

# Informe Laboratorio 1

## Sección 1

Valentina Díaz Gonzalez  
e-mail: valentina.diaz\_go@mail.udp.cl

Septiembre 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 . . . . .	12

## 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 jypwavyhmpf f zlnbyphkh lu ylkz
3 ixovzumxglog e ykmaxojgj kt xkjky
4 hwnuytlwfkf d xjlzwnifi js wjiyx
5 gvmtxskvejme c wikyvmeheh ir vihiw
6 fulswrjudild b vhxulgdg hq uhghv
7 etkrvqitchkc a ugiwtkfch 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 lxznkbtw xg kxwxl
17 ujahlgysxas q kwymjavsv wf jwvwk
18 tizgkfxirwzr p jvxlizuru ve ivuvj
19 shyfjewhqvq o iuwkhytqt ud hutui
20 rgxeidvgpuxp n htvjgxspc tc gtsth
21 qfwdhcufofwo m gsuifwrwr sb ffsrg
22 pevcbgtensvn l frthevqng ra erqrf
23 odubfasdmrum k eqsgdupmp qz dqpqe
24 nctaezrclqtl j dprfctolo py cpopd
25 mbszdyqbksk i coqebnskn 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

#### 3.1. Actividad 1

Para la actividad 1 hice la siguiente consulta en ChatGPT: *“hola, quiero un programa en python en cual permita cifrar texto utilizando el algoritmo Cesar. Me gustaría que al momento de correr en programa en el terminal darle como parametro el string a cifrar y un numero que representa el corrimiento. Y que al momento de mostrar en pantalla no diga texto cifrado sino que solo salga la oracion cifrada”*, me respondió el siguiente código:

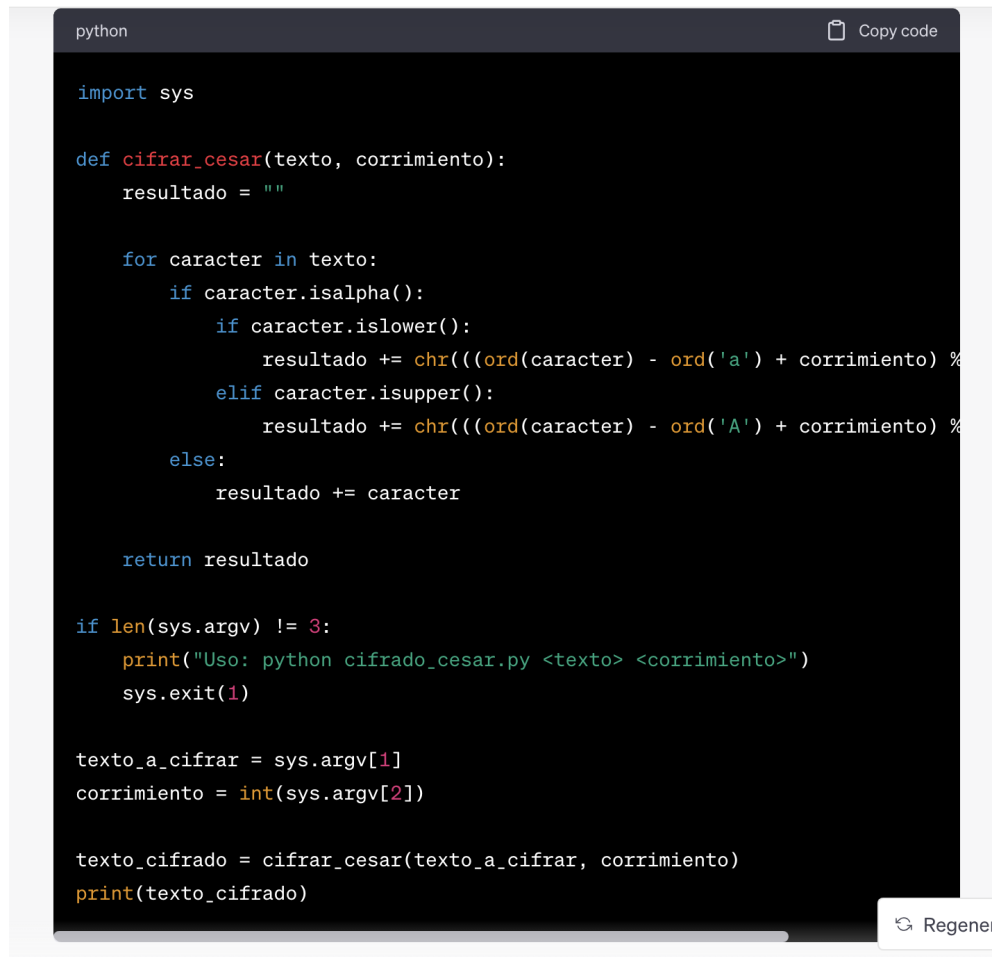


hola, quiero un programa en python en cual permita cifrar texto utilizando el algoritmo Cesar. Me gustaría que al momento de correr en programa en el terminal darle como parametro el string a cifrar y un numero que representa el corrimiento. Y que al momento de mostrar en pantalla no diga texto cifrado sino que solo salga la oracion cifrada

---

Figura 1: Consulta a ChatGPT

La respuesta entregada fue la siguiente:

A screenshot of a code editor window titled 'python'. The code implements a Caesar cipher. It defines a function 'cifrar\_cesar' that takes 'texto' and 'corrimiento' as arguments. The function iterates through each character in 'texto'. If the character is a lowercase letter, it shifts its position by 'corrimiento' within the alphabet. Similarly, if it's an uppercase letter, it shifts it within the uppercase alphabet. Non-alphabetic characters are left unchanged. The function returns the resulting 'resultado' string. Below the function, there's a check for the correct number of command-line arguments (3). If not 3, it prints a usage message and exits. Otherwise, it takes the text and shift from the arguments, calls the cipher function, and prints the result. The code is syntax-highlighted. A 'Copy code' button is in the top right, and a 'Regenerar' button is in the bottom right.

```
python
import sys

def cifrar_cesar(texto, corrimiento):
    resultado = ""

    for caracter in texto:
        if caracter.isalpha():
            if caracter.islower():
                resultado += chr(((ord(caracter) - ord('a') + corrimiento) % 26) + ord('a'))
            elif caracter.isupper():
                resultado += chr(((ord(caracter) - ord('A') + corrimiento) % 26) + ord('A'))
            else:
                resultado += caracter

    return resultado

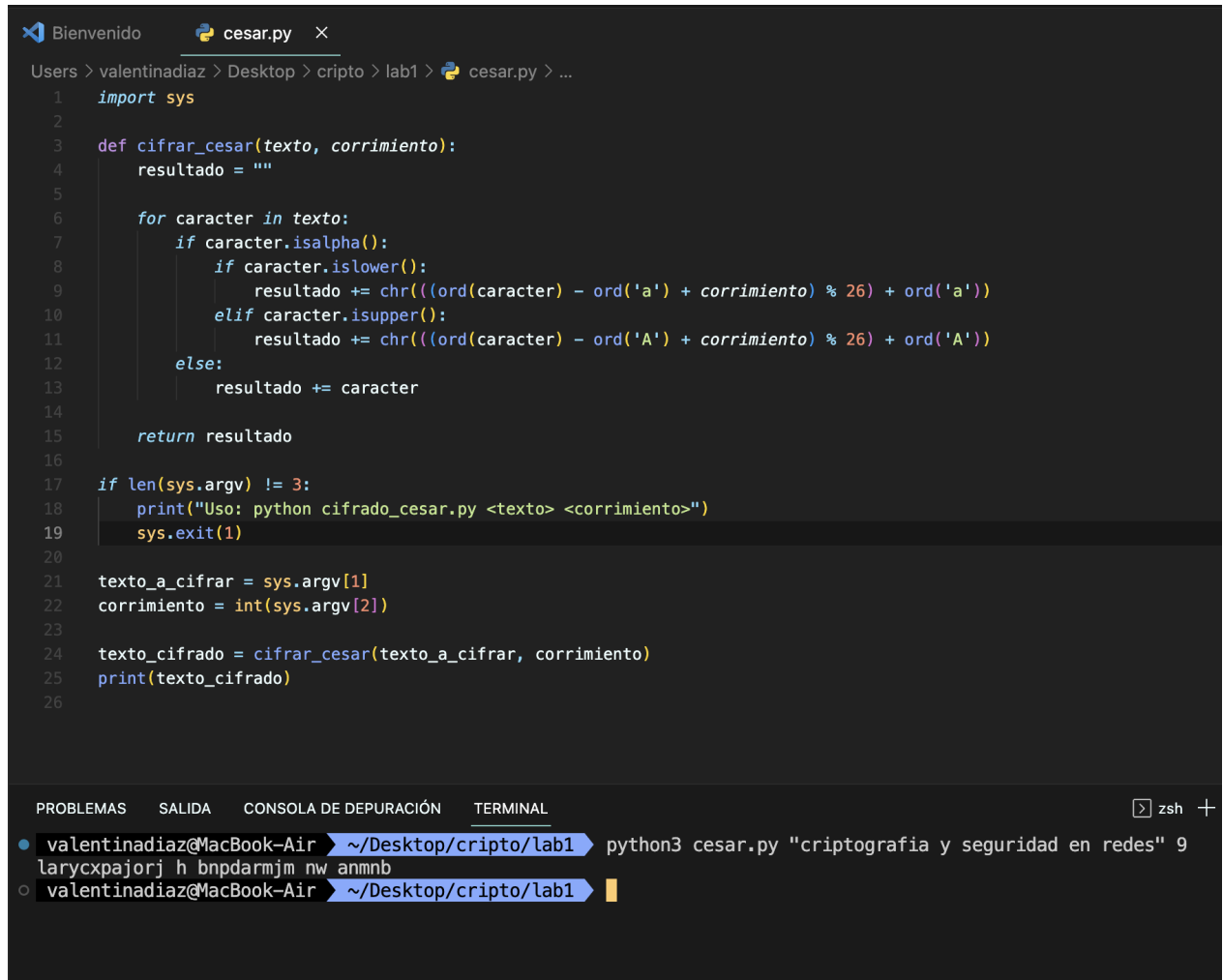
if len(sys.argv) != 3:
    print("Uso: python cifrado_cesar.py <texto> <corrimiento>")
    sys.exit(1)

texto_a_cifrar = sys.argv[1]
corrimiento = int(sys.argv[2])

texto_cifrado = cifrar_cesar(texto_a_cifrar, corrimiento)
print(texto_cifrado)
```

Figura 2: Respuesta de ChatGPT

Se ejecutó el código en VS Code de la manera indicada en el enunciado cumpliendo con el cifrado César. En la siguiente imagen, se evidencia la forma en que se ejecutó el código y el resultado de este mismo, esto se observa en la parte inferior de la imagen, en el terminal.



The image shows a code editor window titled 'cesar.py' with the following Python code:

```
1 import sys
2
3 def cifrar_cesar(texto, corrimiento):
4     resultado = ""
5
6     for caracter in texto:
7         if caracter.isalpha():
8             if caracter.islower():
9                 resultado += chr((ord(caracter) - ord('a') + corrimiento) % 26 + ord('a'))
10            elif caracter.isupper():
11                resultado += chr((ord(caracter) - ord('A') + corrimiento) % 26 + ord('A'))
12            else:
13                resultado += caracter
14
15     return resultado
16
17 if len(sys.argv) != 3:
18     print("Uso: python cifrado_cesar.py <texto> <corrimiento>")
19     sys.exit(1)
20
21 texto_a_cifrar = sys.argv[1]
22 corrimiento = int(sys.argv[2])
23
24 texto_cifrado = cifrar_cesar(texto_a_cifrar, corrimiento)
25 print(texto_cifrado)
26
```

Below the code editor is a terminal window with the following output:

```
valentinadiaz@MacBook-Air ~/Desktop/cripto/lab1 python3 cesar.py "criptografia y seguridad en redes" 9
larycxpajorj h bnpdarmjm nw anmnb
```

Figura 3: Ejecución de código

## 3.2. Actividad 2

Para la actividad 2, primero se investigó como era el paquete ping, que valores tienen los campos de ICMP a partir de la captura adjuntada en el enunciado. Las características estudiadas son las siguientes:

```

▼ Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x7ec8 [correct]
  [Checksum Status: Good]
  Identifier (BE): 11 (0x000b)
  Identifier (LE): 2816 (0x0b00)
  Sequence Number (BE): 1 (0x0001)
  Sequence Number (LE): 256 (0x0100)
  ▶ [No response seen]
  Timestamp from icmp data: Aug 25, 2023 13:42:39.000000000 UTC
  [Timestamp from icmp data (relative): 0.476208058 seconds]
▼ Data (48 bytes)
  Data: fb43070000000000101112131415161718191a1b1c1d1e1f202122232425262728292a2b...
  [Length: 48]

```

Figura 4: Características de paquete ICMP

Por lo tanto, el paquete a crear debe tener:

- Type con valor 8.
- Identifier válido para Ubuntu y distinto de 0.
- Sequence number que sea incremental y válido, que parte desde 1 y aumenta en cada paquete.
- Timestamp debe corresponder a la fecha y hora al momento de ejecutar el programa.
- Data debe ser de 48 bytes, donde el primer debe ser un carácter en hexadecimal y el resto debe ser así 000100000000000101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637.
- La dirección de destino sea 127.0.0.1.

Por lo tanto, la pregunta a ChatGPT fue:

*“hola, necesito crear un programa con python3. y scapy en el cual se haga lo siguiente:*

- *enviar paquetes enviando un caracter de un string por paquete, por lo tanto, se deben enviar tantos paquetes como caracteres tenga el string*
- *el campo type debe valer 8*
- *el code debe valer 0*
- *debe tener un checksum acorde a paquete icmp tipo ping ubuntu*
- *debe tener en el campo identifier de icmp sea valido y distinto de 1 y 0*
- *debe tener un sequence number incremental y valido, que parte desde 1 y aumenta en cada paquete*
- *el campo data debe ser de 48 bytes, donde byte más significativo debe ser el caracter del string en hexadecimal y el resto debe ser así 00 01 00 00 00 00 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37*

- la direccion de destino debe ser 127.0.0.1
- el timestamp debe corresponder al de la hora y fecha local al momento de correr el programa porfavor, gracias”

La siguiente imagen corresponde a la pregunta hecha a ChatGPT:



hola, necesito crear un programa con python3. y scrapy en el cual se haga lo siguiente:

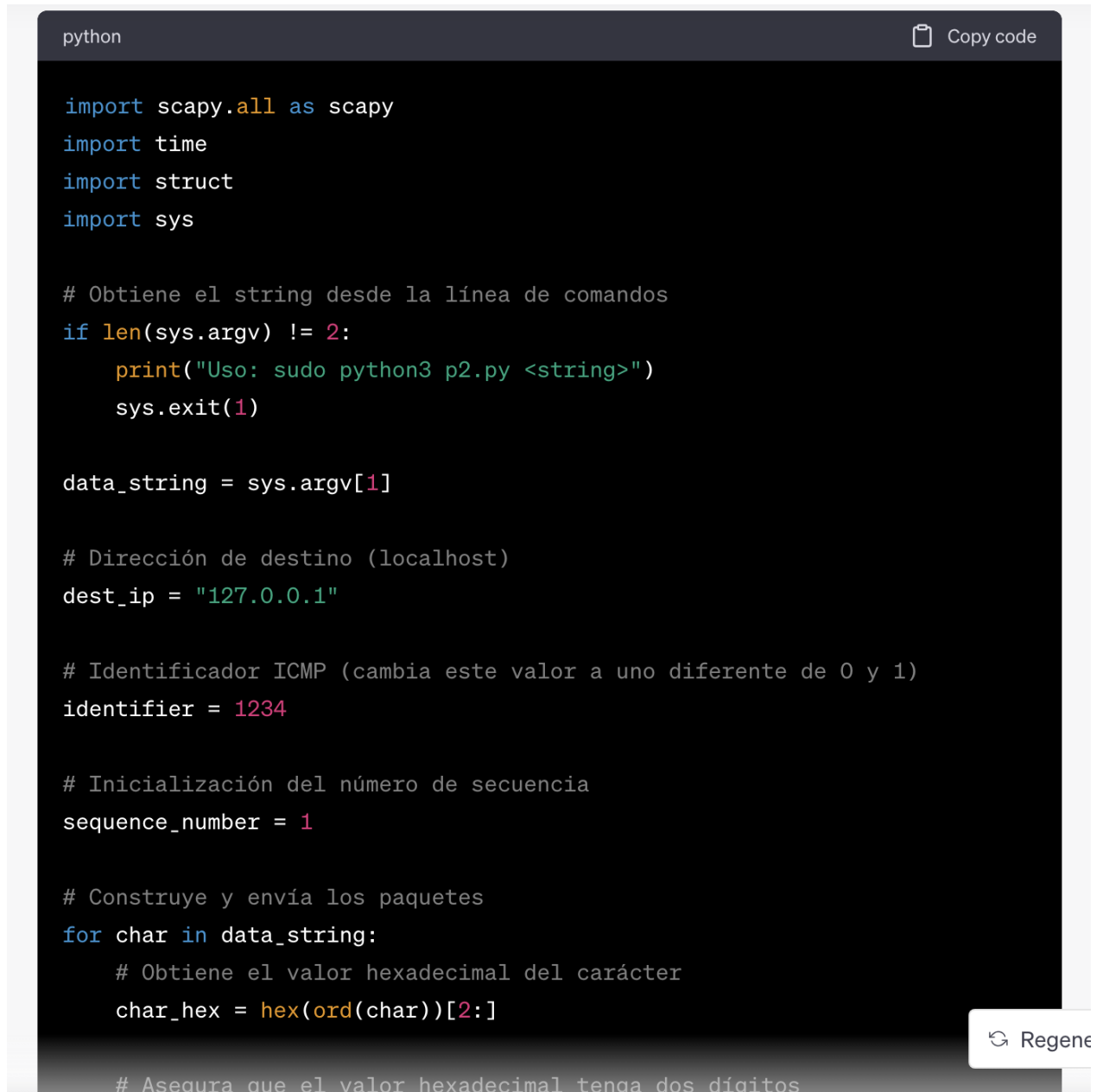


- enviar paquetes enviando un caracter de un string por paquete, por lo tanto, se deben enviar tantos paquetes como caracteres tenga el string
  - el campo type debe valer 8
  - el code debe valer 0
  - debe tener un checksum acorde a paquete icmp tipo ping ubuntu
  - debe tener en el campo identifier de icmp sea valido para ping ubuntu y distinto de 1 y 0
  - debe tener un sequence number incremental y valido, que parte desde 1 y aumenta en cada paquete
  - el campo data debe ser de 48 bytes, donde byte más significativo debe ser el caracter del string en hexadecimal y el resto debe ser así 00 01 00 00 00 00 00 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37
  - la direccion de destino debe ser 127.0.0.1
  - el timestamp debe corresponder al de la hora y fecha local al momento de correr el programa
- porfavor, gracias

Figura 5: Pregunta de actividad 2

La respuesta fue la siguiente:





```
python                                                                    Copy code

import scapy.all as scapy
import time
import struct
import sys

# Obtiene el string desde la línea de comandos
if len(sys.argv) != 2:
    print("Uso: sudo python3 p2.py <string>")
    sys.exit(1)

data_string = sys.argv[1]

# Dirección de destino (localhost)
dest_ip = "127.0.0.1"

# Identificador ICMP (cambia este valor a uno diferente de 0 y 1)
identifier = 1234

# Inicialización del número de secuencia
sequence_number = 1

# Construye y envía los paquetes
for char in data_string:
    # Obtiene el valor hexadecimal del carácter
    char_hex = hex(ord(char))[2:]

    # Asegura que el valor hexadecimal tenga dos dígitos
```

Figura 6: Respuesta de ChatGpT

```
# Asegura que el valor hexadecimal tenga dos dígitos
if len(char_hex) == 1:
    char_hex = "0" + char_hex

# Obtiene el timestamp actual en segundos
timestamp = int(time.time())

# Construye el payload de datos de 48 bytes
payload = bytes.fromhex(char_hex + "01000000000000101112131415161718191a")

# Construye el paquete ICMP
packet = scapy.IP(dst=dest_ip) / scapy.ICMP(type=8, code=0, id=identify)

# Agrega el timestamp al campo ICMP usando options
packet[scapy.ICMP].options = [(32, struct.pack(">I", timestamp))]

# Envía el paquete
scapy.send(packet)

# Incrementa el número de secuencia
sequence_number += 1

# Espera un breve momento (puedes ajustar esto según tus necesidades)
time.sleep(0.1)

print("Paquetes enviados exitosamente.")
```

 Regen

Figura 7: Respuesta de ChatGpT

Al momento de ejecutar el código en VS Code se ve lo siguiente:

```

bvle@bvle:~/Escritorio/lab-cripto$ sudo python3 p2.py "larycxpajorj" h bnpdarmjm nw anmnb"
[sudo] password for bvle:
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.

```

Figura 8: Ejecución de código

Para corroborar el envío se realizó una captura en Wireshark mientras se ejecutaba el programa, el resultado fue el siguiente:

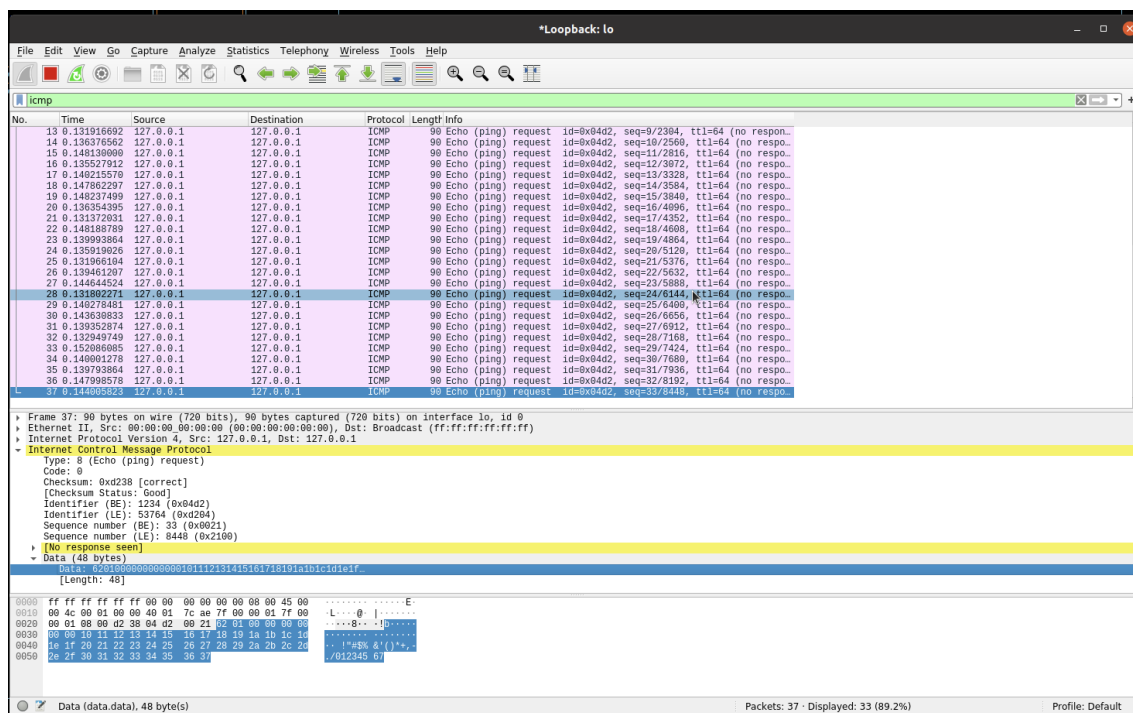


Figura 9: Captura de tráfico en Wireshark

El paquete seleccionado es el último paquete que envía la letra “b”, ese carácter se aprecia en la data que está seleccionada y aparece en el primer byte. Por lo tanto, se logra el envío de la palabra cifrada porque la letra “b” era la última de la cadena.

El archivo de la captura se llama *palabra\_encriptada.pcapng*. El archivo se encuentra en el repositorio.

### 3.3. Actividad 3

Para la actividad 3 se realizó la siguiente pregunta a ChatGPT: *“Hola, me gustaria un programa en python3 que lea los paquetes a partir de un archivo pcapng y forme el string que se envio en el programa anterior, a partir de ese string me gustaria formar todas las combinaciones posibles con algoritmo cesar. A partir de las combinaciones posibles, quiero saber cual es la palabra real en base a las letras más utilizadas en el lenguaje español de la a la z. Me gustaria que se mostraran enumeradas del 0 a 25 todas las palabras formadas a partir del algoritmo cesar y que se ponga en verde la palabra que es la mas probable. Por ultimo, me gustaria poner el nombre del archivo al momento de ejecutarlo asi: sudo python3 p3.py cesar.pcapng.”* La siguiente imagen corresponde a la pregunta hecha:



Hola, me gustaria un programa en python3 que lea los paquetes a partir de un archivo pcapng y forme el string que se envio en el programa anterior, a partir de ese string me gustaria formar todas las combinaciones posibles con algoritmo cesar. A partir de las combinaciones posibles, quiero saber cual es la palabra real en base a las letras más utilizadas en el lenguaje español de la a la z. Me gustaria que se mostraran enumeradas del 0 a 25 todas las palabras formadas a partir del algoritmo cesar y que se ponga en verde la palabra que es la mas probable. Por ultimo, me gustaria poner el nombre del archivo al momento de ejecutarlo asi: sudo python3 p3.py cesar.pcapng.

Figura 10: Pregunta a ChatGPT

El código respondido es el siguiente:

```
python Copy code

import scapy.all as scapy
import argparse
from colorama import Fore, Style # Importa la biblioteca colorama

# Función para descifrar una cadena utilizando el cifrado César
def cesar_decrypt(ciphertext, shift):
    decrypted_text = ""
    for char in ciphertext:
        if char.isalpha():
            shift_amount = shift % 26
            if char.islower():
                decrypted_char = chr(((ord(char) - ord('a')) - shift_amount))
            else:
                decrypted_char = chr(((ord(char) - ord('A')) - shift_amount))
            decrypted_text += decrypted_char
        else:
            decrypted_text += char
    return decrypted_text

# Parsea el nombre del archivo pcapng desde la línea de comandos
parser = argparse.ArgumentParser(description="Descifra un archivo pcapng usando el cifrado César")
parser.add_argument("pcap_file", help="Nombre del archivo pcapng")
args = parser.parse_args()

# Abre el archivo pcapng para leer paquetes
packets = scapy.rdpcap(args.pcap_file) Regenerar
```

Figura 11: Respuesta de ChatGPT

```
# Abre el archivo pcapng para leer paquetes
packets = scapy.rdpcap(args.pcap_file)

# Recorre los paquetes y extrae la carga útil de datos
data_string = ""
for packet in packets:
    if scapy.ICMP in packet and packet[scapy.ICMP].type == 8 and packet[scapy.ICMP].payload:
        # Obtiene la carga útil de datos del paquete ICMP
        payload = packet[scapy.ICMP].load

        # Extrae el valor hexadecimal del primer byte (carácter)
        hex_value = payload[0:2].hex()

        # Agrega el carácter al string
        data_string += bytes.fromhex(hex_value).decode('utf-8')

# Imprime el string recuperado
print("String recuperado del archivo pcapng:", data_string)

# Lista de frecuencias de letras en español (de la más frecuente a la menos)
frequencies = "aeoirlnstcudpgmbvhqjzxfykw"

best_combination = data_string
best_score = 0

# Genera todas las combinaciones posibles usando el cifrado César
for shift in range(26):
    decrypted_text = cesar_decrypt(data_string, shift)
    score = sum(decrypted_text.count(char) * (26 - frequencies.index(char)) for char in decrypted_text)
```

[Regen](#)

Figura 12: Respuesta de ChatGPT

```

# Genera todas las combinaciones posibles usando el cifrado César
for shift in range(26):
    decrypted_text = cesar_decrypt(data_string, shift)
    score = sum(decrypted_text.count(char) * (26 - frequencies.index(char)) i

# Compara el puntaje de la combinación actual con el mejor puntaje encon
if score > best_score:
    best_combination = decrypted_text
    best_score = score

# Genera todas las combinaciones posibles y muestra la palabra más probable
for shift in range(26):
    decrypted_text = cesar_decrypt(data_string, shift)

# Si la combinación actual es la más probable, imprímela en verde
if decrypted_text == best_combination:
    print(Fore.GREEN + f"{shift}: {decrypted_text}" + Style.RESET_ALL)
else:
    print(f"{shift}: {decrypted_text}")

```

Figura 13: Respuesta de ChatGPT

Al ejecutar el programa en VS Code se obtiene lo siguiente:

```

bvle@bvle:~/Escritorio/lab-cripto$ sudo python3 p3.py cesar.pcapng
String recuperado del archivo pcapng: larycxpajorj h bnpdarmjm nw anmnb
0: larycxpajorj h bnpdarmjm nw anmnb
1: kzqxbwozinqi g amoczqlil mv zmlma
2: jypwavyhmpf f zlnbypkhk lu yklkz
3: ixovzumxglog e ykmaxojgj kt xkjky
4: hwnuylwfknd d xjlnwnifi js wjiix
5: gvmtxskvejme c wikyvmeh ir vihiw
6: fulswrjudild b vhxulgdg hq uhghv
7: etkrvqitchkc a ugiwtkfcf gp tgfgu
8: dsjquphsbgjb z tfhvsjebe fo sfeft
9: criptografia y seguridad en redes
10: bqhosnfqzehz x rdftqhczc dm qdcdr
11: apgnrmepydyg w qcespgbyb cl pcbcq
12: zofmqldoxcfx v pbdrofata bk obabp
13: ynelpkcnwbew u oacqnezvz aj nazao
14: xmdkojbmadv t nzbpmdivy zi mzyzn
15: wlcjniauzcu s myaolcxux yh lyxym
16: vkbinhzytyb r lxznkbwtw xg kxwyl
17: ujahlgysxas q kwymjavsv wf jwvwl
18: tizgkfxirwzr p jvxlizuru ve ivuvj
19: shyfjehwqvyq o iuwkhytqt ud hutui
20: rgxeidvgpuxp n htvjgxspc tc gtsth
21: qfwdhucufotw m gsuiwfror sb fsrsg
22: pevcbgtensvn l frthevqnr ra ergrf
23: odubfasdmrum k eqsgdupmp qz dqpqe
24: nctaezrclqtl j dprfctolo py cpopd
25: mbszdyqbkpsk i coqebnskn ox bonoc
bvle@bvle:~/Escritorio/lab-cripto$

```

Figura 14: Ejecución de código

El código entregado por ChatGPT cumple con lo solicitado, siendo encontrada la frase

original y escrita en verde tal como se pidió.

## Conclusiones y comentarios

Durante el desarrollo del laboratorio, se han identificado ciertos desafíos al interactuar con ChatGPT. Uno de los principales problemas que surgieron es la necesidad de ser extremadamente específico al formular preguntas. Es esencial ser claro y directo al hacer una consulta, sin dejar espacio para la ambigüedad, para recibir una respuesta competente. En caso contrario, la respuesta proporcionada puede resultar insuficiente. Para abordar este problema en la actividad 2, opté por crear un esquema detallado de lo que necesitaba preguntar, con el fin de no omitir ningún detalle relevante.

Además, me llamó la atención la variabilidad en las respuestas de ChatGPT, incluso cuando se plantea la misma pregunta de manera idéntica en varias ocasiones. Esto genera confusión y dudas sobre si la primera respuesta es correcta o si es necesario insistir en la misma pregunta.

Otra observación importante es que en ocasiones, ChatGPT parece olvidar algunos detalles de la pregunta original, lo que requiere que repitamos la información omitida para obtener una respuesta completa y precisa.

Otro problema que encontré, es que se pierde mucho tiempo porque a veces la generación de respuesta no concluye a nada útil, y por cada solución hay que fijarse bien si está bueno o no. Si no está buena hay que seguir generando respuestas agregando más información en algunos casos.

En resumen, este laboratorio ha contribuido significativamente a nuestro conocimiento sobre ICMP, en particular en lo que respecta a los paquetes de tipo PING, y también nos ha proporcionado información valiosa sobre el uso de ChatGPT como una herramienta de consulta. Si bien ChatGPT puede ser útil para orientarnos en ciertos momentos, es importante recordar que no debe considerarse como una fuente de documentación oficial.