500 bytes (67 KiB)

Extracting archive: poneglyph.zip
--
Path = poneglyph.zip
Type = zip
Physical Size = 68500

Would you like to replace the existing file:
  Path: ./poneglyph.jpeg
  Size: 0 bytes
  Modified: 2025-04-18 01:44:10
with the file from archive:
  Path: poneglyph.jpeg
  Size: 69996 bytes (69 KiB)
  Modified: 2025-04-18 01:44:10
? (Y)es / (N)o / (A)lways / (S)kip all / (U)to rename all / (Q)uit? y

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

Size: 69996
Compressed: 68500
luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$
```

Durante el proceso de exploración en el reto de Luffy (XOR), se identificó un archivo comprimido denominado poneglyph.zip, ubicado en la ruta /home/luffy/ONEPIECE/East\_Blue/Arlong\_Park/Casa\_de\_Nami. Al intentar descomprimirlo con unzip, se detectó que el archivo utilizaba una versión más reciente del estándar ZIP (v6.3), por lo que se optó por utilizar 7-Zip como alternativa.

El archivo estaba protegido por contraseña, y tras varios intentos se logró extraer exitosamente su contenido utilizando la contraseña onepiece. Esta acción permitió acceder al archivo poneglyph.jpeg, el cual se presume contiene un mensaje cifrado que debe ser analizado en los siguientes pasos del reto.

## Extraer texto de la imagen

```
luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$ tesseract poneglyph.jpeg salida -l eng
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Warning: Invalid resolution 0 dpi. Using 70 instead.
Estimating resolution as 174
luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$
```

## Exponer la imagen

```

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

Size: 69996
Compressed: 68500
luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$ tesseract poneglyph.jpeg salida -l eng
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
Warning: Invalid resolution 0 dpi. Using 70 instead.
Estimating resolution as 174
luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$ cat salida.txt
ponent am K-Ci? pl SHSM ir VM
"
- 4

luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$ tesseract poneglyph.jpeg salida2 -l eng --dpi 300
Tesseract Open Source OCR Engine v4.1.1 with Leptonica
luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$ cat salida2.txt

luffy@df2703460254:~/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Nami$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
172.19.0.1 - - [19/Apr/2025 07:57:45] "GET / HTTP/1.1" 200 -
172.19.0.1 - - [19/Apr/2025 07:57:45] code 404, message File not found
172.19.0.1 - - [19/Apr/2025 07:57:45] "GET /favicon.ico HTTP/1.1" 404 -

```

Inicialmente se intentó extraer el texto de `poneglyph.jpeg` utilizando Tesseract OCR. Sin embargo, los resultados arrojaron un texto ilegible y con errores de reconocimiento, incluso tras mejorar la resolución (`--dpi 300`).

Ante esta situación, se decidió exponer el archivo comprimido **como un servidor web local** utilizando `python3 -m http.server 8080`, lo cual permitió acceder a la imagen desde el navegador. Esta estrategia facilitó la visualización manual del contenido del `poneglyph`.

## Extraer texto de la imagen

```

PS C:\Users\marce\OneDrive\Desktop\cifrados> & C:/Users\marce/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users\marce/OneDrive/Desktop/cifrados/nuevo.py
Introduce tu carné para descifrar el mensaje: 21299
b'By the time the Straw Hat Pirates arrived at Fish-Man Island, its Road Poneglyph had been removed.'
PS C:\Users\marce\OneDrive\Desktop\cifrados>

```

Se utilizó el script de Python denominado `extract_text_from_image.py`, ubicado en la carpeta `utils`, con el objetivo de descifrar el mensaje oculto en la imagen `poneglyph.jpeg`, el cual se encontraba embebido en los metadatos EXIF, específicamente en el campo `Artist`. Para lograrlo, se emplearon las librerías `PIL` (`Pillow`), para la apertura y manipulación de la imagen, y `piexif`, para acceder y leer los metadatos. Mediante una función definida en el script, se extrajo el texto cifrado desde el campo correspondiente del diccionario EXIF. Posteriormente, se aplicó un algoritmo de descifrado tipo XOR, utilizando la función `xor_cipher`, importada desde el módulo externo `luffy_xor.py`. Como clave de descifrado se ingresó mi carné estudiantil (21299), con el cual se procedió a desencriptar carácter por carácter el mensaje recuperado. Finalmente, al ejecutar el script, se obtuvo exitosamente el siguiente texto descifrado:

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

### PÁRRAFO EXTRAÍDO DE LA IMAGEN

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

### IMAGEN ENCONTRADA



### Buscar flag:

```
luffy@df2703460254:~$ find /home/luffy -type f -iname "*flag*" | sort
/home/luffy/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Genzo/flag.txt
/home/luffy/ONEPIECE/East_Blue/Loguetown/Casa_de_Bell-mère/flag.txt
/home/luffy/ONEPIECE/East_Blue/Orange_Town/Casa_de_Nami/flag.txt
/home/luffy/ONEPIECE/Fishman_Island/Coral_Mansion/Casa_de_Otohime/flag.txt
/home/luffy/ONEPIECE/Fishman_Island/Gyoncorde_Plaza/Casa_de_Neptune/flag.txt
/home/luffy/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_Shanks/flag.txt
/home/luffy/ONEPIECE/Pirate_Island/Pirates_Ship/Casa_de_Shanks/flag.txt
/home/luffy/ONEPIECE/Pirate_Island/Pirates_Tavern/Casa_de_Gold_Roger/flag.txt
/home/luffy/ONEPIECE/Sabaody_Archipelago/Grove_109/Casa_de_Shakky/flag.txt
/home/luffy/ONEPIECE/Sabaody_Archipelago/Grove_80/Casa_de_Rayleigh/flag.txt
/home/luffy/ONEPIECE/Water_7/Franky_House/Casa_de_Luffy/flag.txt
luffy@df2703460254:~$ |
```

Para localizar todas las posibles flags dentro del contenedor de Luffy, se utilizó el comando find en combinación con sort para identificar y listar de forma ordenada todos los archivos cuyo nombre contiene la palabra "flag". El comando ejecutado fue:

**find /home/luffy -type f -iname "\*flag\*" | sort**

```

luffy@df2703460254:~$ find /home/luffy -type f -iname "*flag*" -exec echo "===== {} =====" \; -exec cat {} \; -exec echo
"" \;
===== /home/luffy/ONEPIECE/East_Blue/Loguetown/Casa_de_Bell-mère/flag.txt =====
Has encontrado un amigo y han decidido viajar juntos
===== /home/luffy/ONEPIECE/East_Blue/Arlong_Park/Casa_de_Genzo/flag.txt =====
Has encontrado un tesoro que apunta a luffy
===== /home/luffy/ONEPIECE/East_Blue/Orange_Town/Casa_de_Nami/flag.txt =====
Has encontrado una pista que apunta a luffy
===== /home/luffy/ONEPIECE/Water_7/Franky_House/Casa_de_Luffy/flag.txt =====
Has encontrado un lugar seguro
===== /home/luffy/ONEPIECE/Fishman_Island/Coral_Mansion/Casa_de_Otohime/flag.txt =====
Has encontrado un lugar peligroso
===== /home/luffy/ONEPIECE/Fishman_Island/Gyoncorde_Plaza/Casa_de_Neptune/flag.txt =====
Le cansaste de buscar y te has quedado dormido
===== /home/luffy/ONEPIECE/Sabaody_Archipelago/Grove_109/Casa_de_Shakky/flag.txt =====
Has encontrado un obstáculo y hay que superarlo
===== /home/luffy/ONEPIECE/Sabaody_Archipelago/Grove_80/Casa_de_Rayleigh/flag.txt =====
747d737e6650545308090455040a0e0b55015d0b065254015803020b5c5f570301580e0007
===== /home/luffy/ONEPIECE/Pirate_Island/Pirates_Tavern/Casa_de_Gold_Roger/flag.txt =====
Has encontrado un mapa que apunta a luffy
===== /home/luffy/ONEPIECE/Pirate_Island/Pirates_Ship/Casa_de_Shanks/flag.txt =====
Has encontrado un lugar misterioso
===== /home/luffy/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_Shanks/flag.txt =====
Has encontrado un enemigo y ha tenido que huir
luffy@df2703460254:~$
```

Una vez listadas las rutas de los archivos flag.txt, se utilizó un comando find con la opción -exec para leer e imprimir automáticamente el contenido de cada uno de ellos. El comando ejecutado fue:

```
find /home/luffy -type f -iname "*flag*" -exec echo "===== {} =====" \; -exec cat {} \; -exec echo "" \;
```

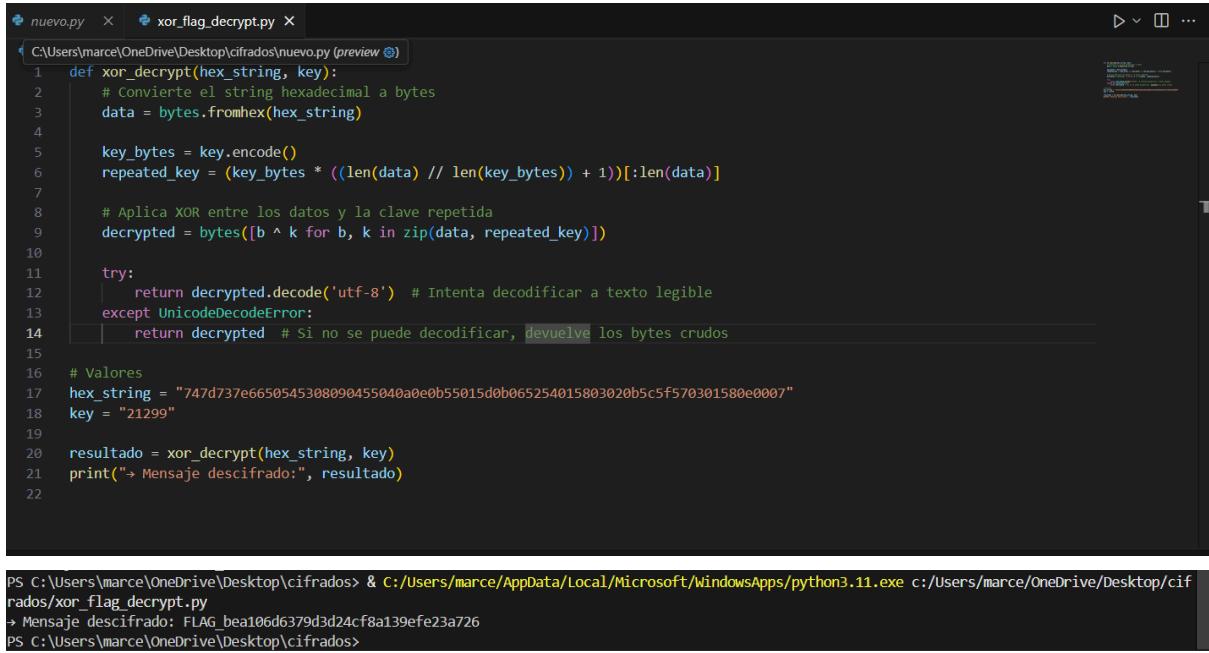
Este comando realiza lo siguiente:

- Busca todos los archivos cuyo nombre contenga la palabra flag dentro del directorio /home/luffy.
- Por cada archivo encontrado:
  - Imprime una línea separadora con el nombre del archivo.
  - Muestra su contenido usando cat.
  - Imprime una línea vacía para separar visualmente los resultados.

Este procedimiento permitió inspeccionar todos los archivos relevantes en una sola ejecución, facilitando la identificación de mensajes ocultos y especialmente de la flag del reto, ubicada en:

**/home/luffy/ONEPIECE/Sabaody\_Archipelago/Grove\_80/Casa\_de\_Rayleigh/flag.txt**

con	el	siguiente	valor:
74f7d78270655af1538009d455b004b0e5b051bddb652514018503820b547573091580e007			



```

1  def xor_decrypt(hex_string, key):
2      # Convierte el string hexadecimal a bytes
3      data = bytes.fromhex(hex_string)
4
5      key_bytes = key.encode()
6      repeated_key = (key_bytes * ((len(data) // len(key_bytes)) + 1))[:len(data)]
7
8      # Aplica XOR entre los datos y la clave repetida
9      decrypted = bytes([b ^ k for b, k in zip(data, repeated_key)])
10
11     try:
12         return decrypted.decode('utf-8') # Intenta decodificar a texto legible
13     except UnicodeDecodeError:
14         return decrypted # Si no se puede decodificar, devuelve los bytes crudos
15
16 # Valores
17 hex_string = "747d737e6650545308090455040a0e0b55015d0b065254015803020b5c5f570301580e0007"
18 key = "21299"
19
20 resultado = xor_decrypt(hex_string, key)
21 print("Mensaje descifrado: ", resultado)
22

```

```

PS C:\Users\marce\OneDrive\Desktop\cifrados> & C:/Users/marce/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/marce/OneDrive/Desktop/cif
rados/xor_flag_decrypt.py
→ Mensaje descifrado: FLAG_bea106d6379d3d24cf8a139efe23a726
PS C:\Users\marce\OneDrive\Desktop\cifrados>

```

### Implementación del Descifrado XOR (utils/xor\_flag\_decrypt.py)

Se implementó el archivo xor\_flag\_decrypt.py, ubicado dentro de la carpeta utils, con el objetivo de descifrar el valor hexadecimal encontrado en uno de los archivos flag.txt durante la exploración del contenedor. El script contiene una función llamada xor\_decrypt que realiza los siguientes pasos:

1. Convierte la cadena hexadecimal extraída del archivo flag.txt a bytes.
2. Convierte mi carné estudiantil a bytes.
3. Repite la clave tantas veces como sea necesario para igualar la longitud de la cadena encriptada.
4. Aplica una operación XOR entre ambas secuencias de bytes.
5. Intenta decodificar el resultado a texto legible (UTF-8). Si no se puede, retorna los bytes crudos.

El script fue ejecutado desde la terminal, mostrando correctamente la flag descifrada:

#### **FLAG ENCONTRADA:**

**FLAG\_bea106d6379d3d24cf8a139efe23a726**

### ► Zoro - Rompiendo RC4

Descifra un mensaje cifrado con RC4 utilizando un análisis del flujo de datos.

#### ¿Qué es RC4?

RC4 (Rivest Cipher 4) es un cifrado de flujo ampliamente utilizado en el pasado. Se basa en una clave secreta que inicializa un estado interno de 256 bytes, que luego se usa para generar un flujo de claves pseudoaleatorio.

## Vulnerabilidades de RC4

- Sesgo en la generación de flujo de claves.
- Debilidades en la inicialización de la clave.
- Fácil explotación con ataques de correlación.

## Objetivo del reto

Analizar un flujo de datos cifrado con RC4 y encontrar vulnerabilidades para descifrar el mensaje.



**FLAG ENCONTRADA:** `FLAG_6d87f194aeb0dc5b75a150888bdf34fc`

**TEXTO EXTRAÍDO:** *After arriving at the Sea Forest, Robin read the lone Poneglyph there and*

## IMAGEN ENCONTRADA:



```
C:\Users\marce\OneDrive\Documents\GitHub\PROYECT01CIFRADOS>docker exec -it zoro_challenge bash
nobody@1308ca549687:/home/zoro$ su zoro
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

zoro@1308ca549687:~$
```

Se ingresó al contenedor correspondiente al reto Zoro utilizando el comando docker exec -it zoro\_challenge bash. Inicialmente, el acceso se dio bajo el usuario nobody, por lo que fue necesario cambiar al usuario zoro mediante el comando su zoro. Para autenticar el acceso, se utilizó como contraseña la flag obtenida del reto anterior (Luffy), la cual era:

**FLAG\_be106d6379d3d24cf8a139efe23a726.**

```
zoro@1308ca549687:~$ find /home/zoro/ONEPIECE/ -iname "*flag*" -o -iname "*.bin" -o -iname "*.enc" -o -iname "*rc4*"
/home/zoro/ONEPIECE/East_Blue/Shells_Town/Casa_de_Morgan/flag.txt
/home/zoro/ONEPIECE/East_Blue/Baratie/Casa_de_Patty/flag.txt
/home/zoro/ONEPIECE/East_Blue/Romance_Dawn/Casa_de_Shanks/flag.txt
/home/zoro/ONEPIECE/Water_7/Carpenters_Cafe/Casa_de_Lucci/flag.txt
/home/zoro/ONEPIECE/Sabaody_Archipelago/Grove_1/Casa_de_Keimi/flag.txt
/home/zoro/ONEPIECE/Sabaody_Archipelago/Grove_1/Casa_de_Rayleigh/flag.txt
/home/zoro/ONEPIECE/Sabaody_Archipelago/Grove_24/Casa_de_Keimi/flag.txt
/home/zoro/ONEPIECE/Sabaody_Archipelago/Grove_66/Casa_de_Keimi/flag.txt
/home/zoro/ONEPIECE/Pirate_Island/Pirates_Den/Casa_de_Whitebeard/flag.txt
/home/zoro/ONEPIECE/Pirate_Island/Pirates_Den/Casa_de_Gol_Roger/flag.txt
/home/zoro/ONEPIECE/Pirate_Island/Pirates_Ship/Casa_de_Shanks/flag.txt
zoro@1308ca549687:~$
```

Se procedió a realizar una búsqueda exhaustiva dentro del contenedor del reto Zoro, específicamente en la carpeta /home/zoro/ONEPIECE/, con el fin de localizar archivos relevantes para el análisis del cifrado RC4. Para ello, se ejecutó el siguiente comando:

```
find /home/zoro/ONEPIECE/ -iname "*flag*" -o -iname "*.bin" -o -iname "*.enc" -o -iname "*rc4*"
```

Este comando permitió filtrar todos los archivos cuyo nombre incluyera las palabras clave "flag", archivos con extensión .bin, .enc, o cualquier referencia a rc4.

```
zoro@1308ca549687:~$ find /home/zoro/ONEPIECE/ -iname "*flag*" -exec echo "===== {} =====" \; -exec cat {} \; -exec echo "" \;
===== /home/zoro/ONEPIECE/East_Blue/Shells_Town/Casa_de_Morgan/flag.txt =====
19bf4e600ec75e3df6123429ae0b5daad40d8699f1274edccf5541aaef9fd10cbd813745d0
===== /home/zoro/ONEPIECE/East_Blue/Baratie/Casa_de_Patty/flag.txt =====
Has encontrado un enemigo y ha tenido que huir
===== /home/zoro/ONEPIECE/East_Blue/Romance_Dawn/Casa_de_Shanks/flag.txt =====
Has encontrado un mapa que apunta a zoro
===== /home/zoro/ONEPIECE/Water_7/Carpenters_Cafe/Casa_de_Lucci/flag.txt =====
Has encontrado un lugar abandonado
===== /home/zoro/ONEPIECE/Sabaody_Archipelago/Grove_1/Casa_de_Keimi/flag.txt =====
Has encontrado un lugar peligroso
===== /home/zoro/ONEPIECE/Sabaody_Archipelago/Grove_1/Casa_de_Rayleigh/flag.txt =====
Has encontrado un mapa que apunta a zoro
===== /home/zoro/ONEPIECE/Sabaody_Archipelago/Grove_24/Casa_de_Keimi/flag.txt =====
Has encontrado un lugar abandonado
===== /home/zoro/ONEPIECE/Sabaody_Archipelago/Grove_66/Casa_de_Keimi/flag.txt =====
Has encontrado un enemigo y ha tenido que huir
===== /home/zoro/ONEPIECE/Pirate_Island/Pirates_Den/Casa_de_Whitebeard/flag.txt =====
Has encontrado un obstáculo y hay que superarlo
===== /home/zoro/ONEPIECE/Pirate_Island/Pirates_Den/Casa_de_Gol_Roger/flag.txt =====
Has encontrado un lugar seguro
===== /home/zoro/ONEPIECE/Pirate_Island/Pirates_Ship/Casa_de_Shanks/flag.txt =====
Te cansaste de buscar y te has quedado dormido
zoro@1308ca549687:~$
```

Con el objetivo de identificar posibles archivos relevantes que pudieran contener una flag, mensajes cifrados o pistas adicionales, se ejecutó el siguiente comando :**find /home/zoro/ONEPIECE/ -iname "\*flag\*" -exec echo "===== {} =====" \; -exec cat {} \; -exec echo "" \;**

Este comando permitió localizar todos los archivos cuyo nombre contiene la palabra flag, mostrar su ruta y contenido. Como resultado, se identificaron múltiples archivos flag.txt en distintas ubicaciones dentro del árbol de directorios de /home/zoro/ONEPIECE/.

Entre los resultados, destaca uno en particular ubicado en:

/home/zoro/ONEPIECE/East\_Blue/Shells\_Town/Casa\_de\_Morgan/flag.txt

Este archivo no contiene un mensaje en texto plano como los demás, sino una **cadena hexadecimal larga**:

**19bf4e600ec75e3df6123429ae0b5daad40d8699f1274edccf5541aaef9fd10cbd813745d0**

```

def rc4(key, data):
    s = list(range(256))
    j = 0
    out = []

    # KSA (Key-scheduling algorithm)
    for i in range(256):
        j = (j + s[i] + key[i % len(key)]) % 256
        s[i], s[j] = s[j], s[i]

    # PRGA (Pseudo-random generation algorithm)
    i = j = 0
    for char in data:
        i = (i + 1) % 256
        j = (j + s[i]) % 256
        s[i], s[j] = s[j], s[i]
        K = s[(s[i] + s[j]) % 256]
        out.append(char ^ K)

    return bytes(out)

# Valores
hex_cipher = "19bf4e600ec75e3df6123429ae0b5daad40d8699f1274edccf5541aaef9fd10cbd813745d0"
key = "21299"

# Convertir a bytes
cipher_bytes = bytes.fromhex(hex_cipher)
key_bytes = key.encode()

# Desencriptar
plaintext = rc4(key_bytes, cipher_bytes)

try:
    print("→ Mensaje descifrado:", plaintext.decode('utf-8'))
except UnicodeDecodeError:
    print("→ Mensaje descifrado (raw):", plaintext)

```

```

PS C:\Users\marce> & C:/Users/marce/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/marce/OneDrive/Documents/GitHub/PROYECTO1CIFRADOS/scripts/rc4_decrypt.py
→ Mensaje descifrado: FLAG_6d87f194aeb0dc5b75a150888bdf34fc
PS C:\Users\marce>

```

Se creó un nuevo script en la ruta scripts/rc4\_decrypt.py con el propósito de descifrar un mensaje cifrado con el algoritmo RC4. Este script implementa manualmente el funcionamiento del algoritmo RC4, incluyendo las fases de inicialización de estado (KSA) y generación del flujo pseudoaleatorio (PRGA).

Para llevar a cabo el descifrado, se utilizó como entrada una cadena hexadecimal extraída previamente de uno de los archivos flag.txt, y como clave de descifrado se usó mi carné estudiantil (21299). Ambos valores se convirtieron a bytes, y se aplicó el algoritmo para obtener el texto plano resultante.

### FLAG ENCONTRADA:

**FLAG\_6d87f194aeb0dc5b75a150888bdf34fc**

### Buscar imagen

```

zoro@1308ca549687:~$ find /home/zoro/ONEPIECE/ -type f \! -iname "*.jpg" -o -iname "*.jpeg" -o -iname "*.png" -o -iname "*.zip" \) 2>/dev/null
/home/zoro/ONEPIECE/Pirate_Island/Pirates_Cove/Casa_de_GoLD_Roger/ponglyph.zip
zoro@1308ca549687:~$ 

```

se ejecutó un comando find desde la ruta base /home/zoro/ONEPIECE/ con el objetivo de localizar archivos potencialmente relevantes que estuvieran en formato de imagen (.jpg, .jpeg, .png) o archivos comprimidos (.zip). El comando utilizado fue:

```
find /home/zoro/ONEPIECE/ -type f \(-iname "*.jpg" -o -iname "*.jpeg" -o -iname "*.png" -o -iname "*.zip"\) 2>/dev/null
```

Este comando permitió listar todos los archivos con esas extensiones dentro de la estructura de carpetas. Como resultado, se identificó un archivo ZIP sospechoso ubicado en:

/home/zoro/ONEPIECE/Pirate\_Island/Pirates\_Cove/Casa\_de\_GoLD\_Roger/poneglyph.zip

### Extraer Imagen

```
zoro@1308ca549687:~$ cd /home/zoro/ONEPIECE/Pirate_Island/Pirates_Cove/Casa_de_GoLD_Roger
zoro@1308ca549687:~/ONEPIECE/Pirate_Island/Pirates_Cove/Casa_de_GoLD_Roger$ ls
poneglyph.zip
zoro@1308ca549687:~/ONEPIECE/Pirate_Island/Pirates_Cove/Casa_de_GoLD_Roger$ unzip poneglyph.zip -d /home/zoro/poneglyph_
extraido
Archive: poneglyph.zip
  skipping: poneglyph.jpeg      need PK compat. v6.3 (can do v4.6)
zoro@1308ca549687:~/ONEPIECE/Pirate_Island/Pirates_Cove/Casa_de_GoLD_Roger$ cd /home/zoro/poneglyph_extraido

zoro@1308ca549687:~$ 7z x "/home/zoro/ONEPIECE/Pirate_Island/Pirates_Cove/Casa_de_GoLD_Roger/poneglyph.zip" -o"/hom
o/poneglyph_extraido"
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,32 CPUs 13th Gen Intel(R) Core(TM) i9-13950HX (B0671),
ES-NI)

Scanning the drive for archives:
1 file, 53398 bytes (53 KiB)

Extracting archive: /home/zoro/ONEPIECE/Pirate_Island/Pirates_Cove/Casa_de_GoLD_Roger/poneglyph.zip
--
Path = /home/zoro/ONEPIECE/Pirate_Island/Pirates_Cove/Casa_de_GoLD_Roger/poneglyph.zip
Type = zip
Physical Size = 53398

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

Size:      53042
Compressed: 53398
zoro@1308ca549687:~$ cd /home/zoro/poneglyph_extraido
zoro@1308ca549687:~/poneglyph_extraido$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
^C
Keyboard interrupt received, exiting.
zoro@1308ca549687:~/poneglyph_extraido$ cd ..
```

```
zoro@1308ca549687:~/poneglyph_extraido$ cd ..
zoro@1308ca549687:~$ cd /home/zoro/poneglyph_extraido
zoro@1308ca549687:~/poneglyph_extraido$ ls
poneglyph.jpeg
```

Imagen encontrada:



```
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>docker cp zoro_challenge:/home/zoro/poneglyph_extraido/poneglyph.jpeg C:\Users\marce\OneDrive\Desktop
Successfully copied 54.8kB to C:\Users\marce\OneDrive\Desktop
```

Para acceder al contenido del archivo poneglyph.zip, utilicé el comando **7z x "/home/zoro/ONEPIECE/Pirate\_Island/Pirates\_Cove/Casa\_de\_GoID\_Roger/poneglyph.zip" -o"/home/zoro/poneglyph\_extraido"** para descomprimirlo en la ruta /home/zoro/poneglyph\_extraido, utilizando como contraseña la flag obtenida previamente. En lugar de exponer la imagen resultante a través de un servidor HTTP como en ocasiones anteriores, opté por copiarla directamente al escritorio de mi máquina anfitriona. Para ello, ejecuté el siguiente comando desde el host:

```
docker cp zoro_challenge:/home/zoro/poneglyph_extraido/poneglyph.jpeg C:\Users\marce\OneDrive\Desktop,
```

asegurándome previamente de haber asignado permisos de lectura adecuados al archivo desde dentro del contenedor.

### Extraer texto de la imagen

```
PS C:\Users\marce\OneDrive\Desktop\cifrados> & C:/Users/marce/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/marce/OneDrive/Desktop/cifrados/extract_text_from_image.py
Introduce tu carnet para descifrar el mensaje: 21299
b'After arriving at the Sea Forest, Robin read the lone Poneglyph there and'
PS C:\Users\marce\OneDrive\Desktop\cifrados>
```

Para descifrar el mensaje oculto en la imagen poneglyph.jpeg, utilicé el script extract\_text\_from\_image.py ubicado en la carpeta utils. Este script hace uso de la biblioteca PIL y piexif para extraer los metadatos EXIF de la imagen, específicamente el campo Artist, donde se encontraba almacenado el mensaje cifrado.

Posteriormente, se utilizó la función xor\_cipher (importada desde el módulo luffy\_xor) para descifrar dicho mensaje, utilizando como clave mi carné estudiantil.

Se obtuvo el siguiente texto descifrado:

#### **Texto descifrado**

*After arriving at the Sea Forest, Robin read the lone Poneglyph there and*

### ► Usopp - Stream Cipher Custom

Descifra un mensaje oculto en archivos de texto usando XOR con claves derivadas del carné.

#### ¿Qué es un cifrado de flujo?

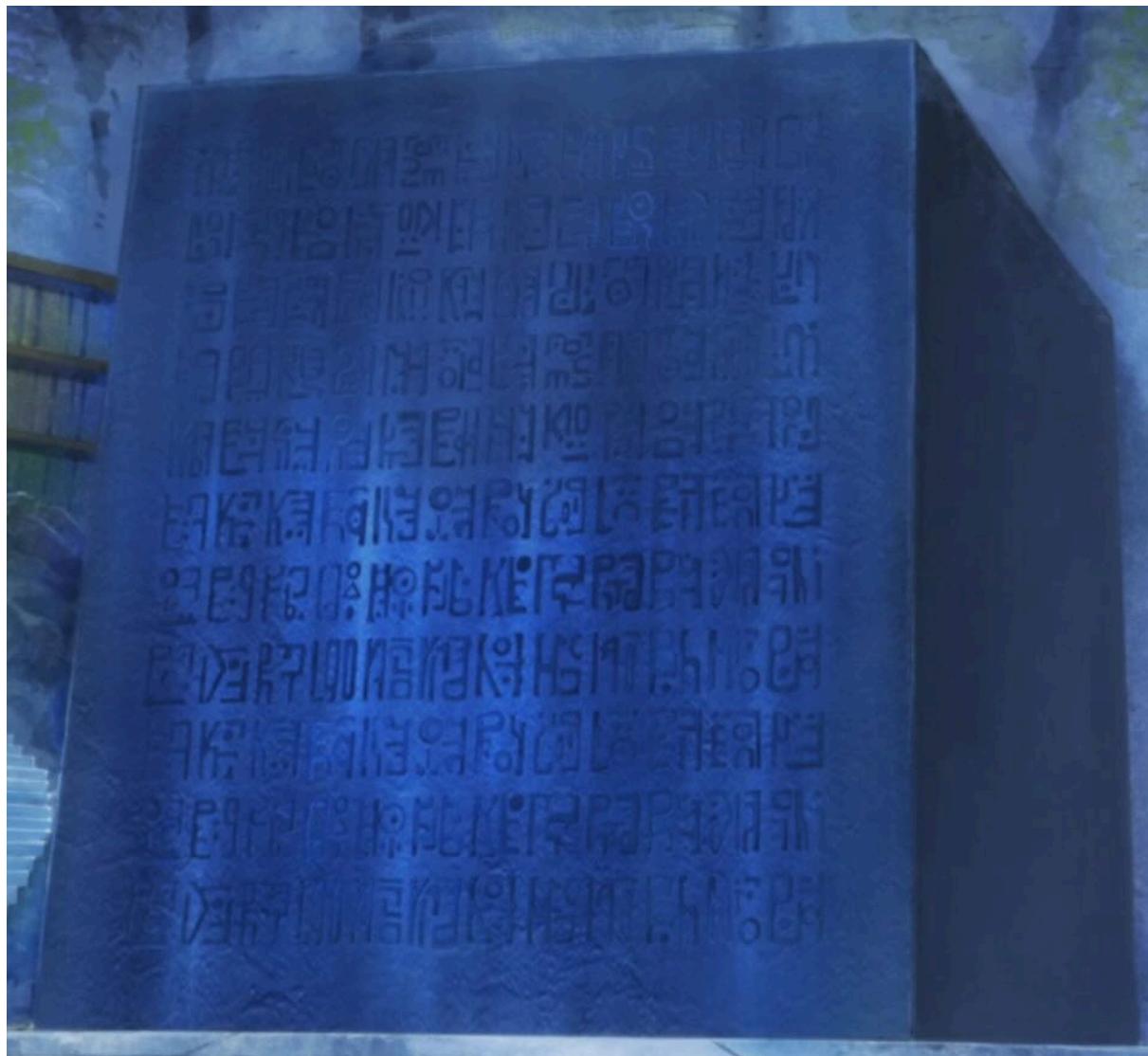
Los cifrados de flujo generan una secuencia pseudoaleatoria de bits (keystream) que se combina con el texto original mediante una operación XOR.

#### Debilidades de los cifrados de flujo personalizados

- Si la clave se reutiliza, es posible extraer información del texto original.
- Si el generador de claves es predecible, se puede reconstruir la secuencia.
- La seguridad depende de la calidad del PRNG utilizado.

#### Objetivo del reto

Analizar la implementación de un cifrado de flujo personalizado y encontrar fallos en la generación de claves para descifrar el mensaje.



**FLAG ENCONTRADA:** *FLAG\_daf9645e48e3ccab692fafc0c551ed75*

**TEXTO EXTRAÍDO:**

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

**IMAGEN ENCONTRADA:**



## Acceder al contenedor

```
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>docker exec -it usopp_challenge bash
ls /home/usopp
nobody@af02485f82cf:/home/usopp$ ls /home/usopp
ls: cannot open directory '/home/usopp': Permission denied
nobody@af02485f82cf:/home/usopp$ su usopp
Password:
```

Para iniciar el análisis del reto Usopp - Stream Cipher Custom, se accedió de forma interactiva al contenedor Docker correspondiente mediante el comando:

***docker exec -it usopp\_challenge bash***

Al ingresar, el usuario por defecto no tenía permisos suficientes para explorar el directorio /home/usopp, como se evidenció al intentar listar su contenido. Para solucionar esto, se procedió a escalar privilegios utilizando el comando su usopp, el cual solicita la contraseña (flag obtenida en un reto anterior).

## Exploración y lectura de archivos flag.txt en el contenedor

```
usopp@af02485f82cf:~$ find /home/usopp/ONEPIECE/ -iname "*flag*" \
    -exec echo "===== {} =====" \; \
    -exec cat {} \; \
    -exec echo "" \;
=====
===== /home/usopp/ONEPIECE/East_Blue/Shells_Town/Casa_de_Morgan/flag.txt =====
Has encontrado un lugar lleno de secretos y misterios que apuntan a usopp
===== /home/usopp/ONEPIECE/East_Blue/Orange_Town/Casa_de_Boodle/flag.txt =====
Has encontrado un amigo y han decidido viajar juntos
===== /home/usopp/ONEPIECE/Water_7/GalleyLa_Headquarters/Casa_de_Lucci/flag.txt =====
a77742694e4e53d3403e3b3dd4c3d5291d3e59669b4918dc454c71e910f9c380b91a02df3c
===== /home/usopp/ONEPIECE/Water_7/Carpenters_Cafe/Casa_de_Icebburg/flag.txt =====
Has encontrado un enemigo y ha tenido que luchar
===== /home/usopp/ONEPIECE/Fishman_Island/Coral_Mansion/Casa_de_Shirahoshi/flag.txt =====
Has encontrado un lugar peligroso
===== /home/usopp/ONEPIECE/Fishman_Island/Gyoncorde_Plaza/Casa_de_Otohime/flag.txt =====
Has encontrado un obstáculo y hay que superarlo
===== /home/usopp/ONEPIECE/Fishman_Island/Gyoverly_Hills/Casa_de_Otohime/flag.txt =====
Has encontrado un lugar misterioso
===== /home/usopp/ONEPIECE/Sabaody_Archipelago/Grove_109/Casa_de_Shakky/flag.txt =====
Te cansaste de buscar y te has quedado dormido
===== /home/usopp/ONEPIECE/Sabaody_Archipelago/Grove_109/Casa_de_Rayleigh/flag.txt =====
Has encontrado un enemigo y ha tenido que luchar
===== /home/usopp/ONEPIECE/Sabaody_Archipelago/Grove_41/Casa_de_Rayleigh/flag.txt =====
Has encontrado una pista que apunta a usopp
===== /home/usopp/ONEPIECE/Sabaody_Archipelago/Grove_80/Casa_de_Keimi/flag.txt =====
Has encontrado un lugar abandonado
usopp@af02485f82cf:~$ |
```

Para localizar y leer todos los archivos cuyo nombre contuviera la palabra "flag" en cualquier parte del contenedor, se utilizo el siguiente comando :**find /home/usopp/ONEPIECE/ -iname "\*flag\*" \**

```
-exec echo "===== {} =====" \; \
-exec cat {} \; \
-exec echo "" \;
```

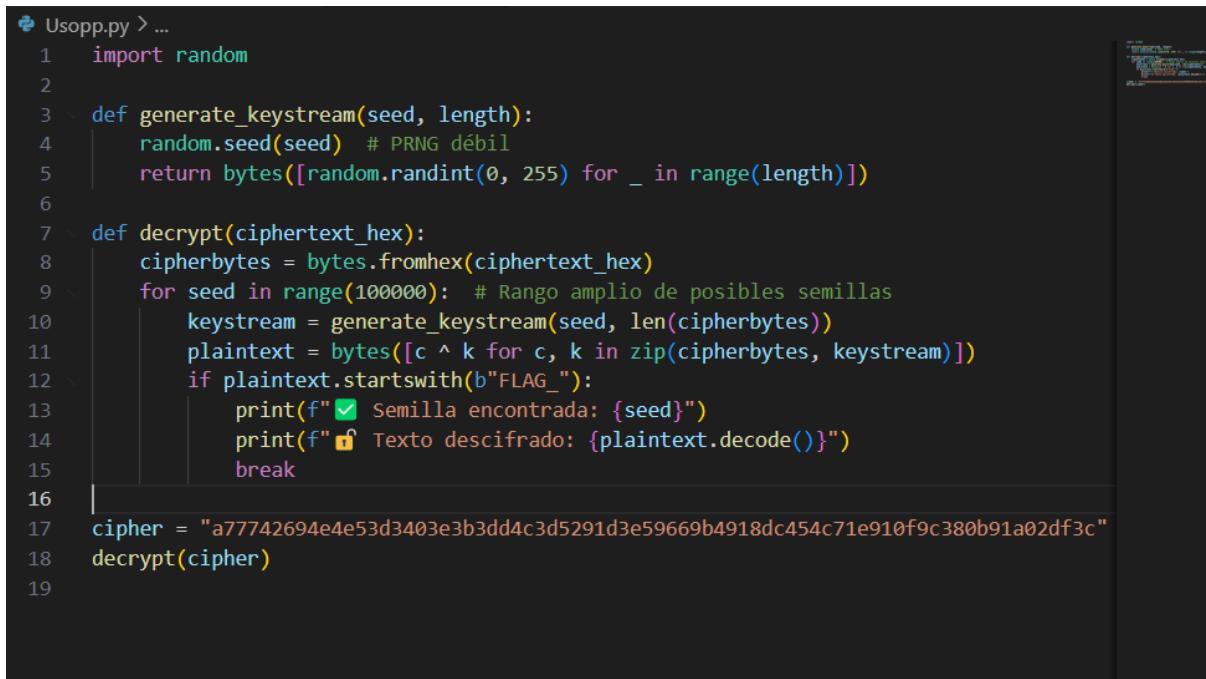
Este comando realiza una búsqueda recursiva desde la carpeta /home/usopp/ONEPIECE/, identificando todos los archivos cuyo nombre contenga la palabra "flag" (independientemente de mayúsculas o minúsculas)

Como se puede observar, se identificó una cadena hexadecimal que destaca por su posible relevancia criptográfica:

**a77742694e4e53d3403e3b3dd4c3d5291d3e59669b4918dc454c71e910f9c380b91a02df3c**

Esta cadena se identificó en el archivo flag.txt ubicado en la ruta:/home/usopp/ONEPIECE/Water\_7/GalleyLa\_Headquarters/Casa\_de\_Lucci/flag.txt

### Recuperación de la FLAG mediante Fuerza Bruta al Generador de Claves



```
Usopp.py > ...
1  import random
2
3  def generate_keystream(seed, length):
4      random.seed(seed) # PRNG débil
5      return bytes([random.randint(0, 255) for _ in range(length)])
6
7  def decrypt(ciphertext_hex):
8      cipherbytes = bytes.fromhex(ciphertext_hex)
9      for seed in range(100000): # Rango amplio de posibles semillas
10         keystream = generate_keystream(seed, len(cipherbytes))
11         plaintext = bytes([c ^ k for c, k in zip(cipherbytes, keystream)])
12         if plaintext.startswith(b"FLAG_"):
13             print(f"✓ Semilla encontrada: {seed}")
14             print(f"💡 Texto descifrado: {plaintext.decode()}")
15             break
16
17 cipher = "a77742694e4e53d3403e3b3dd4c3d5291d3e59669b4918dc454c71e910f9c380b91a02df3c"
18 decrypt(cipher)
19
```

```
✓ Semilla encontrada: 1234
💡 Texto descifrado: FLAG_daf9645e48e3ccab692fafc0c551ed75
PS C:\Users\marce\OneDrive\Desktop\cifrados> 
```

Para resolver este reto, se utilizó el archivo `usopp.py`, ubicado en la carpeta `scripts`, el cual fue construido a partir del código base proporcionado en el archivo `usopp_cipher.py` de la carpeta `utils`.

El script tenía como objetivo descifrar un texto cifrado en formato hexadecimal, el cual utiliza un cifrado de flujo personalizado basado en la operación XOR. La clave para este cifrado es generada por un PRNG débil (mediante la función `random.seed(seed)`), y el desafío consistía en encontrar la semilla correcta que permitiera recuperar el mensaje original.

El script implementa un ataque de fuerza bruta, probando todas las semillas posibles desde 0 hasta 99999, hasta encontrar una que produzca una salida legible que comience con el prefijo "FLAG\_".

El texto cifrado fue:

`a77742694e4e53d3403e3b3dd4c3d5291d3e59669b4918dc454c71e910f9c380b91a02df3c`

La semilla correcta encontrada fue: 1234

El mensaje descifrado fue:

**FLAG ENCONTRADA:**

**`FLAG_daf9645e48e3ccab692fafc0c551ed75`**

[Buscar Imagen](#)

```
usopp@af02485f82cf:~$ find /home/usopp -iname "*.zip"
/home/usopp/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_GoLD_Roger/poneglyph.zip
usopp@af02485f82cf:~$
```

Para continuar con el proceso de extracción de la imagen cifrada, fue necesario ubicar el archivo .zip que la contenía dentro del sistema de archivos del contenedor usopp\_challenge. Para ello, se ejecutó el siguiente comando desde la terminal del contenedor:

```
find /home/usopp -iname "*.zip"
```

Este comando busca de forma recursiva todos los archivos con extensión .zip, sin distinguir entre mayúsculas y minúsculas, dentro del directorio /home/usopp.

Como resultado, se encontró el archivo:

```
/home/usopp/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_GoLD_Roger/poneglyph.zip
```

### Extracción de imagen desde archivo ZIP protegido

```
root@af02485f82cf:/home/usopp# find /home/usopp/ONEPIECE -iname "*.zip"
/home/usopp/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_GoLD_Roger/poneglyph.zip
root@af02485f82cf:/home/usopp# 7z x /home/usopp/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_GoLD_Roger/poneglyph.zip
-p-o"/home/usopp/poneglyph_extraido"

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,32 CPUs 13th Gen Intel(R) Core(TM) i9-13950HX (B0671),ASM,AES-NI)

Scanning the drive for archives:
1 file, 64478 bytes (63 KiB)

Extracting archive: /home/usopp/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_GoLD_Roger/poneglyph.zip
--
Path = /home/usopp/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_GoLD_Roger/poneglyph.zip
Type = zip
Physical Size = 64478

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

Size: 64643
Compressed: 64478
root@af02485f82cf:/home/usopp#
```

Una vez identificado el archivo poneglyph.zip, se procedió a extraer su contenido utilizando la herramienta 7-Zip (7z). Para ello, se utilizó el siguiente comando dentro del contenedor:

```
7z x
/home/usopp/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_GoLD_Roger/poneglyph.zip
-p-o"/home/usopp/poneglyph_extraido"
```

Este comando realiza lo siguiente:

- x: extrae el contenido del archivo.
- Se especifica la ruta absoluta del .zip.
- La opción -o define el directorio de destino para los archivos extraídos (/home/usopp/poneglyph\_extraido).

Durante la extracción, se solicitó una contraseña, fue ingresa la flag encontrada anteriormente.

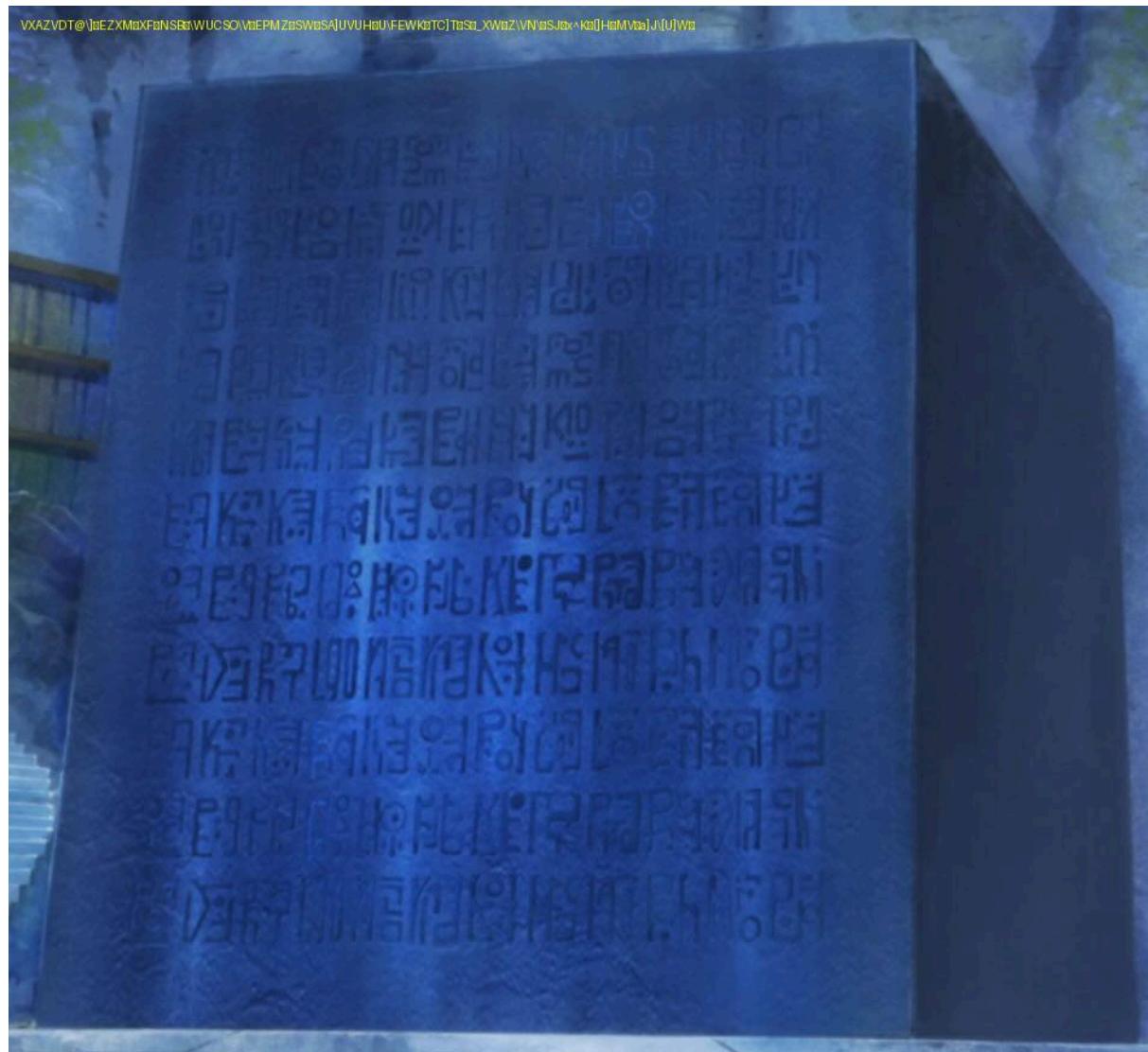
#### **Copia del archivo *poneglyph.jpeg* desde el contenedor hacia la máquina anfitriona**

```
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>docker cp usopp_challenge:/home/usopp/poneglyph_extraido/poneglyph.jpeg C:\Users\marce\OneDrive\Desktop\  
Successfully copied 66.6kB to C:\Users\marce\OneDrive\Desktop\
```

Una vez extraída la imagen *poneglyph.jpeg* dentro del contenedor *usopp\_challenge*, el siguiente paso fue transferirla a mi computadora local para su posterior análisis. Para ello, utilicé el comando *docker cp*, que permite copiar archivos entre el contenedor y la máquina anfitriona.

```
docker cp usopp_challenge:/home/usopp/poneglyph_extraido/poneglyph.jpeg  
C:\Users\marce\OneDrive\Desktop\
```

#### **Imagen Encontrada**



**Extracción y descifrado del texto oculto en la imagen *poneglyph.jpeg***

```

❷ extract_text_from_image.py > ...
1  from PIL import Image
2  import piexif
3  from luffy_xor import xor_cipher
4
5  def extraer_texto_metadata(imagen_path):
6      # Abrir la imagen
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     if texto:
15         return texto.decode('utf-8')
16     return None
17
18
19     # Uso del código para descifrar la imagen
20     # Ejemplo de uso
21     image_path = "ussop.jpeg"
22     student_id = input("Introduce tu carné para descifrar el mensaje: ")
23     texto_cifrado = extraer_texto_metadata(image_path)
24     decrypted_text = xor_cipher(texto_cifrado, student_id)
25     print(decrypted_text)

```

```

Introduce tu carné para descifrar el mensaje: 21299
b'discovered that it was engraved with an apology letter from a man known as Joy Boy to Poseidon.'
PS C:\Users\marce\OneDrive\Desktop\cifrados>

```

Para este paso se utilizó el script `extract_text_from_image.py` ubicado en la carpeta `utils`, el cual hace uso de la librería `piexif` para extraer los metadatos EXIF de una imagen. Específicamente, se accede al campo `Artist`, donde estaba oculto un mensaje cifrado. Luego, dicho texto fue descifrado mediante el uso del algoritmo XOR implementado en el módulo `luffy_xor.py`, utilizando como clave mi número de carné.

El mensaje descifrado obtenido fue:

### **Mensaje obtenido**

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

### **► Nami - ChaCha20 Playground**

Experimenta con el cifrado ChaCha20 y observa el impacto del nonce y la clave.

### **¿Qué es ChaCha20?**

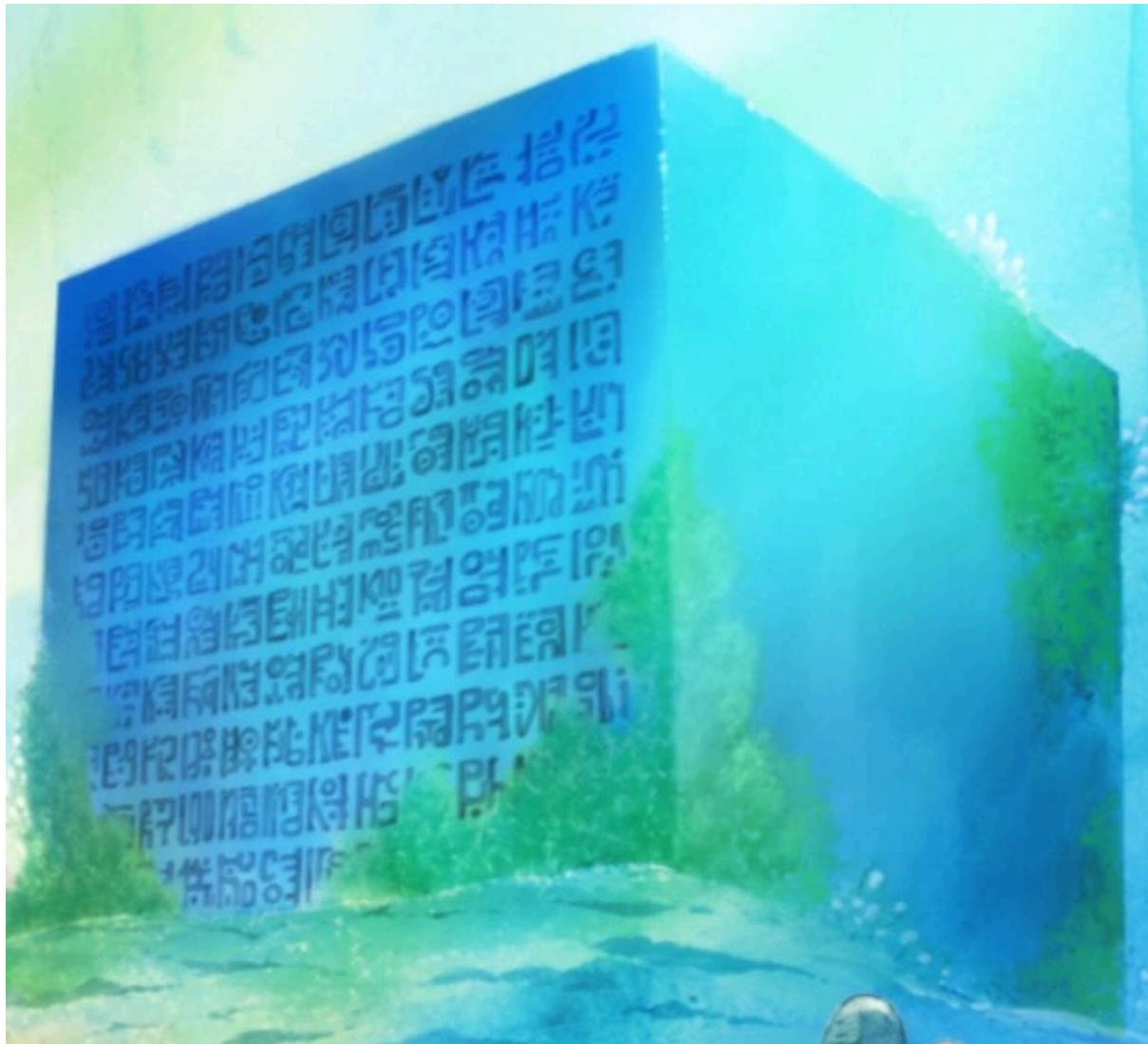
ChaCha20 es un cifrado de flujo moderno diseñado para ser seguro y eficiente, usado en protocolos como TLS y WireGuard.

## Importancia del Nonce

- El nonce debe ser único para cada mensaje cifrado.
- Si se reutiliza un nonce con la misma clave, se pueden realizar ataques criptográficos.

## Objetivo del reto

Experimentar con ChaCha20 cifrando y descifrando mensajes, analizando el impacto del nonce y la clave.

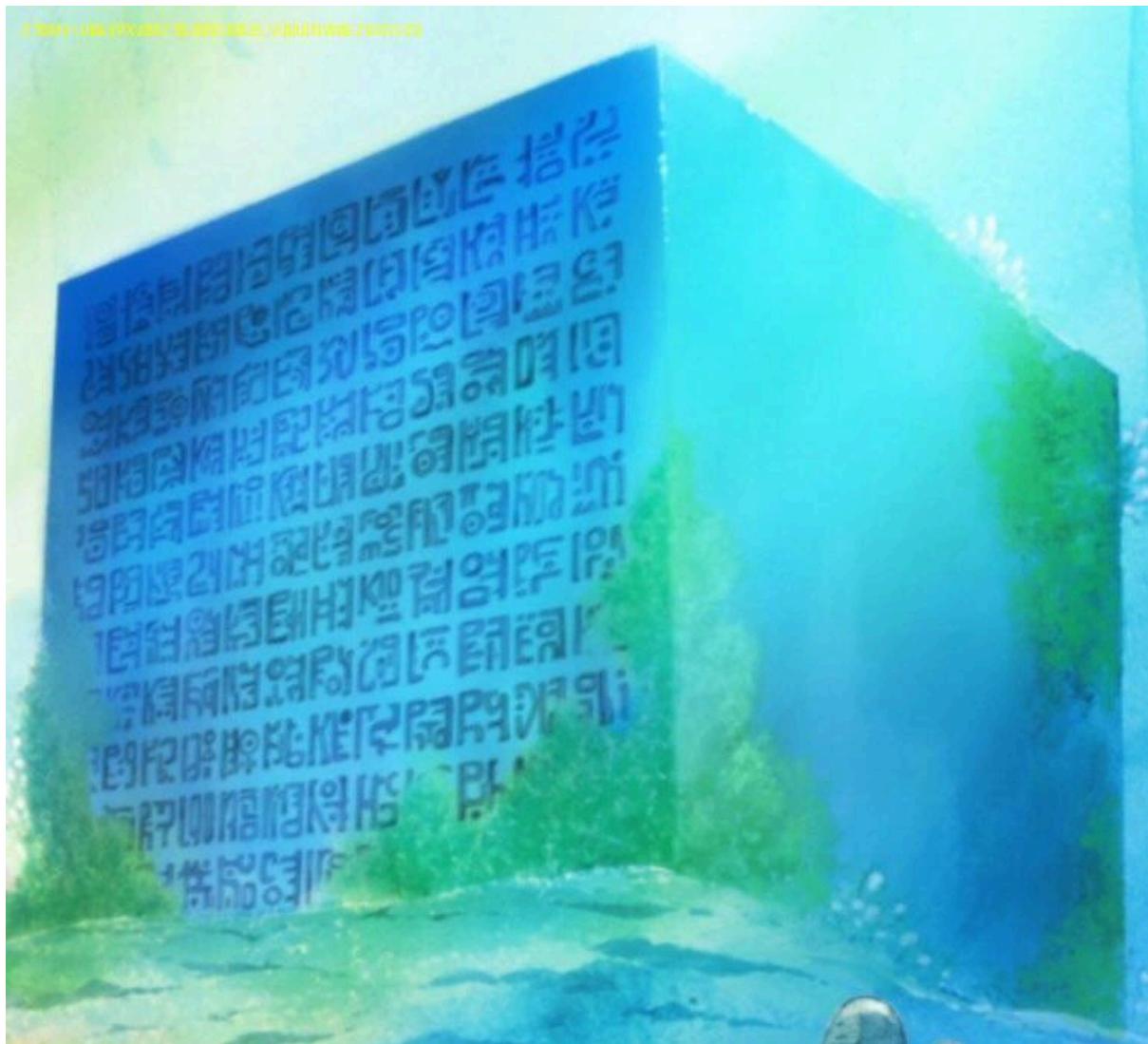


**FLAG ENCONTRADA:** FLAG\_6c1f8h3f78dba385c8eb17f841e49be

**TEXTO EXTRAÍDO:**

b'he told Robin the story handed down through '

**IMAGEN ENCONTRADA:**



### Acceso al contenedor nami\_challenge con Docker

```
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>docker exec -it nami_challenge bash
Error response from daemon: container f8ed5c802c33914c642d03e6c3ba19945f640f678e075d5364e3eff184e9879 is not running

C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>docker start nami_challenge
nami_challenge

C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>docker exec -it nami_challenge bash
nobody@f8ed5c802c33:/home/nami$
```

Para iniciar el análisis del reto asociado al cifrado ChaCha20, fue necesario acceder al contenedor Docker llamado nami\_challenge .Se utilizó el comando docker start nami\_challenge para iniciarla manualmente. Posteriormente, se logró acceder de forma interactiva al contenedor utilizando el comando:docker exec -it nami\_challenge bash.

#### Acceder al usuario

```
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

nami@f8ed5c802c33:~$
```

All ingresar se accedió como un usuario sin privilegios (nobody). Para obtener los permisos necesarios, se procedió a cambiar al usuario adecuado dentro del contenedor utilizando: su nami , como contraseña se ingresó la flag encontrada anteriormente.

### **Buscar Flag: Encontrar y leer archivos .flag**

```
nami@f8ed5c802c33:~$ find /home/nami/ONEPIECE/ -iname "*flag*" \
-exec echo "===== {} =====" \; \
-exec cat {} \; \
-exec echo "" \;
===== /home/nami/ONEPIECE/East_Blue/Shells_Town/Casa_de_Rika/flag.txt =====
Has encontrado un objeto misterioso y no sabes qu* es
===== /home/nami/ONEPIECE/East_Blue/Orange_Town/Casa_de_Boodle/flag.txt =====
Has encontrado un tesoro que apunta a nami
===== /home/nami/ONEPIECE/Water_7/Carpenters_Cafe/Casa_de_Paulie/flag.txt =====
Has encontrado un enemigo y ha tenido que huir
===== /home/nami/ONEPIECE/Water_7/Blue_Station/Casa_de_Paulie/flag.txt =====
Has encontrado un mapa que apunta a nami
===== /home/nami/ONEPIECE/Fishman_Island/Gyoncorde_Plaza/Casa_de_Otohime/flag.txt =====
Has encontrado un mapa que apunta a nami
===== /home/nami/ONEPIECE/Fishman_Island/Mermaid_Cove/Casa_de_Otohime/flag.txt =====
Has encontrado una pista que apunta a nami
===== /home/nami/ONEPIECE/Sabaody_Archipelago/Grove_80/Casa_de_Keimi/flag.txt =====
765d6c96519b340580f1439e3549fe151c33ef4a226a61f51b9265636ea7f64d009a6a0b40
===== /home/nami/ONEPIECE/Sabaody_Archipelago/Grove_80/Casa_de_Rayleigh/flag.txt =====
Has encontrado una pista que apunta a nami
===== /home/nami/ONEPIECE/Sabaody_Archipelago/Grove_66/Casa_de_Rayleigh/flag.txt =====
Has encontrado un lugar lleno de secretos y misterios que apuntan a nami
===== /home/nami/ONEPIECE/Pirate_Island/Pirates_Hideout/Casa_de_GoLD_Roger/flag.txt =====
Has encontrado un lugar misterioso
nami@f8ed5c802c33:~$
```

Para localizar y visualizar posibles pistas o mensajes relacionados con el reto ChaCha20, se ejecutó un comando que recorre recursivamente la carpeta /home/nami/ONEPIECE/ en busca de archivos cuyo nombre contenga la palabra "flag". Esto permitió listar y leer automáticamente su contenido.

### **Descifrado del mensaje con ChaCha20**

```

1  from Crypto.Cipher import ChaCha20
2
3  def generate_key_nonce(user_id):
4      key = (user_id.encode() * 32)[:32]
5      nonce = (user_id.encode() * 8)[:8]
6      return key, nonce
7
8  def chacha20_decrypt(ciphertext_hex, user_id):
9      ciphertext = bytes.fromhex(ciphertext_hex)
10     key, nonce = generate_key_nonce(user_id)
11     cipher = ChaCha20.new(key=key, nonce=nonce)
12     plaintext = cipher.decrypt(ciphertext)
13     return plaintext.decode(errors='ignore')
14
15 # Cadena obtenida del archivo .flag
16 cipher_hex = "765d6c96519b340580f1493e3549fe151c33ef4a226a61f51b9265636ea7f64d009a6a0b40"
17
18 # Reemplazá con tu número de carné
19 user_id = "21299"
20
21 # Desencriptar
22 resultado = chacha20_decrypt(cipher_hex, user_id)
23 print(["Mensaje descifrado:", resultado])
24

```

```

PS C:\Users\marce> & C:/Users/marce/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/marce/OneDrive/Desktop/nana.py
Mensaje descifrado: FLAG_6cf18h3f78dba385c0eb17f8b14e9be
PS C:\Users\marce>

```

Para completar el reto relacionado con el cifrado ChaCha20, se utilizó un script Python llamado nana.py, que implementa el algoritmo de descifrado usando la librería Crypto.Cipher.ChaCha20 de pycryptodome.

El objetivo fue descifrar una cadena hexadecimal encontrada previamente en un archivo .flag, utilizando como clave y nonce un valor derivado del número de carné del estudiante. Para ello, el script genera una clave de 256 bits y un nonce de 64 bits repitiendo el carné las veces necesarias y truncando al tamaño requerido.

El resultado del descifrado fue: **FLAG\_6c1f8h3f78dba385c8eb17f841e49be**

### **FLAG ENCONTRADA:**

**FLAG\_6c1f8h3f78dba385c8eb17f841e49be**

### **Encontrando la imagen**

#### **Buscar .zip**

```

nami@f8ed5c802c33:~$ find /home/nami/ONEPIECE/ -iname "*.zip"
/home/nami/ONEPIECE/Water_7/GalleyLa_Headquarters/Casa_de_Paulie/poneglyph.zip
nami@f8ed5c802c33:~$
```

Para iniciar el análisis del reto asociado al cifrado ChaCha20, se procedió a buscar archivos comprimidos .zip dentro del sistema de archivos del contenedor nami\_challenge. Esto con el fin de localizar posibles archivos que contuvieran imágenes o datos cifrados relevantes para el desafío.

Se utilizó el siguiente comando:

```
find /home/nami/ONEPIECE/ -iname "*.zip"
```

Este comando recorre recursivamente el directorio /home/nami/ONEPIECE/ y lista todos los archivos que terminen en .zip, sin importar si están en mayúsculas o minúsculas gracias al flag -iname.

Como resultado, se localizó el archivo:

/home/nami/ONEPIECE/Water\_7/GalleyLa\_Headquarters/Casa\_de\_Paulie/poneglyph.zip

### Extracción del archivo contenido en el ZIP

```
root@f8ed5c802c33:/home/nami# 7z x "/home/nami/ONEPIECE/Water_7/GalleyLa_Headquarters/Casa_de_Paulie/poneglyph.zip" -o"/home/nami/poneglyph_extraido"
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,32 CPUs 13th Gen Intel(R) Core(TM) i9-13950HX (B0671),ASM,AE-S-NI)

Scanning the drive for archives:
1 file, 51981 bytes (51 KiB)

Extracting archive: /home/nami/ONEPIECE/Water_7/GalleyLa_Headquarters/Casa_de_Paulie/poneglyph.zip

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

Size:      52030
Compressed: 51981
root@f8ed5c802c33:/home/nami#
```

Para continuar con el reto asociado al cifrado ChaCha20, se procedió a buscar y extraer un archivo comprimido .zip que contenía una imagen cifrada. Para ello, se utilizó el siguiente comando dentro del contenedor ya accedido como usuario root:

```
7z x
"/home/nami/ONEPIECE/Water_7/GalleyLa_Headquarters/Casa_de_Paulie/poneglyph.zip"
-o"/home/nami/poneglyph_extraido"
```

Al ejecutar el comando, el sistema solicitó una contraseña para realizar la extracción. Se utilizó como contraseña la **flag obtenida en el reto anterior**, que resultó ser válida para desencriptar el contenido del archivo.

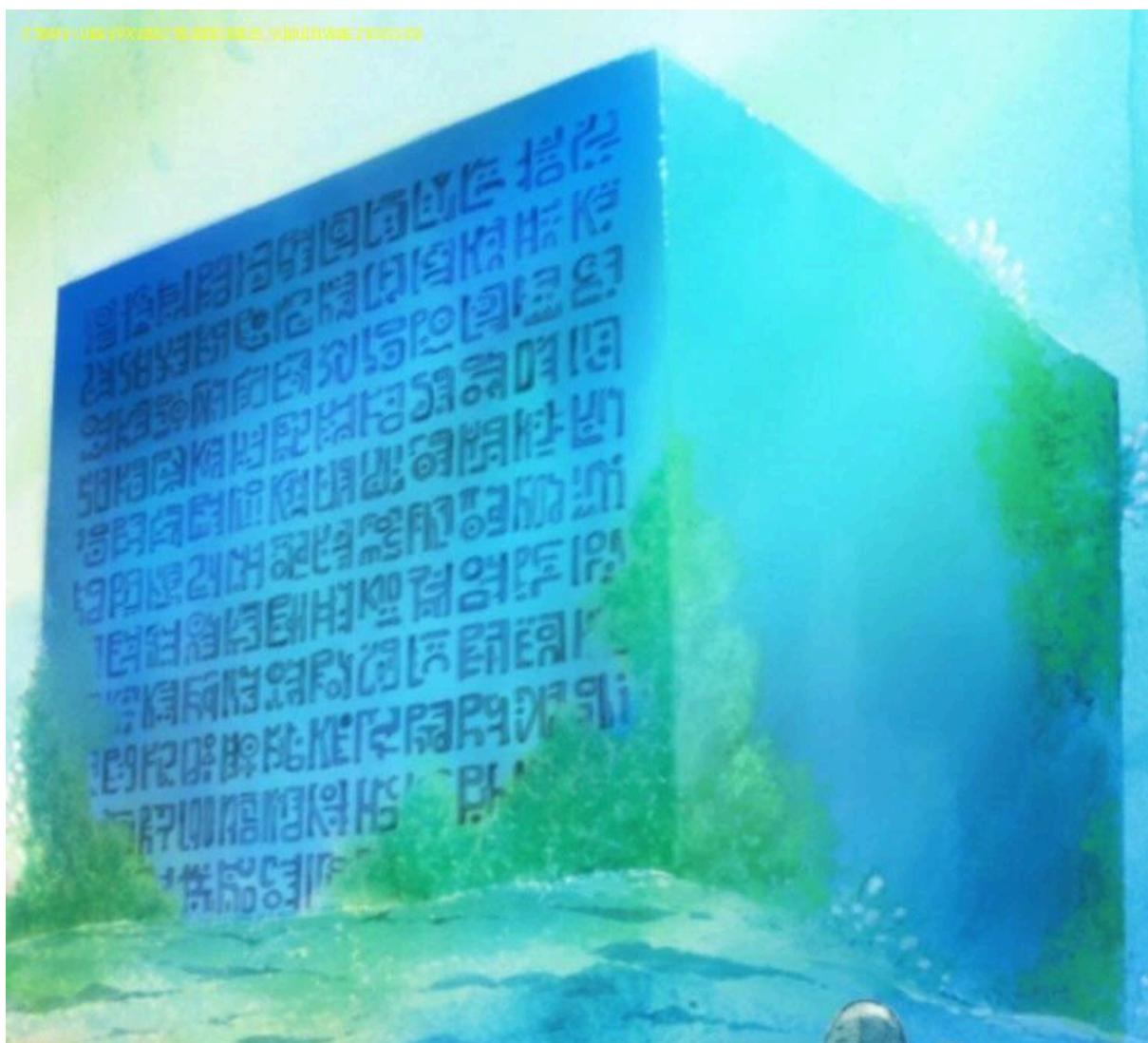
### Copiar a computadora anfitriona

```
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>docker cp nami_challenge:/home/nami/poneglyph_extraido/poneglyph.jpeg C:\Users\marce\OneDrive\Desktop\
Successfully copied 53.8kB to C:\Users\marce\OneDrive\Desktop\
C:\Users\marce\OneDrive\Documents\GitHub\PROYECTO1CIFRADOS>
```

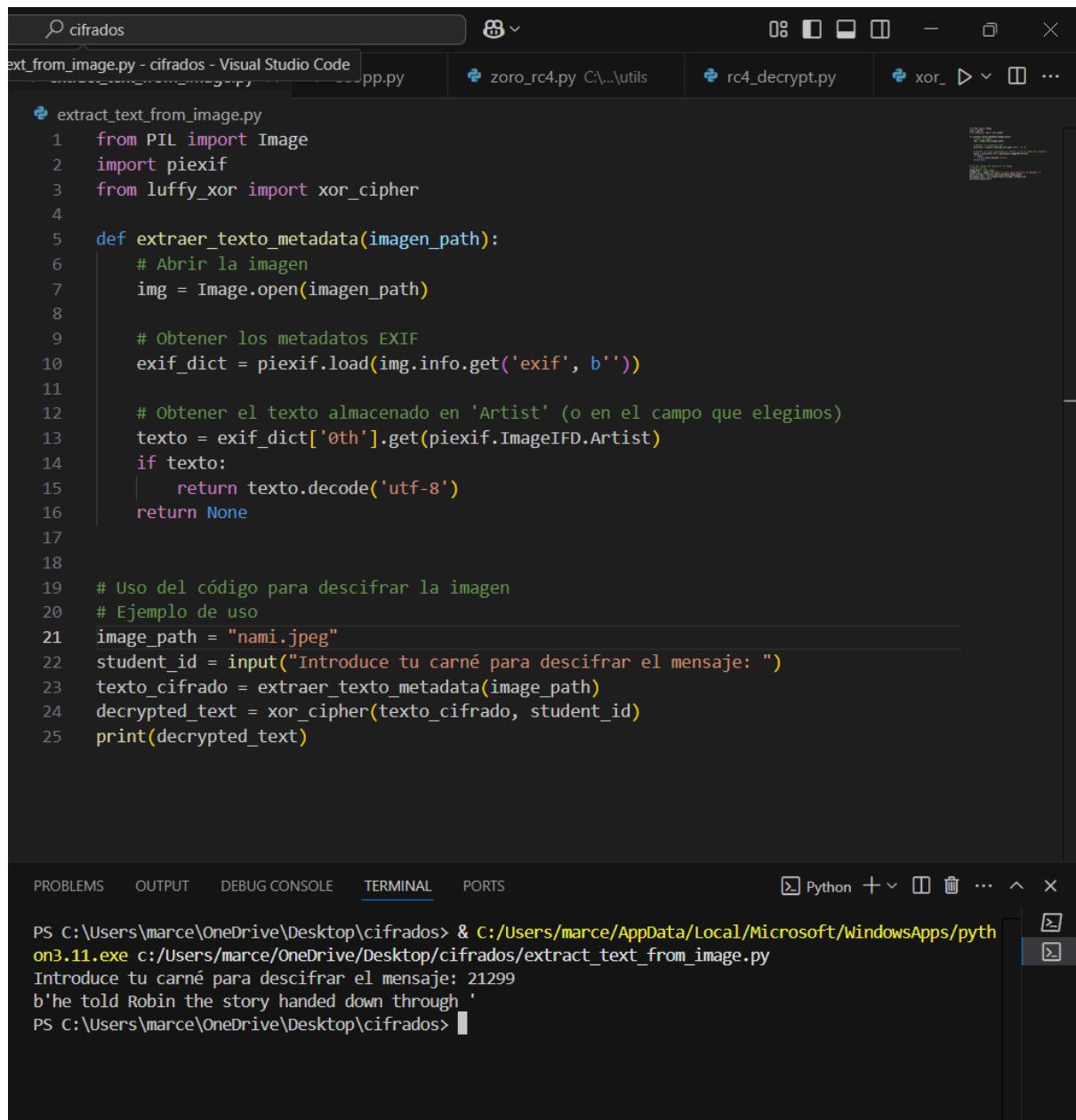
Una vez que se extrajo la imagen poneglyph.jpeg dentro del contenedor Docker del reto asociado al cifrado ChaCha20, fue necesario transferirla hacia la máquina local para su análisis. Para ello, se utilizó el siguiente comando desde la terminal en el sistema anfitrión (Windows):  
docker cp nami\_challenge:/home/nami/poneglyph\_extraido/poneglyph.jpeg C:\Users\marce\OneDrive\Desktop\

Este comando copia el archivo ubicado dentro del contenedor nami\_challenge hacia la ruta local C:\Users\marce\OneDrive\Desktop\

**IMAGEN ENCONTRADA:**



**Extracción y Descifrado de Texto desde la Imagen**



The screenshot shows a Visual Studio Code interface. The top bar has a search field containing 'cifrados'. Below the search bar, there are tabs for 'extract\_text\_from\_image.py - cifrados - Visual Studio Code', 'pp.py', 'zoro\_rc4.py C:\...\utils', 'rc4\_decrypt.py', and 'xor\_'. The main editor area contains the following Python code:

```

1  from PIL import Image
2  import piexif
3  from luffy_xor import xor_cipher
4
5  def extraer_texto_metadata(imagen_path):
6      # Abrir la imagen
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     if texto:
15         return texto.decode('utf-8')
16     return None
17
18
19     # Uso del código para descifrar la imagen
20     # Ejemplo de uso
21     image_path = "nami.jpeg"
22     student_id = input("Introduce tu carné para descifrar el mensaje: ")
23     texto_cifrado = extraer_texto_metadata(image_path)
24     decrypted_text = xor_cipher(texto_cifrado, student_id)
25     print(decrypted_text)

```

The bottom part of the interface shows a terminal window with the following output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\marce\OneDrive\Desktop\cifrados> & C:/Users/marce/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/marce/OneDrive/Desktop/cifrados/extract_text_from_image.py
Introduce tu carné para descifrar el mensaje: 21299
b'he told Robin the story handed down through '
PS C:\Users\marce\OneDrive\Desktop\cifrados>

```

Para finalizar el análisis del reto correspondiente al cifrado XOR, se utilizó la imagen nami.jpeg ubicada en la carpeta imágenes , previamente extraída desde el contenedor. El objetivo fue recuperar un mensaje oculto dentro de sus metadatos EXIF.

Para ello, se empleó el script extract\_text\_from\_image.py, ubicado en la carpeta utils. Este script hace uso de la librería piexif para acceder al campo Artist de los metadatos EXIF, donde se encontraba almacenado el texto cifrado. Posteriormente, el mensaje fue descifrado utilizando la función xor\_cipher importada desde el archivo luffy\_xor.py, también ubicado en utils.

Se utilizó como entrada mi número de carné 21299 para realizar el descifrado.

## TEXTO DESCIFRADO:

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

## Reflexión Final

Este proyecto permitió aplicar técnicas fundamentales de cifrado y descifrado de mensajes, explorando de manera práctica la seguridad criptográfica en entornos simulados. A lo largo de los cuatro retos se abordaron distintos algoritmos de cifrado de flujo, cada uno con sus propias características, ventajas y vulnerabilidades:

En el reto de Usopp, se utilizó un cifrado XOR con una secuencia generada por un PRNG débil. Esta debilidad permitió aplicar un ataque por fuerza bruta para encontrar la semilla, lo que evidenció cómo una implementación deficiente puede comprometer completamente un sistema.

En el reto de Zoro, se trabajó con el algoritmo RC4, logrando descifrar una cadena hexadecimal extraída desde metadatos utilizando una clave derivada del carné. Esta experiencia demostró la importancia de una correcta gestión de claves, y dejó claro por qué algoritmos antiguos como RC4 han sido descartados en contextos modernos debido a sus vulnerabilidades conocidas.

En el reto de Nami, se experimentó con ChaCha20, un algoritmo de cifrado de flujo moderno y seguro. La clave y el nonce fueron derivados de mi carné estudiantil, y se observó cómo la reutilización del nonce puede comprometer gravemente la seguridad, permitiendo ataques similares al XOR. Este caso sirvió para comprender el papel crítico del nonce en la criptografía contemporánea.

Finalmente, en el reto de Luffy, se trabajó con el descifrado de texto oculto en imágenes mediante metadatos, aplicando un algoritmo XOR simple. Esto permitió aplicar técnicas de manipulación de imágenes utilizando bibliotecas como Pillow y piexif, y reflexionar sobre conceptos de esteganografía y canales de comunicación encubierta.

A lo largo del proyecto, se evidenció cómo una mala práctica, como reutilizar claves o emplear generadores de números aleatorios inseguros, puede comprometer por completo la seguridad de los datos. Además de fortalecer conocimientos técnicos en criptografía, este proceso mostró que la implementación debe cuidarse con tanto rigor como los conceptos teóricos.

Este proyecto fue una muestra clara de cómo incluso los detalles más pequeños—como una semilla, un nonce o una mala decisión en el código—pueden marcar la diferencia entre un sistema protegido y uno completamente vulnerable.