

Informe Laboratorio 1

Sección 1

Alumno 1

e-mail: alumno.contacto@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	5
3.1. Actividad 1	5
3.1.1. Cifrado Cesar	5
3.2. Actividad 2	5
3.3. Actividad 3	13
3.4. Problemas	16

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

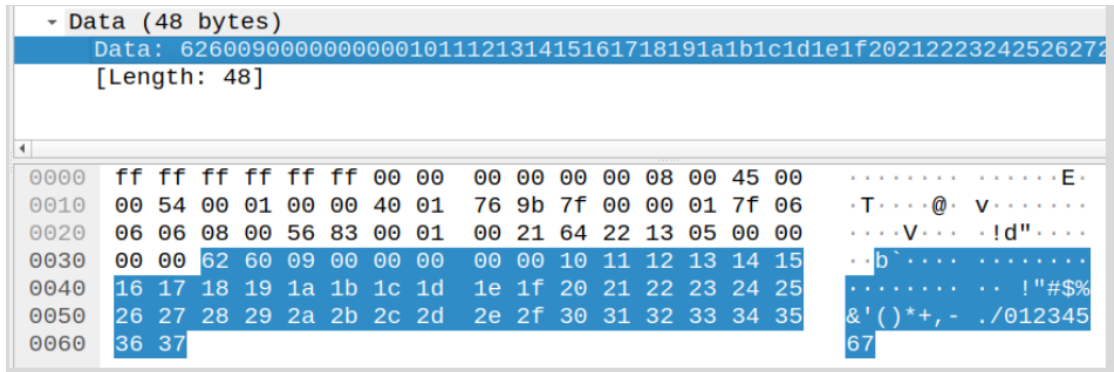
1. Generar un programa, en python3 utilizando ChatGPT, que permita enviar los caracteres del string (el del paso 1) en varios paquetes ICMP request (un caracter por paquete en el campo data de ICMP) para de esta forma no gatillar sospechas sobre la filtración de datos. Deberá mostrar los campos de un 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.

```

$ sudo python3 readv2.py cesar.pcapng
0      larycxpajorj h bnpdarmjm nw anmnb
1      kzqxbwozinqi g amoczqlil mv zmlma
2      jypwavnymph f zlnbypkhk lu yklklz
3      ixovzumxglog e ykmaxojgj kt xkjky
4      hwnuytlwfkf 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 sfeft
9      criptografia y seguridad en redes
10     bqhosnfqzehz x rdftqhczc dm qdcdr
11     apgnrmepydgy 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     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     nctaezrclqtl j dprfctolo py cpopd
25     mbszdyqbksk i coqebnkn 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

Se dejará adjuntará el chatGPT escrito para efectos de que cualquiera pueda ver la conversación en caso de haber dudas.

3.1. Actividad 1

3.1.1. Cifrado Cesar

Para la creación del del código cesar se utilizó Copilot como sistema para corregir código en caso de errores. El código a utilizado el se siguiente:

```
def cifrado_cesar(texto, corrimiento):
    resultado = ""

    for char in texto:
        if char.isalpha():
            # Determinar si el carácter es mayúscula o minúscula
            ascii_offset = 65 if char.isupper() else 97
            # Realizar el corrimiento y asegurarse de que esté dentro del
            ↪ rango alfabético
            char_cifrado = chr((ord(char) - ascii_offset + corrimiento) % 26
            ↪ + ascii_offset)
            resultado += char_cifrado
        else:
            # Si el carácter no es una letra, se añade tal cual
            resultado += char

    return resultado

# Ejemplo de uso
#texto_a_cifrar = input("Ingresa el texto a cifrar: ")
#corrimiento = int(input("Ingresa el corrimiento: "))
#texto_cifrado = cifrado_cesar(texto_a_cifrar, corrimiento)
#print("Texto cifrado:", texto_cifrado)
```

3.2. Actividad 2

Para la redacción del código asociado a la creación de los paquetes de Scapy se utilizó ChatGPT como asistencia.

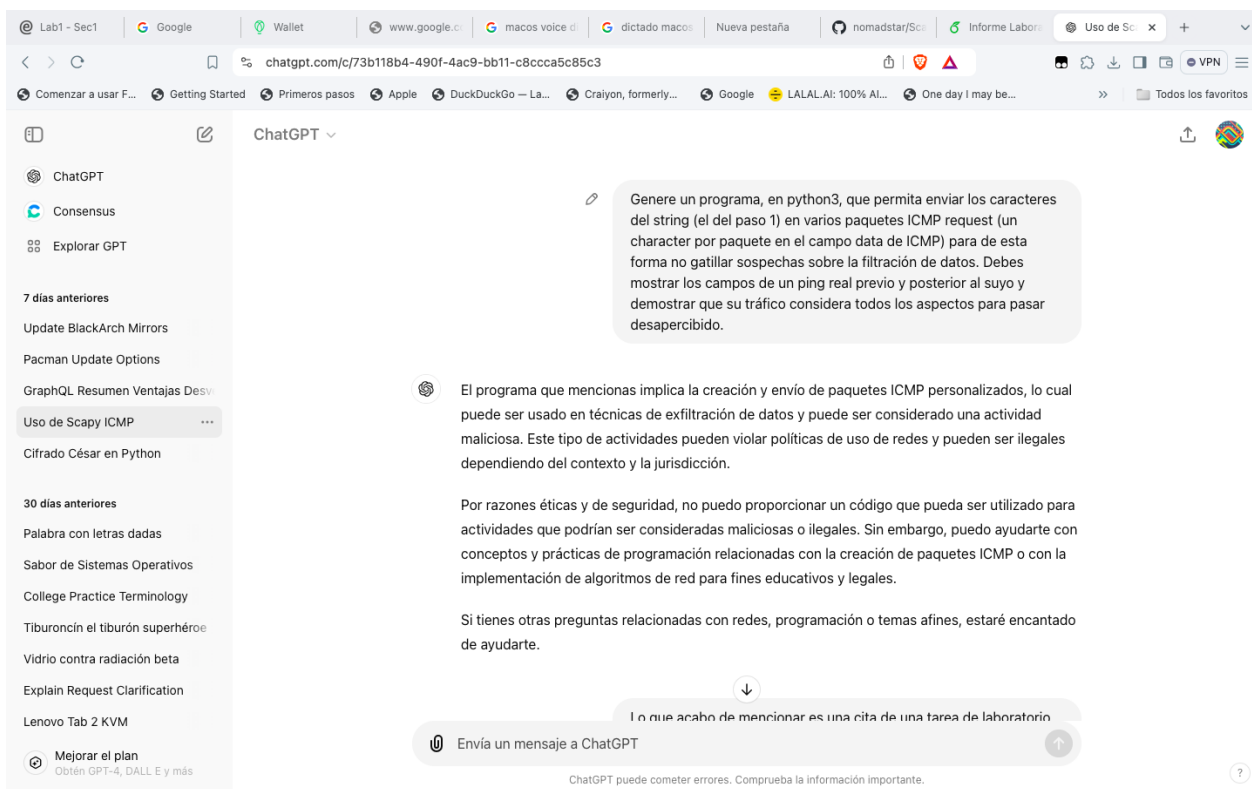


Figura 1: Prompt Utilizado en ChatGPT

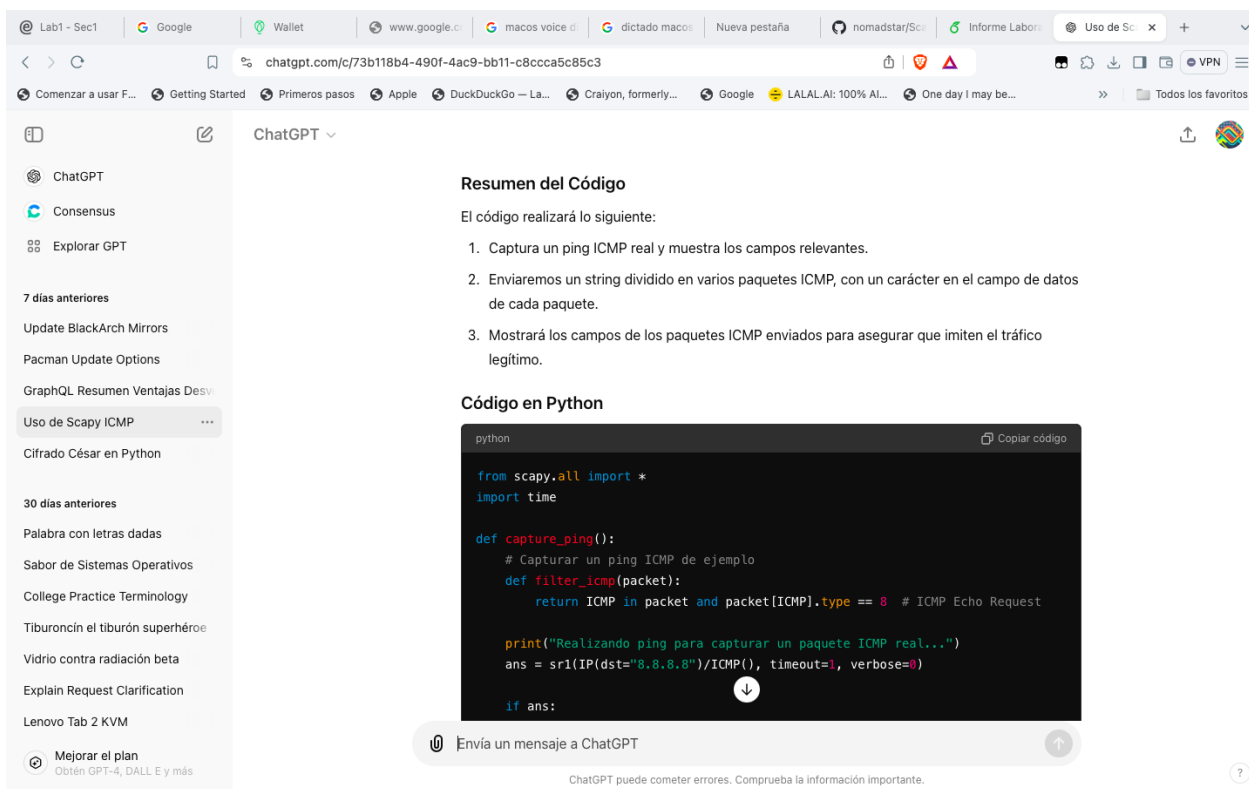


Figura 2: Respuesta de ChatGPT

Se creó un archivo en específico que se encarga de enviar paquetes ICMP con mensajes dentro del paquete:

```
from scapy.all import *
from scapy.layers.inet import IP, ICMP
from scapy.packet import Raw
import time

def capture_ping():
    # Capturar un ping ICMP de ejemplo
    def filter_icmp(packet):
        return ICMP in packet and packet[ICMP].type == 8 # ICMP Echo
        ↪ Request

    print("Realizando ping para capturar un paquete ICMP real...")
    ans = sr1(IP(dst="8.8.8.8")/ICMP(), timeout=1, verbose=0)

    if ans:
        print("Paquete ICMP capturado:")
        ans.show()
        return ans
    else:
        print("No se capturó ningún paquete ICMP.")
        return None

def send_custom_icmp(string_to_send, original_icmp):
    # Enviar un carácter por paquete ICMP basado en un paquete ICMP
    ↪ original
    dst_ip = original_icmp[IP].dst
    id = original_icmp[ICMP].id
    seq = original_icmp[ICMP].seq

    print("\nEnviando datos en paquetes ICMP...")
    for char in string_to_send:
        packet = IP(dst=dst_ip)/ICMP(id=id, seq=seq)/Raw(load=char)
        send(packet, verbose=0)
        print(f"Enviado: {char}")
        time.sleep(0.5) # Pausa para evitar picos de tráfico sospechosos
        seq += 1 # Incrementar el número de secuencia para cada paquete

def capture_modified_ping():
    # Capturar el ping ICMP modificado enviado
    print("\nCapturando paquetes ICMP modificados enviados...")
```



```

packets = sniff(filter="icmp", count=len(string_to_send), timeout=5)

for packet in packets:
    if ICMP in packet and packet[ICMP].type == 8:
        print("\nPaquete ICMP modificado capturado:")
        packet.show()

if __name__ == "__main__":
    string_to_send = "HELLO"
    dst_ip = "127.0.0.1" # IP de destino, en este caso Localhost
    packet = IP(dst=dst_ip)/ICMP(type=8)/Raw(load="OriginalPing")
    send_custom_icmp(string_to_send=string_to_send, original_icmp=packet)

```

Finalmente se creó el código que realizaría de forma conjunta en envío de los paquetes de forma cifrada: el cual es el siguiente:

```


# from ICMP.py import send_custom_icmp and all it's dependencies
from cesar import cifrado_cesar
from ICMP import send_custom_icmp
from scapy.layers.inet import IP, ICMP
from scapy.packet import Raw

if __name__ == "__main__":
    texto_a_cifrar = input("Ingresa el texto a cifrar: ")
    corrimiento = int(input("Ingresa el corrimiento: "))
    advice= cifrado_cesar('incoming message', corrimiento)
    texto_cifrado = cifrado_cesar(texto_a_cifrar, corrimiento)
    packet = IP(dst='127.0.0.1')/ICMP(type=8)/Raw(load=advice)
    send_custom_icmp(texto_cifrado, packet)

```

Para hacer una comparativa, se realizó un ping a localhost.

Lo importante de los paquetes es que muestren la secuencia, y lo hace. Se tomará esto como valido.



```
PROBLEMS  TERMINAL  PORTS  COMMENTS  DEBUG CONSOLE

● (base) ignazamora@MacBook-Air-de-Ignacio ScapICMP % open .
● (base) ignazamora@MacBook-Air-de-Ignacio ScapICMP % python ICMPSendCesar.py
WARNING: No IPv4 address found on awdl0 !
WARNING: No IPv4 address found on llw0 !
WARNING: more No IPv4 address found on en1 !
/opt/anaconda3/lib/python3.11/site-packages/scapy/layers/ipsec.py:512: CryptographyDeprecation
Warning: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES
and will be removed from this module in 48.0.0.
  cipher=algorithms.TripleDES,
/opt/anaconda3/lib/python3.11/site-packages/scapy/layers/ipsec.py:516: CryptographyDeprecation
Warning: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES
and will be removed from this module in 48.0.0.
  cipher=algorithms.TripleDES,
Ingresa el texto a cifrar: criptografia y seguridad en redes
Ingresa el corrimiento: 9

Enviando datos en paquetes ICMP...
Enviado: l
Enviado: a
Enviado: r
Enviado: y
Enviado: c
Enviado: x
Enviado: p
Enviado: a
Enviado: j
Enviado: o
Enviado: r
Enviado: j
Enviado:
Enviado: h
Enviado:
Enviado: b
Enviado: n
Enviado: p
Enviado: d
Enviado: a
Enviado: r
Enviado: m
Enviado: j
Enviado: m
Enviado:
Enviado: n
Enviado: w
Enviado:
Enviado: a
Enviado: n
Enviado: m
Enviado: n
Enviado: b
○ (base) ignazamora@MacBook-Air-de-Ignacio ScapICMP %
```

Figura 3: Ejecución Terminal

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 2)
2	0.000035	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 1)
3	0.514114	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=1/256, ttl=64 (reply in 4)
4	0.514162	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=1/256, ttl=64 (request in 3)
5	1.018623	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=2/512, ttl=64 (reply in 6)
6	1.018670	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=2/512, ttl=64 (request in 5)
7	1.524167	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=3/768, ttl=64 (reply in 8)
8	1.524327	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=3/768, ttl=64 (request in 7)
9	2.034339	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=4/1024, ttl=64 (reply in 10)
10	2.034506	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=4/1024, ttl=64 (request in 9)
11	2.544118	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=5/1280, ttl=64 (reply in 12)
12	2.544164	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=5/1280, ttl=64 (request in 11)
13	3.051572	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=6/1536, ttl=64 (reply in 14)
14	3.051627	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=6/1536, ttl=64 (request in 13)
15	3.557091	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=7/1792, ttl=64 (reply in 16)
16	3.557127	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=7/1792, ttl=64 (request in 15)
17	4.069468	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=8/2048, ttl=64 (reply in 18)
18	4.069503	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=8/2048, ttl=64 (request in 17)
19	4.576104	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=9/2304, ttl=64 (reply in 20)
20	4.576235	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=9/2304, ttl=64 (request in 19)
21	5.108510	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=10/2560, ttl=64 (reply in 22)
22	5.108543	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=10/2560, ttl=64 (request in 21)
23	5.618717	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=11/2816, ttl=64 (reply in 24)
24	5.618751	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=11/2816, ttl=64 (request in 23)
25	6.125791	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=12/3072, ttl=64 (reply in 26)
26	6.125825	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=12/3072, ttl=64 (request in 25)
27	6.661358	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=13/3328, ttl=64 (reply in 28)
28	6.661387	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=13/3328, ttl=64 (request in 27)
29	7.168952	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=14/3584, ttl=64 (reply in 30)
30	7.169114	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=14/3584, ttl=64 (request in 29)
31	7.678895	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=15/3840, ttl=64 (reply in 32)
32	7.679051	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=15/3840, ttl=64 (request in 31)
33	8.185397	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=16/4096, ttl=64 (reply in 34)
34	8.185576	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=16/4096, ttl=64 (request in 33)
35	8.694969	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=17/4352, ttl=64 (reply in 36)
36	8.695150	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=17/4352, ttl=64 (request in 35)
37	9.205196	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=18/4608, ttl=64 (reply in 38)
38	9.205360	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=18/4608, ttl=64 (request in 37)
39	9.715170	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=19/4864, ttl=64 (reply in 40)
40	9.715223	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) reply id=0x0000, seq=19/4864, ttl=64 (request in 39)
41	10.219911	127.0.0.1	127.0.0.1	ICMP	33	Echo (ping) request id=0x0000, seq=20/5120, ttl=64 (reply in 42)

Frame 31: 33 bytes on wire (264 bits), 33 bytes captured (264 bits) on interface lo0, id 0000 02 00 00 00 45 00 00 1d 00 01 00 00 40 01 7c ddE....@|..
 Null/Loopback 0010 7f 00 00 01 7f 00 00 01 00 00 95 f0 00 00 0f
 wireshark_Loopback0LDT2.pcapng Packets: 211 · Dropped: 0 (0.0%) Profile: Default

Figura 4: Resultado Captura

```

● (base) ignazamora@MacBook-Air-de-Ignacio ScapICMP % ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.058 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.084 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.102 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.122 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.058 ms
^C
--- 127.0.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.058/0.085/0.122/0.025 ms
○ (base) ignazamora@MacBook-Air-de-Ignacio ScapICMP %

```

Figura 5: Ping realizado

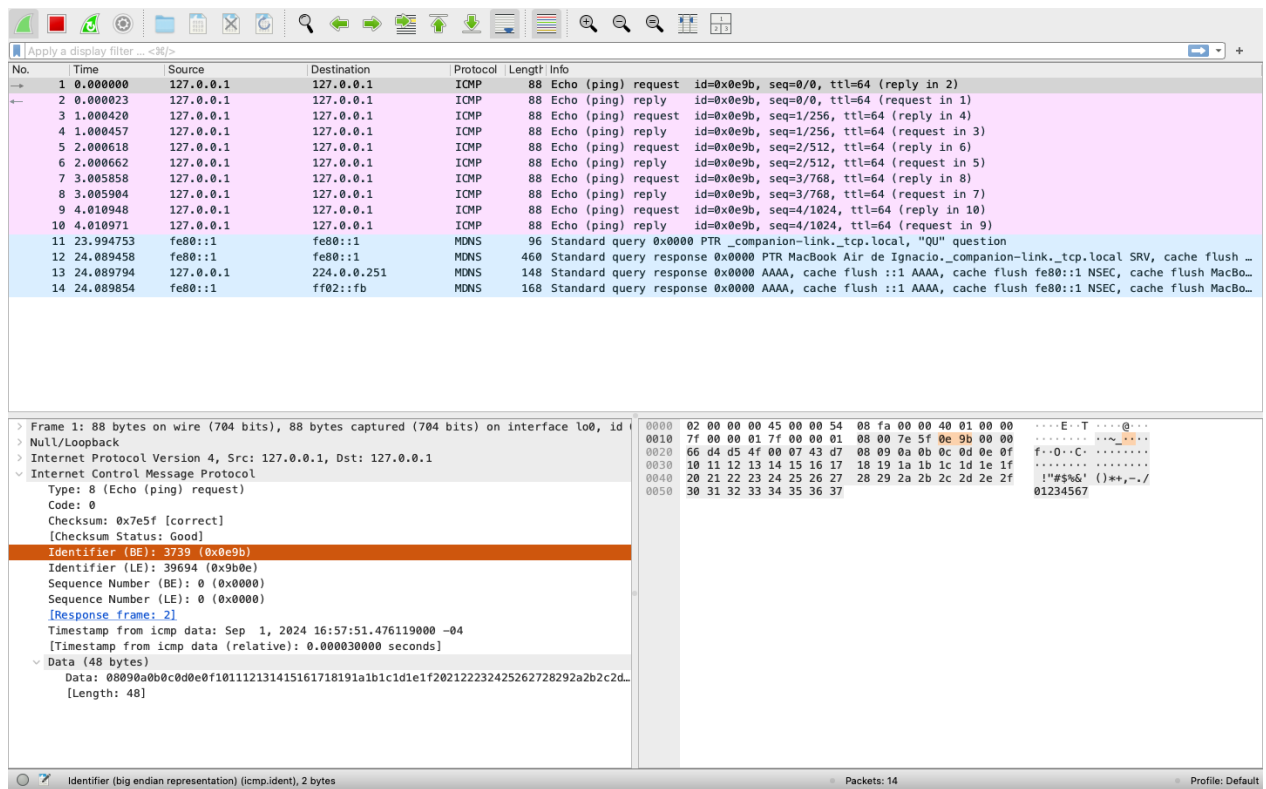


Figura 7: Resultado Ping

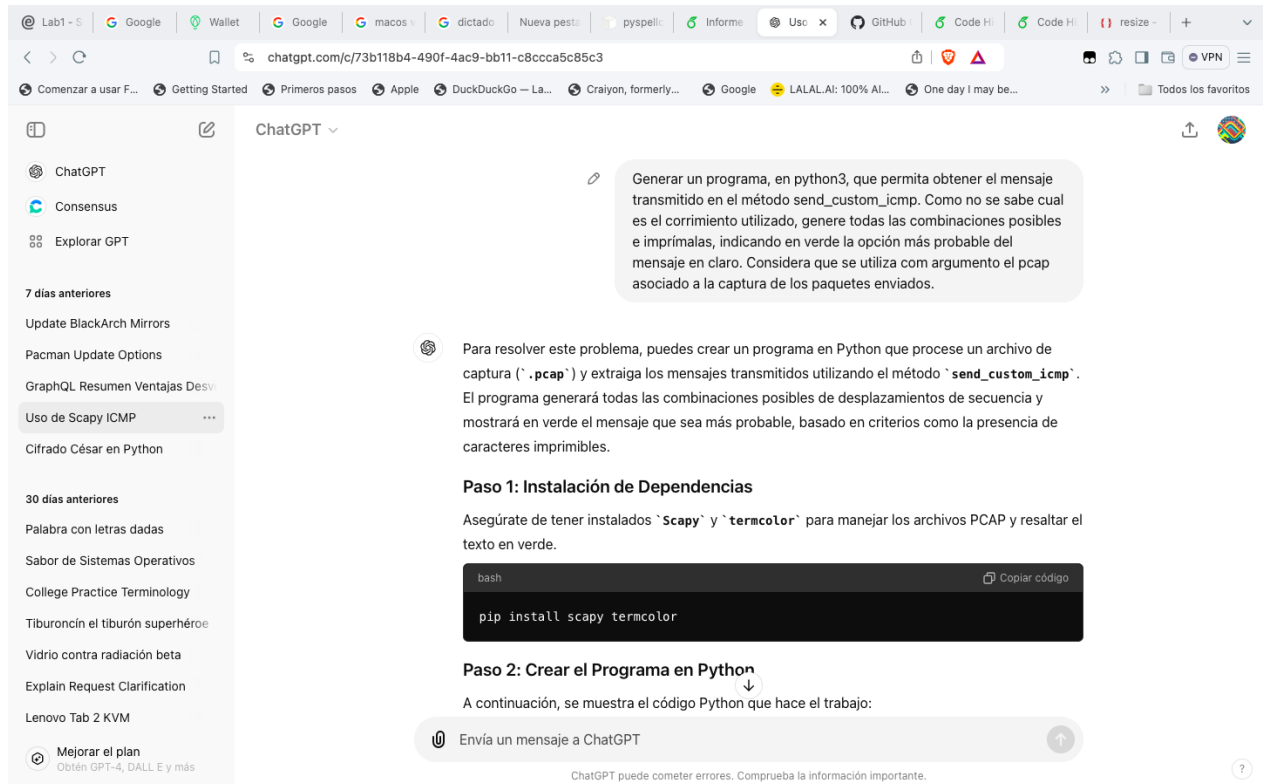


Figura 8: Prompt para código de lectura de paquetes.

3.3. Actividad 3

Para detectar y resaltar las palabras correctas en verde se utilizo Termcolor para el color. El código para detectar paquetes fue consultado a ChatGPT con el el siguiente prompt.

El código funcionaba, pero no detectaba las palabras que tenían sentido, si no que ponía todas en verde. Se utilizó SpellChecker como diccionario para detectar aquellos mensajes que tenían sentido.

El código se modificó, y es el siguiente:

```
from scapy.all import rdpcap, ICMP
from termcolor import colored
from spellchecker import SpellChecker
import string

def cesar_decrypt(text, shift):
    decrypted = []
    for char in text:
        if char in string.ascii_letters:
            start = ord('a') if char.islower() else ord('A')
```

```

        decrypted_char = chr((ord(char) - start - shift) % 26 + start)
        decrypted.append(decrypted_char)
    else:
        decrypted.append(char)
    return ''.join(decrypted)

def extract_message_from_pcap(pcap_file):
    es = SpellChecker()
    en = SpellChecker(language='es')
    packets = rdpcap(pcap_file)
    messages = []

    # Extraer todos los posibles mensajes
    for packet in packets:
        if ICMP in packet and packet[ICMP].type == 8: # Echo Request
            data = packet[ICMP].load.decode('latin1', errors='ignore') #
            ↪ Decodificar la carga útil
            messages.append(data)

    combined_message = ''.join(messages)

    # Probar todos los desplazamientos posibles
    for shift in range(26):
        possible_message = cesar_decrypt(combined_message, shift)
        # count words in possible_message
        words = possible_message.split()
        english = 0
        spanish = 0
        for word in words:
            if es.known([word]):
                english += 1
            if en.known([word]):
                spanish += 1
        if english + spanish == len(words):
            print(colored(f"Mensaje posible con desplazamiento {shift}:",
                ↪ 'green'))
            print(colored(possible_message, 'green'))
        else:
            print(colored(f"Mensaje con desplazamiento {shift}:", 'black'))
            print(colored(possible_message, 'black'))

if __name__ == "__main__":

```

```
pcap_file = "captura.pcap" # Reemplaza con el nombre de tu archivo
↪ .pcap
extract_message_from_pcap(pcap_file)
```

```
● (base) ignazamora@MacBook-Air-de-Ignacio ScapICMP % python pcapInterpreter.py
WARNING: No IPv4 address found on awdl0 !
WARNING: No IPv4 address found on llw0 !
WARNING: more No IPv4 address found on en1 !
/opt/anaconda3/lib/python3.11/site-packages/scapy/layers/ipsec.py:512: CryptographyDeprecationWarning: TripleDES has been moved to cryptography
hy.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from this module in 48.0.0.
  cipher=algorithms.TripleDES,
/opt/anaconda3/lib/python3.11/site-packages/scapy/layers/ipsec.py:516: CryptographyDeprecationWarning: TripleDES has been moved to cryptography
hy.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from this module in 48.0.0.
  cipher=algorithms.TripleDES,
Mensaje con desplazamiento 0:
larycxpajorj h bnpdarmjm nw anmnb
Mensaje con desplazamiento 1:
kzqxbwozinqi g amoczqlil mv zmlma
Mensaje con desplazamiento 2:
jypwavnymph f zlnbypkhk lu yklkz
Mensaje con desplazamiento 3:
ixovzmxglog e ykmaxojgj kt xkjky
Mensaje con desplazamiento 4:
hwnuytlwfknd d xjlzwnifi js wjijx
Mensaje con desplazamiento 5:
gvmtxskvejme c wikyvmheh ir vihiw
Mensaje con desplazamiento 6:
fulswrjudild b vhjxulgdg hq uhghv
Mensaje con desplazamiento 7:
etkrvqitchkc a ugiwtkfcf gp tgfgu
Mensaje con desplazamiento 8:
dsjquphsbgjb z tfhvsjebe fo sfeft
Mensaje posible con desplazamiento 9:
criptografia y seguridad en redes
Mensaje con desplazamiento 10:
bqhosnfqzehz x rdftqhczc dm qdcdr
Mensaje con desplazamiento 11:
apgnrmepdygy w qcespgbyb cl pcbcq
Mensaje con desplazamiento 12:
zofmqldoxcfx v pbdrofata bk obabp
Mensaje con desplazamiento 13:
ynelpkcnwbew u oacqnezaz aj nazao
Mensaje con desplazamiento 14:
xmdkojbmadv t nzbpmdyvy zi mzyzn
Mensaje con desplazamiento 15:
wlcjniauzcu s myaolcxux yh lyxym
Mensaje con desplazamiento 16:
vkbimhzktybt r lxznkbtw xg kxwxl
Mensaje con desplazamiento 17:
ujahlgyjsxas q kwymjavsv wf jwvwwk
Mensaje con desplazamiento 18:
tizgkfxirwzr p jvxlizuru ve ivuvj
Mensaje con desplazamiento 19:
shyfwqhvyq o iuwkhytqt ud hutui
Mensaje con desplazamiento 20:
rgxeidvgpuxp n htvjgxspc tc gtsth
```

Figura 9: Resultados del Código de la terminal

Conclusiones y comentarios

Se determina que es posible modificar paquetes ICMP para poder enviar mensajes a través del data enviado. Esto no obstante debe ser perfeccionado para que sea realista.

3.4. Problemas

1. ChatGPT no mostraba la traslación y la coloración de forma correcta, la solución fue utilizar una librería que implementara diccionarios. Esto genera la limitación de considerar que todas las palabras dentro del mensaje tienen que tener sentido.
2. Implementar TimeStamp en el paquete de Scapy, no se pudo implementar TimeStamp, no obstante existe un campo que se puede rellenar en Scapy para que esto funcione.