

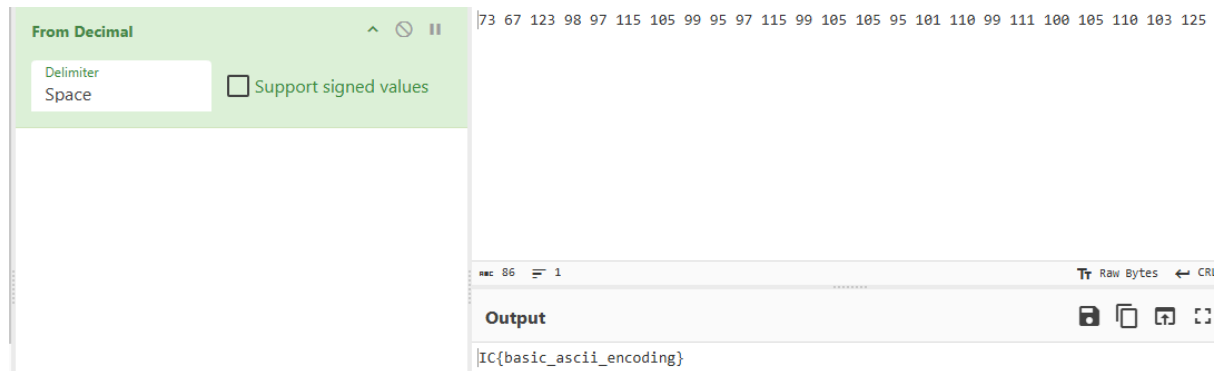
IC- Práctica 1 (PARTE 1) Protección / Ocultamiento de información

Ejercicio 01

Develar el mensaje que se intentó ocultar utilizando un sistema de codificación

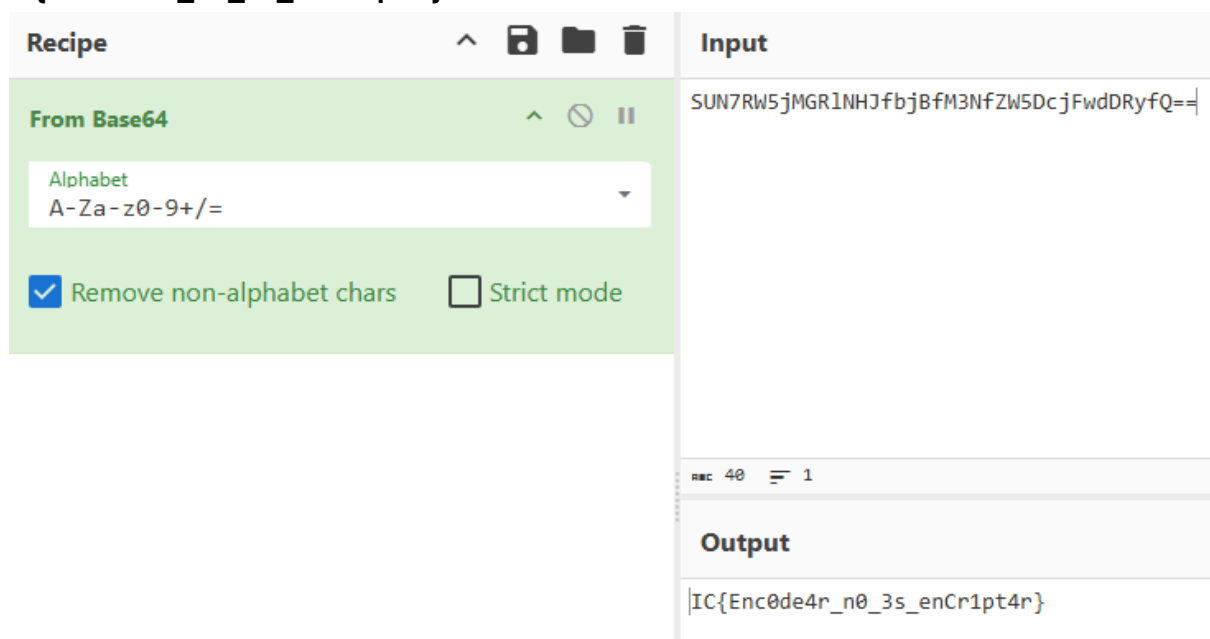
a. 73 67 123 98 97 115 105 99 95 97 115 99 105 105 95 101 110 99 111 100 105 110 103 125

IC{basic_ascii_encoding}



b. SUN7RW5jMGR1NHJfbjBfM3NfZW5DcjFwdDRyfQ==

IC{Enc0de4r_n0_3s_enCr1pt4r}



c. 9-14-20-18-15-4-21-3-3-9-15-14 1 12-1 3-9-2-5-18-19-5-7-21-18-9-4-1-4

Orden del alfabeto: **IC{INTRODUCCION A LA CIBERSEGURIDAD}**

d. 05110006_08130308020418_001115070001041908021418

Pista: parecido al 1.c

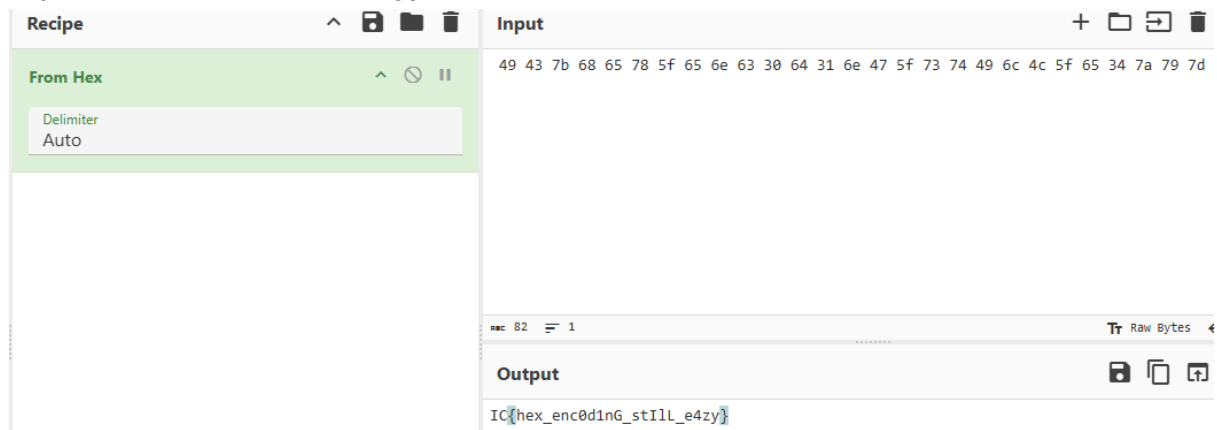
Como la posición 0 en el alfabeto no existe hay que sumarle +1 a cada número y esa cadena que te da cada numero es una posicion del abecedario.

16221117_19241419131529_112261811121521019132529

IC{flag_alphabetic_indices}

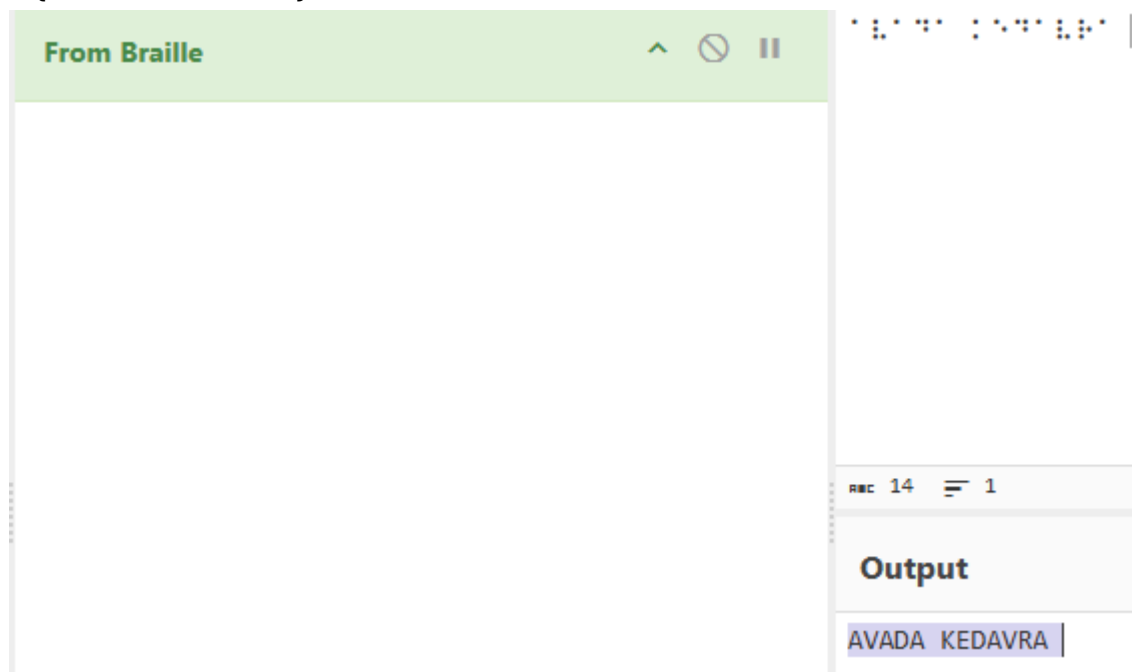
e. 49 43 7b 68 65 78 5f 65 6e 63 30 64 31 6e 47 5f 73 74 49 6c 4c 5f 65 34 7a 79 7d

IC{hex_enc0d1nG_st1lL_e4zy}



f. $\frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{4} \cdot \frac{1}{5} \cdot \frac{1}{6} \cdot \frac{1}{7} \cdot \frac{1}{8} \cdot \frac{1}{9} \cdot \frac{1}{10}$

IC{AVADA KEDAVRA}



g. Revele el mensaje en la imagen

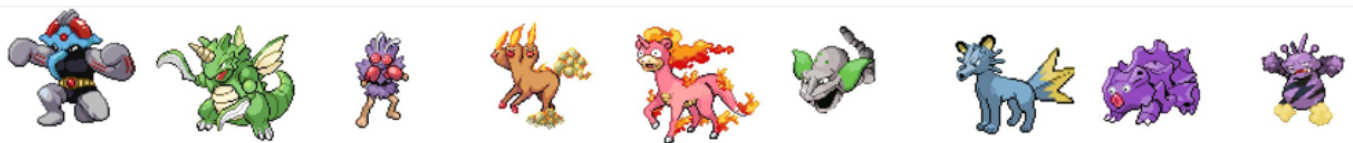


[83 85 78 55 85 71]
[57 114 121 85 53 106]
[84 50 81 120 98 109]
[100 102 84 106 66 102]
[77 50 53 68 99 110]
[120 119 86 68 82 57]

Base64 → IC{PokÉncOd1ng_N0_3nCr|pT4}



h. Revela el mensaje en la imagen



prueba 1:

Son todos la combinación de 2 pokemons. Busco los códigos de las fusiones que forman cada pokemon y eso lo paso a ASCII y eso a base64.

el primero son machoke y tentacruel 67 73,

el segundo rhydon y Scyther, 123 112

tercero Venonat y Hitmonchan, 48 107

el cuarto ponyta y dugtrio, 77 51

quinto rapidash y slowpoke, 78 79

sexto onix y Weepinbell, 95 70

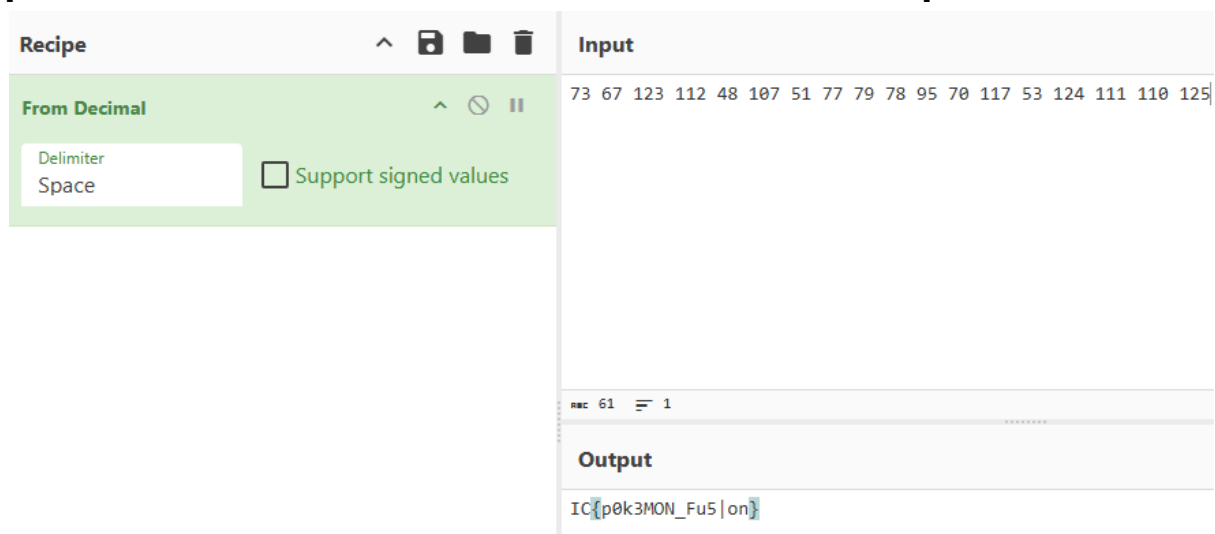
séptimo seadra y persian, 117 53

octavo jynx y Rhyhorn, 124 111

el último es weezing y Electabuzz 110 125

Me quedaría

[73 67, 123 112, 48 107, 51 77, 79 78, 95 70, 117 53, 124 111, 110 125]



IC{p0k3MON_Fu5|on}

Ejercicio 02:

Resolver el reto alojado en el puerto 11002 del sitio ic.catedras.linti.unlp.edu.ar

```
(ic)-(kali@kali)-[~]  
$ nc ic.catedras.linti.unlp.edu.ar 11002  
  
      _ _ _ _ _  
     / / / / /  
    / / / / /  
   / / / / /  
  / / / / /  
 / / / / /  
/ / / / /  
  
      _ _ _ _ _  
     / / / / /  
    / / / / /  
   / / / / /  
  / / / / /  
 / / / / /  
/ / / / /  
  
Bienvenidos! Tienen un segundo para encodear en base64 esta palabra:  
coger  
Mmmm tardaste mucho amiguito
```

mmmm rara la palabra

[illegible]

Pista: siempre es en base64

probar mas tarde ejecutar python tp1_ejer2.py ahora el servidor esta caido.

Ejercicio 03

Revele los siguientes mensajes. Para cada uno, indique qué cifrado se utilizó y si es de transposición o sustitución. Puede utilizar el sitio <http://rumkin.com/tools/cipher/>

a. }ratnelac a odnazepmE{CI

Esta invertido, es transposicion: **IC{Empezando a calentar}**

b. FZ{BPQL PF NRB PB ZLJMIFZX}
Pista: Mensaje que data del año 60 a.c

Cifrado Cesar, es de sustitución. El cifrado Cesar permite agregar un valor arbitrario, desplazando cada letra hacia adelante o hacia atrás. Tradicionalmente, el desplazamiento es 3, convirtiendo A en D, B en E, etc. En este caso se uso un desplazamiento -3 .// sacado del link del enunciado.

IC{ESTO SI QUE SE COMPLICA} // **NO ANDUVO**

The image shows a web-based Caesar cipher decryption tool. The interface is divided into two main sections: 'Recipe' on the left and 'Input/Output' on the right. The 'Recipe' section has a green background and contains a 'Substitute' panel. This panel has two input fields: 'Plaintext' with the value 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' and 'Ciphertext' with the value 'DEFGHIJKLMNOPQRSTUVWXYZ...'. There is also an 'Ignore case' checkbox which is currently unchecked. The 'Input' section on the right has a header 'Input' and a text area containing 'FZ{BPQL PF NRB PB ZLJMIFZX}'. Below this is a section for 'Output' which shows the result 'IC{ESTO SI QUE SE COMPLICA}'. At the bottom of the interface, there are controls for 'REC 28' and a list icon with the number '1'.

c. QK{IPWZI CV XWKW UIA}
Pista: Mensaje recibido de Roma
Desplazamiento Cesar. En este caso, desplazamiento -8 (desplazamiento negativo es a la izq, positivo a la derecha):
IPWZI CV XWKW UIA → AHORA UN POCO MAS

IC{AHORA UN POCO MAS}

Recipe
^
📁
🗑️

Substitute
^
🔇
⏸️

Plaintext
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Ciphertext
STUVWXYZABCDEFGHIJKLMNO...
☐ Ignore case

Input
QK{IPWZI CV XWKW UIA} |

REC 22
≡ 1

Output
IC{AHORA UN POCO MAS} |

d. pp epnwfus dvjipèym jx ln dtjcefv jfxrhw rq hkmmwjetfd wpvkla ij teznfxgymx t ceuced hgs knkielb féwcy ntwdaoos pwwva hfiekgvhgz csf kacwe, wpctiif kejyd hg cqljeèrf, byp wg baf hfqw poexl. mq hzfslh z hg cqljeèvm rv yp jqkwrp oi dyuaqyztmóv flqrsn utcibwjlfévpkt. rlc jvhr! nh nqfx et tg{g1k3plz3_wzc3w} . arjyí czí!

Pista: passphrase:le chiffre indechiffable

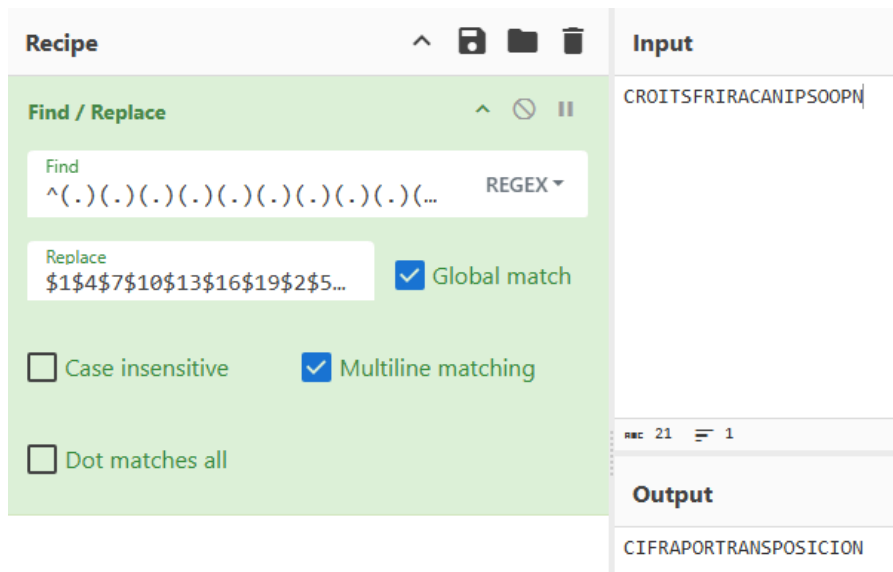
La pista “passphrase: le chiffre indechiffable” indica que el texto está cifrado con el cifrado Vigenère, usando como clave la frase francesa le chiffre indechiffable (“el cifrado indescifrable”)

Sacado del link del enunciado: [Vigenère](#) : Basado en cierta medida en el cifrado de desplazamiento cesáreo, este cifrado cambia la cantidad de desplazamiento con cada letra del mensaje, basándose en una frase de contraseña. Un cifrado bastante potente para principiantes. Funcionalmente similar a la "Variante Beaufort" y también compatible con autoclave

En Key: se debe ingresar la pista pero sin espacios

Key: lechiffreindechiffable

ic{v1g3ner3_rul3s}



Replace: \$1\$4\$7\$10\$13\$16\$19\$2\$5\$8\$11\$14\$17\$20\$3\$6\$9\$12\$15\$18\$21

An unauthorized encoded message was sent this morning. This may be very dangerous. Based on previous SIGINT our cryptographers have been told that to read it "a rail fence is needed". Can you help us read the message?

Se uso: [Rail Fence](#) que es un método ligeramente complicado en el que se alinean las letras en diferentes filas y luego se juntan las letras para crear el texto cifrado.//sacado del link del enunciado.

Recipe

Rail Fence Cipher Decode

Key

3

Offset

0

Input


TSaeile nh umnrwl ev tnoebi laao

Output

The Submariner will leave at noon

Resolver el reto alojado en el puerto 11004 del sitio ic.catedras.linti.unlp.edu.ar

Salida 1



Salida 2

por lo que veo mi pista parece ser que tengo que resolver algo en ROT en poco tiempo. Tengo que tener en cuenta que los tipos de ROT cambian, pero siempre con ROT.


```

data = conn.recvuntil(b'de esta frase:\n').decode()
print(data)

# Extraer el número de ROT con regex
match = re.search(r'ROT\s*(\d+)', data)
shift = int(match.group(1)) if match else 0

# Leer la frase
frase = conn.recvline().decode().strip()
print(f"Frase recibida: {frase}")
print(f"Aplicando ROT-{shift}")

# Aplicar ROT dinámico
respuesta = rot(frase, shift)
print(f"Respuesta cifrada: {respuesta}")

# Enviar respuesta
conn.sendline(respuesta.encode())

# Leer respuesta final
print(conn.recvall().decode())

```

Ejercicio 05

Averigue el mensaje al que se le aplicó la función de hash para generar los siguientes resúmenes:

Pista: Deduzca la función de hash a partir del formato (longitud) del resumen o hash.

a. 21232f297a57a5a743894a0e4a801fc3

longitud de 32 hex chars → MD5 hay que crackearlo. // no se puede hacer en cyberchef (no pude) use → <https://crackstation.net/>

IC{admin}

b. e731a7b612ab389fcb7f973c452f33df3eb69c99

longitud de 40 hex chars → SHA-1. **ic{p4ssw0rd}** // use → <https://crackstation.net/>

c. 796DD619207C4E357FD432FDF962C958BA1DF4CD6785246937223BC8D
C4FBF01794EBFF0159A175D9BE65B8EA4E7F46B80CCFFA4ED2A21773
D358C523DDDD382

longitud → cadena de 128 hex chars → SHA-512. use → <https://crackstation.net/> Tener en cuenta que no se admiten espacios, ni saltos de línea, poner todo junto.

IC{!!!gotosleep!!!}

Ejercicio 06

Resolver el reto alojado en el puerto 11006 del sitio ic.catedras.linti.unlp.edu.ar

Salida 1

```
(ic)-(kali@kali)-[~]  
$ nc ic.catedras.linti.unlp.edu.ar 11006  
  
████████████████████████████████████████████████████████████████████████████████  
████████████████████████████████████████████████████████████████████████████████  
  
████████████████████████████████████████████████████████████████████████████████  
████████████████████████████████████████████████████████████████████████████████  
  
Bienvenidos! Tienen un segundo para mandarme el hash MD5 (32 caracteres) de la siguiente  
labra:  
monkey  
Mmmm tardaste mucho amiguito
```

Salida 2

```
(ic)-(kali@kali)-[~]  
$ nc ic.catedras.linti.unlp.edu.ar 11006  
  
████████████████████████████████████████████████████████████████████████████████  
████████████████████████████████████████████████████████████████████████████████  
  
████████████████████████████████████████████████████████████████████████████████  
████████████████████████████████████████████████████████████████████████████████  
  
Bienvenidos! Tienen un segundo para mandarme el hash MD5 (32 caracteres) de la siguiente  
labra:  
butterfly  
Mmmm tardaste mucho amiguito
```

Salida 3, hago otra prueba para ver si fue casualidad que pida hash MD5 o es siempre asi.

```
(ic)-(kali@kali)-[~]  
$ nc ic.catedras.linti.unlp.edu.ar 11006  
  
████████████████████████████████████████████████████████████████████████████████  
████████████████████████████████████████████████████████████████████████████████  
  
████████████████████████████████████████████████████████████████████████████████  
████████████████████████████████████████████████████████████████████████████████  
  
Bienvenidos! Tienen un segundo para mandarme el hash MD5 (32 caracteres) de la siguiente  
labra:  
mickey  
Mmmm tardaste mucho amiguito
```

SI, la pista es esa, siempre es MD5.
primer script falla:

```
(ic)-(kali@kali)-[~]
└─$ python tp1_ejer6.py
[+] Opening connection to ic.catedras.linti.unlp.edu.ar on port 11006: Done

  _ _ _ _ _
 / / / / /
/_/_/_/_/_

  _ _ _ _ _
 / / / / /
/_/_/_/_/_

Bienvenidos! Tienen un segundo para mandarme el hash MD5 (32 caracteres) de la siguiente
labra:

Palabra recibida:
Hash MD5: d41d8cd98f00b204e9800998ecf8427e
[+] Receiving all data: Done (31B)
[*] Closed connection to ic.catedras.linti.unlp.edu.ar port 11006
666666
Mandaste fruta! nos vimo
```

hice algo mal y creo que es la lectura de la palabra.

```
(ic)-(kali@kali)-[~]
└─$ python tp1_ejer6.py
[+] Opening connection to ic.catedras.linti.unlp.edu.ar on port 11006: Done
palabra recibida: michael
Hash MD5: 0acf4539a14b3aa27deeb4cbdf6e989f
[+] Receiving all data: Done (41B)
[*] Closed connection to ic.catedras.linti.unlp.edu.ar port 11006
Correcto! la flag es IC{4gu4nt4_cr4ck3r!}

(ic)-(kali@kali)-[~]
```

IC{4gu4nt4_cr4ck3r!}

Script → tp1_ejer6.py

```
from pwn import *
import hashlib
import re

# Desactivar chequeo de actualizaciones de pwntools
import os
os.environ['PWNLIB_UPDATE'] = 'never'

# Conexión al servidor
conn = remote('ic.catedras.linti.unlp.edu.ar', 11006)

# Leer hasta la línea que contiene la palabra
conn.recvuntil(b'de la siguiente palabra:\n')
palabra = conn.recvline().decode().strip()
print(f"palabra recibida: {palabra}")
```

```
# Calcular hash MD5
hash_md5 = hashlib.md5(palabra.encode()).hexdigest()
print(f"Hash MD5: {hash_md5}")

# Enviar hash
conn.sendline(hash_md5.encode())

# Leer respuesta final
print(conn.recvall().decode())
```

Ejercicio 07

Resolver el reto alojado en el puerto 11007 del sitio ic.catedras.linti.unlp.edu.ar

El servidor informa el hash SHA-256 de una contraseña. El servicio pide que se crackee y devuelva la password que se usó para generarlo.

Pista: La password está entre las primeras 100 passwords del diccionario rockyou.

El diccionario lo puede descargar de <https://wiki.skullsecurity.org/Passwords>

```
(ic)-(kali@kali)-[~]
└─$ python tp1_ejer7.py
[+] Opening connection to ic.catedras.linti.unlp.edu.ar on port 11007: Done

      _ _ _ _ _
     / / / / /
    / / / / /
   / / / / /
  / / / / /
 / / / / /
/ / / / /

  _ _ _ _ _
 / / / / /
/ / / / /
/ / / / /
/ / / / /
/ / / / /
/ / / / /

Bienvenidos! Tienen un segundo para averiguar a que password pertenece este hash SHA-256 (
pista: Se encuentra entre las primeras 100 del diccionario rockyou.txt):

Hash recibido: 6382deaf1f5dc6e792b76db4a4a7bf2ba468884e000b25e7928e621e27fb23cb
Password encontrada: football
[+] Receiving all data: Done (44B)
[*] Closed connection to ic.catedras.linti.unlp.edu.ar port 11007
Correcto! la flag es IC{ea5y_Cr4ck_ExamPl3}
```

IC{ea5y_Cr4ck_ExamPl3}

Script → tp1_ejer7.py

```
from pwn import *
import hashlib
import os

# Desactivar chequeo de actualizaciones
os.environ['PWNLIB_UPDATE'] = 'never'
```

```

# Cargar las primeras 100 contraseñas del diccionario rockyou
with open('/home/kali/rockyou.txt', 'r', encoding='utf-8',
errors='ignore') as f:
    top_passwords = [line.strip() for _, line in zip(range(100), f)]

# Conexión al servidor
conn = remote('ic.catedras.linti.unlp.edu.ar', 11007)

# Leer hasta el hash
data_inicial = conn.recvuntil(b':\n').decode()
print(data_inicial)
hash_objetivo = conn.recvline().decode().strip()
print(f"Hash recibido: {hash_objetivo}")

# Buscar la contraseña que genera ese hash
password_encontrada = None
for pwd in top_passwords:
    hash_actual = hashlib.sha256(pwd.encode()).hexdigest()
    if hash_actual == hash_objetivo:
        password_encontrada = pwd
        break

# Enviar la contraseña si se encontró
if password_encontrada:
    print(f"Password encontrada: {password_encontrada}")
    conn.sendline(password_encontrada.encode())
    print(conn.recvall().decode())
else:
    print("No se encontró la contraseña en las primeras 100 del
diccionario.")

```

Ejercicio 09

AES en modo ECB El siguiente es el resultado de cifrar un string utilizando AES y codificando luego la salida en base64. Para cifrar se utilizó el algoritmo AES-128 modo ECB con la clave "CLAVE RE SECRETA" (sensible a mayúsculas, sin comillas)

dV5t6M4m2AcjYWsx9iO+YXlc0r0ClfwyTGtpuWdPh9fvH+8cejJWOHYq1qH7qA+Kj
7Lci133Awj3rnoq42p532+fvbN64oZ8R/TIMkhw47nmIM5gPN+rt45985jeilDbdpC
u1ig09Rzepl4/kawM1AzFtoMzTvadm11qSFp+UD81yiRz6HjaFLIIIIQnbzFrmcOI
OGEQ6LBEYz2cTW6JPBs7MHpqDrcrzZoLcb7Ah2jQSIIId+YZ90JmRt83yTe66a60kqL
5SoW7/463Suyyp9xDhrgFu6YS3ScNDgOamADlcKmLUTxrvYooZljl7s+thek3aBPrv
/yB84YNUhX7MOxjiTiP02nBJ1E1dOA0ew75BeARB4cHKVfLMnPMkjSYyiQ2eTWqYd4
cZ+14Z9joNVA1Uei8Pg4KITPfJYy3Mc=

Develar el mensaje mediante un script en python que use una librería como pycrypto


```
(ic)-(kali@kali)-[~]  
$ python tp1_ejer9.py
```

Texto descifrado: Advanced Encryption Standard (AES) es uno de los algoritmos de cifrado más utilizados y seguros actualmente disponibles. Es de acceso público, y es el cifrado que la NSA utiliza para asegurar documentos con la clasificación "top secret". La flag de este reto es IC{Si_sabes_la_password_es_facil_viteh}

IC{Si_sabes_la_password_es_facil_viteh}

Script → tp1_ejer9.py

```
from Crypto.Cipher import AES  
import base64  
  
# Clave exacta (debe tener 16 bytes para AES-128)  
clave = b'CLAVE RE SECRETA'  
  
# Texto cifrado en base64  
texto_base64 = (  
  
    'dV5t6M4m2AcjYwsxC9i0+YX1c0r0ClfwyTGtpuWdPh9fvH+8cejJWOHYq1qH7qA+Kj7Lci1  
    33Awj3rnoq42p532+fvbN64oZ8R/TlMkhW47nmIM5gPN+rt45985jeiIDbdpCu1ig09Rzep1  
    4/kawM1AzFtoMzTvadmX11qSFp+UD81yiRz6HjaFLIIIIQnbzFrmcOI0GEQ6LBEYz2cTW6JP  
    Bs7MHPqDrcrzZoLcb7Ah2jQSIId+YZ90JmRt83yTe66a60kqL5SoW7/463Suyyp9xDhrgFu6  
    YS3ScNDgOamADICkMLUTxrvYooZIJL7s+thek3aBPrv/yB84YNUhX7MOxjiTiP02nBJ1E1d0  
    A0ew75BeARB4cHKVfLMnPMkjSYyiQ2eTWqYd4cZ+14Z9joNVA1Uei8Pg4KITPfJYy3Mc='  
    )  
  
# Decodificar base64  
datos_cifrados = base64.b64decode(texto_base64)  
  
# Crear el descifrador AES en modo ECB  
cipher = AES.new(clave, AES.MODE_ECB)  
  
# Descifrar  
texto_plano = cipher.decrypt(datos_cifrados)  
  
# Eliminar padding (PKCS7)  
def quitar_padding(texto):  
    pad_len = texto[-1]  
    return texto[:-pad_len]  
  
# Mostrar resultado  
print("Texto descifrado:", quitar_padding(texto_plano).decode())
```

Ejercicio 10
Cifrado XOR

El siguiente es el resultado de cifrar un string utilizando XOR y codificando luego la salida en hex. El cifrado XOR utilizó una clave de 1 byte (1 carácter). Programar un script para encontrar la clave utilizada y develar el mensaje original.

Pista: Usar fuerza bruta para probar todas las posibles claves.

08296632232822342f27356637332366252f2034273466252928661e09146a66252e236
866162334296624332328296a662a2766202a2721662223662335322366342332296623
35660f053d092c7619257628193e7634676767737e7f737f73737f192527352f192e272
52d232334343b

```
(ic)-(kali@kali)-[~]  
$ python tp1_ejer10.py  
[+] Clave posible: 70 (F)  
Mensaje descifrado:  
No tendrias que cifrar con XOR, che. Pero bueno, la flag de este reto es IC{0j0_c0n_x0r!!!58959559_casi_hackeerr}
```

IC{0j0_c0n_x0r!!!58959559_casi_hackeerr}

Script → tp1_ejer10.py

```
import binascii  
  
hex_cifrado = (  
  
    '08296632232822342f27356637332366252f2034273466252928661e09146a66252e236  
    866162334296624332328296a662a2766202a27216622236623353223663423322966233  
    5660f053d092c7619257628193e7634676767737e7f737f73737f192527352f192e27252  
    d232334343b'  
)  
  
datos_cifrados = bytes.fromhex(hex_cifrado)  
  
def es_legible(texto):  
    letras = sum(c.isalpha() for c in texto)  
    espacios = texto.count(' ')  
    return letras > len(texto) * 0.6 and espacios > 5  
  
for clave in range(256):  
    resultado = bytes([b ^ clave for b in datos_cifrados])  
    try:  
        texto = resultado.decode('utf-8')  
        if es_legible(texto):  
            print(f"[+] Clave posible: {clave} ({chr(clave)})")  
            print("Mensaje descifrado:")  
            print(texto)  
            print("-" * 60)  
    except UnicodeDecodeError:  
        continue
```

