Practica 1

Por Mateo Carusotti

Ctfd caru

Ejercicio 01

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

a. 73 67 123 99 97 108 101 110 116 97 110 100 111 95 109 111 116 111 114 101 115 125

   Es un mensaje codificado en Ascii decimal por lo que su decodificacion seria:

   https://gchq.github.io/CyberChef/#recipe=From_Charcode('Space',10)&input=Cgo3MyA2NyAxMjMgOTkgOTcgMTA4IDEwMSAxMSAxM

   IC{calentando_motores}

b. SUN7M25jMGRlYXJfbjBfM3NfZW5jcjFwdDRyfQ==

   Es un mensaje codificado en algun tipo de Base, en este caso Base64

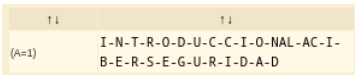
   https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true,false)&input=CgpTVU43TTI1ak1HUmxZWEpmYmpCZk0zTmZaVzVqY2pGd2REUnlmUT09&oenc=65001

   IC{3nc0dear_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

   Esta en un formato donde cada numero representa una letra del alfabeto

   En CiberChef no me retorno el mensaje en cambio en dcode si pero con los guiones en el medio



   IC{INTRODUCCION A LA CIBERSEGURIDAD}

d. 05110006_08130308020418_0011050001041908021418

   IC{FLAG_INDICES_ALFABETICOS}

   Primero separo el codigo dado en pares, sin contar los "_" junto el resultado dado por decodificarlo en A1Z26

e. 49 43 7b 77 68 65 6e 5f 75 5f 64 69 45 5f 79 6f 75 5f 61 72 65 5f 35 37 30 30 35 7d

   Esta codificado en Hexadecimal, se aprecia por la separacion en 2 caracteres y por los que usa como numero de 1-9 y letras de a-f

   https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')&input=NDkgNDMgN2IgNzcgNjggNjUgNmUgNWYgNzUgNWYgNjQg

   IC{when_u_diE_you_are_57005}

f. .- -... .-. .- -.-. .- -.. .- -... .-. .-

   A simple vista se ve que es morse por lo que su decodificacion seria

   https://gchq.github.io/CyberChef/#recipe=From_Morse_Code('Space','Line feed')&input=Li0gLS4uLiAuLS4gLi0gLS4tLiAuLSAtLi4gLi0gLS4uLiAuLS4gLi0

   ABRACADABRA

Ejercicio 02

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

IC{EaSy_enc0d1ng_exaMple}

```python
io = start()

io.readuntil('palabra:\n')


texto = io.readline().strip()



print(texto)

texto = texto.decode()

print(texto)

resultado = base64.b64encode(texto.encode())

print(resultado)

io.send(resultado + b"\n")

# shellcode = asm(shellcraft.sh())
# payload = fit({
#     32: 0xdeadbeef,
#     'iaaa': [1, 2, 'Hello', 3]
# }, length=128)
# io.send(payload)
# flag = io.recv(...)
# log.success(flag)

io.interactive()
```

Pasos:

1. Obtuve la palabra que me daba el host al que me conecté, le quite el salto de linea
2. Lo decodifique para tenerlo como un strintg y no un byte array
3. Lo encode a base64
4. Como el resultado ya esta en bytes solo le agrego el salto de linea

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

   IC{Empezando_a_calentar}

   No utilice ningun cifrado simplemente inverti el orden de las palabras

   Es un cifrado de Sustitucion, ya que las letras del texto original siguen en el texto cifrado pero en otro orden

b. FZ{BPQL PF NRB PB ZLJMIFZX}

   Pista: Mensaje que data del año 60 a.c

   IC{ESTO SI QUE SE COMPLICA}

   Utilice Rot13 con fuerza bruta, termino resultando Rot3

   Es un cifrado por sustitucion ya que la letras se remplazan por otras sistematicamente

   https://gchq.github.io/CyberChef/#recipe=ROT13_Brute_Force(true,true,false,100,0,true,'')&input=Rlp7QlBRTCBQRiBOUklgUElgW

c. QK{IPWZI CV XWKW UIA}

   Pista: Mensaje recibido de Roma

   IC{AHORA UN POCO MAS}

   Utilice Rot13 con fuerza bruta, termino resultando Rot18

   Es un cifrado por sustitucion ya que la letras se remplazan por otras sistematicamente

   https://gchq.github.io/CyberChef/#recipe=ROT13_Brute_Force(true,true,false,100,0,true,'')&input=UUt7SVBXWkkgQ1YgWFdLVyB\

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

   Pista: passphrase:le chiffre indechiffrable

   ic{v1g3ner3_rul3s}

   En este caso con ayuda de la pista y buscando al incluir una palabra clave y por la forma del texto pude saber que debia utilizar el sistema Vigenere

   Es un cifrado de sustitucion ya que las letras del mensaje original se remplazan por otras.

   https://gchq.github.io/CyberChef/#recipe=Vigenère_Decode('lechiffreindechiffrable')&input=cHAgZXBud2Z1cyBkdmppcOh5bSB(

e. Descifra esta extraña palabra: CROITSFRIRACANIPSOOPN

   Pista: 7 × 3

   IC{CIFRAPORTRANSPOSICION}

   Primero separe la frase original en 7 grupos de 3 letras, los puse en orden de columna, fui juntandolas letras de izquierda a derecha y de arriba a abajo.

   Es un cifrado de trasnposicion ya que las letra del mensaje oculto estan en la frase original.

f. Militar exfiltration

   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?

   TSaeile nh umrnrwl ev tnoebi laao

   Clave The Submariner will leave at noon

   Con la pista de "a rail fence is needed" se puede deducir que se tiene que usar el sistema de Rail Fence, prove con la clave igual a 3 y dio este resultado

   Es un sistema de trasnposicion ya que se cambian las letras de la palabra encriptada para dar como resultado la palabra sin encriptar.

   https://gchq.github.io/CyberChef/#recipe=Rail_Fence_Cipher_Decode(3,0)&input=VFNhZWlsZSBuaCB1bXJucndsIGV2IHRub2Via3

Ejercicio 04

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

```python
def rot_funcion(frase,desplazamiento):
    lowercase_positions = {
    'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5,
    'f': 6, 'g': 7, 'h': 8, 'i': 9, 'j': 10,
    'k': 11, 'l': 12, 'm': 13, 'n': 14, 'o': 15,
    'p': 16, 'q': 17, 'r': 18, 's': 19, 't': 20,
    'u': 21, 'v': 22, 'w': 23, 'x': 24, 'y': 25, 'z': 26
}
    inverted_dict = {v: k for k, v in lowercase_positions.items()}
    resultado = []
    for i in frase:
        palabra = ""
        for letra in i:
            pos = lowercase_positions[letra]
            letra_posicion = pos + desplazamiento
            if(letra_posicion > 26):
                letra_posicion = letra_posicion - 26

            palabra = palabra + str(inverted_dict[letra_posicion])

        resultado.append(palabra)

    return resultado

io = start()

io.readuntil('ROT ')

rot = io.readline()

rot= rot.decode()

rot = rot.split()
rot = rot[0]

frase = io.readline()
frase = frase.decode()

res = rot_funcion(frase.split(),int(rot))


resultado = " ".join(res)


io.send(resultado.encode())

# shellcode = asm(shellcraft.sh())
# payload = fit({
#     32: 0xdeadbeef,
#     'iaaa': [1, 2, 'Hello', 3]
# }, length=128)
# io.send(payload)
# flag = io.recv(...)
# log.success(flag)

io.interactive()
```

IC{izi_rOt_ex4mpLe}

Pasos que hice para resolver el problema

1. Para este ejercicio primero tomo el tipo de ROT que debo hacer
2. Tomo la palabra a encriptar
3. Para obtener el resultado llamo a mi funcion rot_funcion

   Detallo la funcion al final
4. Voy concatenando las palabras para formar la frase resultado y envio encodeado el resultado

Pasos de la funcion rot_funcion()

1. Primero defino un diccionario para cada letra le asigno su valor, representando su posicion en el alfabeto
2. Defino un diccionario inverso, para mas comodidad, es decir por cada posicion del alfabeto tengo la letra asociada
3. Itero por cada palabra de la frase
4. Itero por cada letra de la palabra
5. De la letra obtengo su posicion
6. Miro si me paso de 26
7. Le concateno a una palabra resultado la letra pasada por ROT
8. Agrego a una lista resultado la palabra Roteada

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

   IC{admin}

   Primero con la herramienta CyberChef puse este hash en la opcion de analizar hash, una de las opciones de hash que arrojo comor esultado fue MD5. Entonces con este comando utilizando John the Ripper `$ john --format=raw-md5 hash_p1.txt` obtuve como resultado que el texto plano era admin.

b. e731a7b612ab389fcb7f973c452f33df3eb69c99

   IC{p4ssw0rd}

   Por el formato esta hasheado con SHA1, CyberChef me dio esta respuesta, con hashcat utilice el siguiente comando utilizando un diccinario que descagargue de Rockyou

   ```
   hashcat -m 100 -a 0 hash_p5b.txt /home/caru/Downloads/rockyou.txt
   ```

   El resultado fue: p4ssw0rd

c. 796DD619207C4E357FD432FDF962C958BA1DF4CD6785246937223BC8DC4FBF01794EBFF0159A175D9BE65B8EA4E7F46B

   IC{!!!gotosleep!!!}

   Por el formato esta hasheado con SHA512, CyberChef me dio esta respuesta, con hashcat utilice el siguiente comando utilizando un diccinario que descagargue de Rockyou

   ```
   $ hashcat -m 1700 -a 0 hash_p5c.txt /home/caru/Downloads/rockyou.txt
   ```

Ejercicio 06

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

```python
io = start()

io.readuntil('palabra:\n')

palabra = io.readline().decode().strip()

print(palabra)

palabra_haseada = hashlib.md5(palabra.encode('utf-8')).hexdigest()

print(palabra_haseada)

io.send(palabra_haseada)


# shellcode = asm(shellcraft.sh())
# payload = fit({
#     32: 0xdeadbeef,
#     'iaaa': [1, 2, 'Hello', 3]
# }, length=128)
# io.send(payload)
# flag = io.recv(...)
# log.success(flag)

io.interactive()
```

IC{4gu4nt4_cr4ck3r!}

Pasos:

1. Importo hashlib para poder hashear la clave
2. Leo la palabra, la desencode y le quito el salto de linea
3. Hasheo la palabra en md5
4. Envio la palabra ya hasheada

Ejercicio 07

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

IC{ea5y_Cr4ck_ExamPl3}

PD: No subi el archivo de rockyou_100.txt asi que no va a funcionar asi nomas

Pasos:

1. Obtengo la clave hasheada
2. Abro el archivo de rockyou (lo modifique para que solo tenga las priemras 100 claves y no todo)
3. Hago un list comprehension para obtener la contraseña en ese archivo que cumpla con la condicion de que si la hashea sha256 se igual a la contraseña hasheada que me dieron al principio
4. Envio la contraseña crackeada

Ejercicio 08

Crackear usando hashcat o johntheripper el siguiente hash md5 con salt:

85f978e2c12fedbf8869b219a1b2576a

Pista: Formato MD5($salt.$pass)

Pista: Usa como salt el prefijo “IntroCiberseguridad”

Pista: rockyou.txt

Flag: welovejesus

Utilice hashcat con el siguiente comando

`hashcat -m 20 -a 0 hash_p1e8t.txt rockyou.txt`

En el archivo `hash_p1e8t.txt` tengo la clave hasheada y el prefijo.

Con -m 20 indico que el sistema de hash es md5 pero con el formato $salt.$pass

Ejercicio09

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)

dV5t6M4m2AcjYWsxC9iO+YXlc0r0ClfwyTGtpuWdPh9fvH+8cejJWOHYq1qH7qA+Kj7Lci133Awj3rnoq42p532+fvbN64oZ8R/TlMkhw4

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

IC{Si_sabes_la_password_es_facil_viteh}

Pasos:

1. Me guardo la clave
2. Utilizo el modulo de base64 para descencriptar el mensaje y obtener le mensaje encriptado AES.
3. Creo un objeto AES en el modo ECB con la clave
4. Descencripto el mensaje
5. Lo imprimo quitandole caracteres no deseables

Ejercicio 10

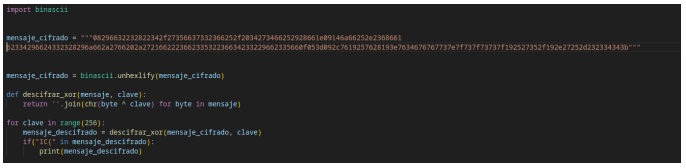
Cifrado XOR

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.

0829663223282234 2f27356637332366252f203427346625292866 1e09146a66252e2368661623342966243323282 96a662a2766 2

IC{Oj0_c0n_x0r!!!58959559_casi_hackeerr}



Pasos:

1. El mensaje cifrado en hexadecimal lo paso a bytes
2. Luego por cada convinacion bits en un byte itero aplicando la funcion xor con el mensaje y la clave que es el indice de la iteracion
3. La funcion xor eleva cada byte del mensaje cifrado por la clave que recibe como parametro
4. Imprimo solo un resultado que tenga el formato de la Flag por comodidad

Aclaracion: el mensaje deberia estar todo unido para que funciones, le puse un salto de linea para que entre en la captura

Ejercicio 12

Revele el mensaje cifrado con RSA:

p:14116810449622477004714246307083749256487585440938819 77

q:10254777647391161702320017559629265694898389491212327 67

e:65537

C:24480032935390633635038225308868097264670696263978384433594823408502234840076325655977009553817777036

IC{sabiendo_P_y_Q_es_muy_facil}

Pasos:

1. Calculo **n,** n = 14116810449622477004714246307083749256487585440938819 77 * 10254777647391161702320017559629265694898389491212327 67
2. Calculo phi(n)**,** phi(n) = (14116810449622477004714246307083749256487585440938819 77 - 1) * (10254777647391161702320017559629265694898389491212327 67 - 1)
3. Calculo d, que es el inverso multiplicativo modular, d = modinv(65537,68718166020) = 10139518779603842858325297218453200859096744106990096371075591599781241579878692783419384913985639659159
4. Ahora para obtener el mensaje cifrado M hago: M = (C^d) mod n = 12944558095847225227225518176664911278275639975012800448778328962921617318 1
5. Paso M a string

Ejercicio 13

Resolver el reto alojado en el puerto **11012** del sitio **ic.catedras.linti.unlp.edu.ar**

IC{rsa_is_eeeeeeasy}

```python
import libnum

io = start()

io.readuntil('texto:\n')

p = int(io.readline().split()[1].decode())

q = int(io.readline().split()[1].decode())

e = int(io.readline().split()[1].decode())

c = int(io.readline().split()[1].decode())

n = p * q
phi_n = (p - 1) * (q - 1)

d = libnum.invmod(e, phi_n)

M = pow(c, d, n)

M = libnum.n2s(M)

io.send(M)

# shellcode = asm(shellcraft.sh())
# payload = fit({
#     32: 0xdeadbeef,
#     'iaaa': [1, 2, 'Hello', 3]
# }, length=128)
# io.send(payload)
# flag = io.recv(...)
# log.success(flag)

io.interactive()
```

Pasos:

1. Obtengo los datos que me envian, p,q,e y c
2. Calculo n
3. Calculo phi(n)
4. Calculo d, usando invmod(e,phi(n)) que es el inverso multiplicativo modular
5. Calculo M que es el mesaje original, haciendo (c^d) mod n
6. Paso M de numero a string, envio M

Ejercicio 14

Revele el mensaje cifrado con RSA, esta vez no tenemos P ni Q.

Pista: Hay que factorizar o encontrar un buen lugar donde lo hagan...

n: 14524491846245356357574490859882044874942222485094938992997 59

e: 65537

C: 12807439447128571430606279699385388519111719501259799450261 52

IC{factordb_ftw}

```python
import libnum

p = 11533247751794313121781207976 79

q = 12593583489078931081753915715 21

e = 65537

c = 12807439447128571430606279699385388519111719501259799450261 52

n = p * q
phi_n = (p - 1) * (q - 1)

d = libnum.invmod(e, phi_n)


M = pow(c, d, n)

M = libnum.n2s(M)

print(f'Mensaje descifrado: {M}')
```

Pasos:

1. Calculo p y q a partir de ingresar el valor de n en http://www.factordb.com/index.php?query=14524491846245356357574490859882044874942222485094938992997 59
2. Me dio como resultado los valores 11533247751794313121781207976 79 y 12593583489078931081753915715 21
3. El procedimiento sigue de manera normal

Ejercicio 15

Desencriptar RSA sin p ni q

Resolver el reto alojado en el puerto **11017** del sitio **ic.catedras.linti.unlp.edu.ar**

IC{Listo:_la_NSA_te_esta_buscando}

```python
import libnum
from sympy import factorint

io = start()

io.readuntil('texto:\n')

n = int(io.readline().split()[1].decode())

e = int(io.readline().split()[1].decode())

c = int(io.readline().split()[1].decode())

factors = factorint(n)

factors = list(factors.keys())

print(factors)

p= factors[0]
q = factors[1]

n = p * q

phi_n = (p - 1) * (q - 1)

d = libnum.invmod(e, phi_n)

M = pow(c, d, n)

M = libnum.n2s(M)

print(f'Mensaje descifrado: {M}')

io.send(M)

io.interactive()
```

Pasos:

1. Recibo n,e y c
2. Factorizo n con la funcion factorint que me brinda sympy, de ese diccionario resultado tomo las keys que son el numero factorizado y se lo asigno a p y q arbitariamente
3. Realizo el calculo de M de manera exacta a como venia haciendo

Ejercicio 16

Diffie-Hellman

Resolver el reto alojado en el puerto **11018** del sitio **ic.catedras.linti.unlp.edu.ar**

IC{dame_tu_clave_millhouse}

```python
import libnum
import sympy
io = start()

io.readuntil('Hellman:\n')

p = int(io.readline().decode().strip().split()[1])

g = int(io.readline().decode().strip('=')[2])

public_alice = int(io.readline().decode().strip().split()[1])

private_bob = int(io.readline().decode().strip().split()[1])

clave_compartida = pow(public_alice, private_bob, p)

io.send(str(clave_compartida).encode())

io.interactive()
```

Pasos:

1. Tomo los datos que recibo del host al que em conecto p, g, la clave publica de alice y la clave privada de bob, los paso a enteros.
2. Calculo la clave compartida en base de elevar la clave publica de alice por la de bob, eso mod p

Explicacion del paso 2:

Yo tengo los numeros primos p y g. Cuento la clave publica de alice que vendria a ser g^a mod p.

Cuento la clave privada de Bob, es decir de b. Yo se que bob para saber cual es la clave compartida con alice tiene que hacer la clave publica de alice elevado con su clave privada mod p

Ejercicio 17

IC{Steg0_basiiiccccc!}

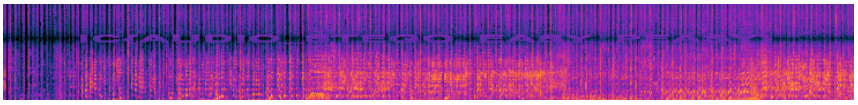
Pasos:

1. Con la herramienta steghide obtengo la informacion oculta de la imagen oculto.jpg, `steghide extract -sf oculto.jpg`
2. Obtengo la frase en base 32 en un archivo, decodifico la informacion desde base32 https://gchq.github.io/CyberChef/#recipe=From_Base32('A-Z2-7%3D',true)&input=SVZXQ0FZTFNPUlNTQVpERkVCWFdHNUxNT1JRWEVJREJOUlRXNklERk5ZUUhLM1JBTU5RVzRZTE1FQlJYS1l

Ejercicio 18

IC{AUDIO_STEGO_EASY_PEASY}

Usando Audacity abro el archivo aereje.wav, selecciona la opcion de verlo con Espectrograma



Ejercicio 20

IC{symmetric_gpg_d0n3!}

```python
import subprocess
from multiprocessing import Process, Value

def decriptar(comienzo, fin, contraseñas, encontrado):
    for i in contraseñas[comienzo:fin]:
        if encontrado.value:
            break
        resultado = subprocess.run(
            ['gpg', '--batch', '--passphrase', i, '--decrypt', 'flag.txt.gpg'],
            capture_output=True,
            text=True
        )
        print(f'intento: {i}')
        if 'descifrado fallido:' not in resultado.stderr:
            print(resultado.stdout)
            encontrado.value = True
            break

with open('diccionario', 'r') as dic:
    contraseñas = dic.readlines()

with open('diccionarioreves', 'r') as dicr:
    contraseñas2 = dicr.readlines()

mitad = len(contraseñas) // 2


encontrado = Value('b', False)

p1 = Process(target=decriptar, args=(0, mitad, contraseñas, encontrado))
p2 = Process(target=decriptar, args=(0, mitad, contraseñas2, encontrado))

p1.start()
p2.start()

p1.join()
p2.join()

print('Terminaron los procesos')
```

Aclaracion: Importe multiprocessing para obtener los resultados mas rapido pero voy a explicar como funciona un unico proceso

Pasos:

1. Abro el diccionario de claves y me guardo las contraseñas
2. Llamo a la funcion decriptar la cual recorre de incio a fin las contraseñas
3. Por cada contraseñas utiliza el comando de gpg `gpg --batch --passphrase i --decrypt flag.txt.gpg` el cual por cada contraseñas intenta utilizar la misma parta descencriptar el texto en gpg

4. Si no esta la cadena descifrado fallido (esto es asi ya que tengo la consola en español) es que encontre la contraseña correcta

Ejercicio 21

IC{Se_encriptar_con_pgp}

Pasos:

1. gpg --import public.gpg

   gpg: clave FACBCF6385B18782: clave pública "Introduccion a la Ciberseguridad (IC{flagggggggggg_1n_c0mm3nt}) jere@ic.com" importada
   gpg: Cantidad total procesada: 1
   gpg: importadas: 1

2. gpg --output encriptar.txt.asc --armor --encrypt --recipient "jere@ic.com" encriptar.txt

3. cat encriptar.txt.asc | nc ic.catedras.linti.unlp.edu.ar 12003
   Correcto! la flag es IC{Se_encriptar_con_pgp}