

Informe Laboratorio 1

Sección 1

Fernando Cabrera Legue
fernando.cabrera1@mail.udp.cl

Agosto de 2025

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	5
3.3. Actividad 3	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 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.

```
└─$ ~/Desktop $ 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.

```
└─$ ~/Desktop $ 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      jypwavnyhmp h f zlnbypkhh lu ylkhl
3      ixovzumxglog e ykmaxojgj kt xkjky
4      hwnuytlwfknd d xjlnwnifi js wjiyx
5      gvmtxskvejme c wikyvmheh ir vihiw
6      fulswrjudild b vhxulgdg hq uhghv
7      etkrvqitchkc a ugiwtkfcf gp tgfgu
8      dsjquphsbgjb z tfhvsjebe fo sfefl
9      criptografia y seguridad en redes
10     bqhosnfqzehz x rdftqhczc dm qdcdr
11     apgnrmepdygy w qcespgbyb cl pcbbc
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 htvjgxsp s tc gtsth
21     qfwdhcufotwo m gsuifwr or sb ffsrg
22     pevcbgtensvn l frthevq nq ra erqrf
23     odubfasdmrum k eqsgdupmp qz dqpqe
24     nctaezrcqltl j dprfctolo py cpopd
25     mbszdyqbksk i coqeb snkn ox bonoc

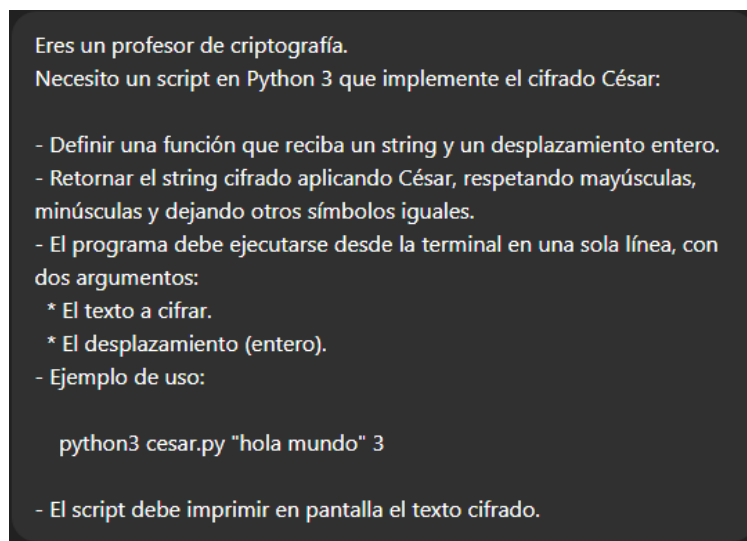
```

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

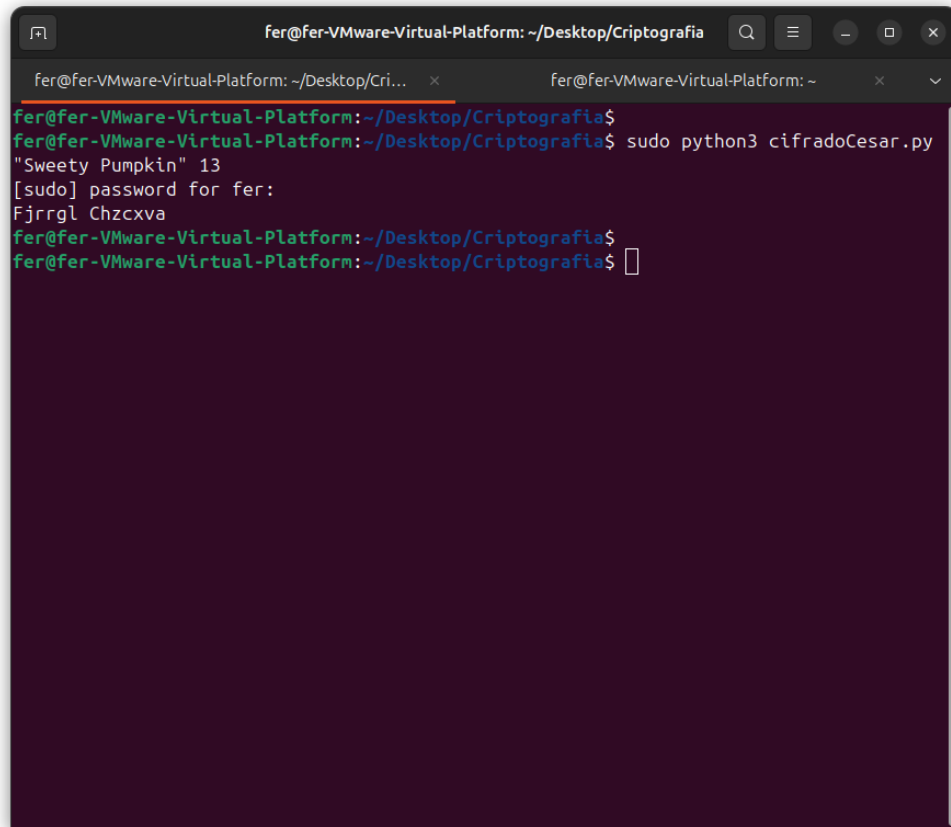
3.1. Actividad 1

Para esta actividad debemos crear un programa en `python3` el cual logre cifrar palabras en el clásico cifrado César. Para esto, utilizaremos a la herramienta de inteligencia artificial ChatGPT para que nos ayude durante todo el laboratorio.

A screenshot of a ChatGPT interface with a dark background and light text. The text is a prompt for a Python script to implement a Caesar cipher. It includes instructions on function definition, argument handling, and a specific command to run the script.

```
Eres un profesor de criptografía.  
Necesito un script en Python 3 que implemente el cifrado César:  
  
- Definir una función que reciba un string y un desplazamiento entero.  
- Retornar el string cifrado aplicando César, respetando mayúsculas,  
  minúsculas y dejando otros símbolos iguales.  
- El programa debe ejecutarse desde la terminal en una sola línea, con  
  dos argumentos:  
  * El texto a cifrar.  
  * El desplazamiento (entero).  
- Ejemplo de uso:  
  
  python3 cesar.py "hola mundo" 3  
  
- El script debe imprimir en pantalla el texto cifrado.
```

Figura 1: Prompt para generar el código del cifrado César.

A screenshot of a terminal window titled 'fer@fer-VMware-Virtual-Platform: ~/Desktop/Criptografia'. The terminal shows the following commands and output:

```
fer@fer-VMware-Virtual-Platform: ~/Desktop/Criptografia$  
fer@fer-VMware-Virtual-Platform: ~/Desktop/Criptografia$ sudo python3 cifradoCesar.py  
"Sweety Pumpkin" 13  
[sudo] password for fer:  
Fjjrgl Chzcxva  
fer@fer-VMware-Virtual-Platform: ~/Desktop/Criptografia$  
fer@fer-VMware-Virtual-Platform: ~/Desktop/Criptografia$
```

Figura 2: Caption

Tras esto, en la terminal podemos ejecutar el programa con una palabra a cifrar, en este caso la palabra es “**Sweety Pumpkin**”, con un desplazamiento de 13 caracteres. Por lo tanto, la palabra cifrada es “**Fjjrgl Chzcxva**”.

3.2. Actividad 2

En esta actividad se pide fragmentar cada uno de los caracteres de la palabra cifrada anteriormente, luego que sean enviados en paquetes ICMP `Request` individuales, y con la ayuda de `Wireshark` registrar el tráfico y así analizar si el mensaje se transmitio correctamente. Nuevamente requerimos de la ayuda de `ChatGPT`, el cual nos puede proporcionar un código en `python3` el cual ejecute todo lo mencionado.

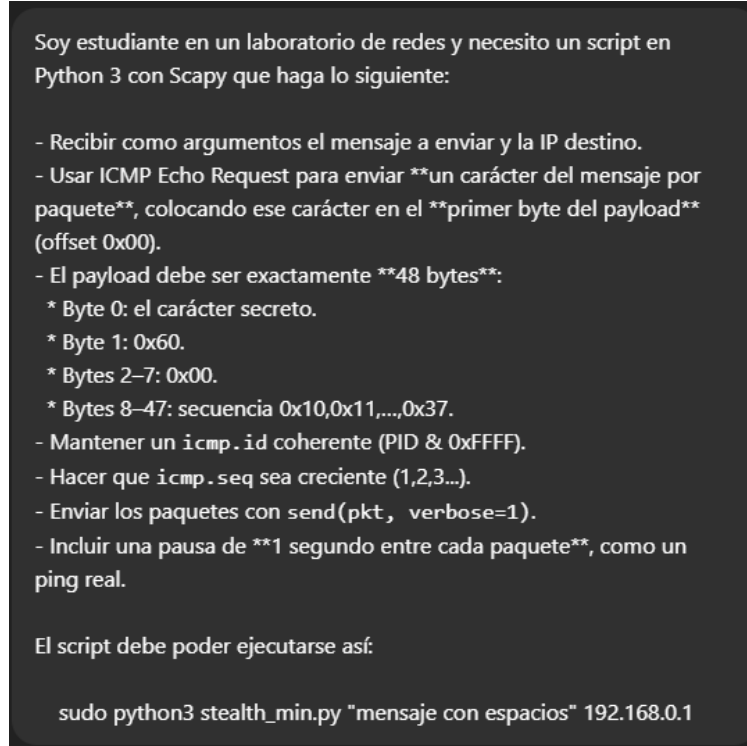
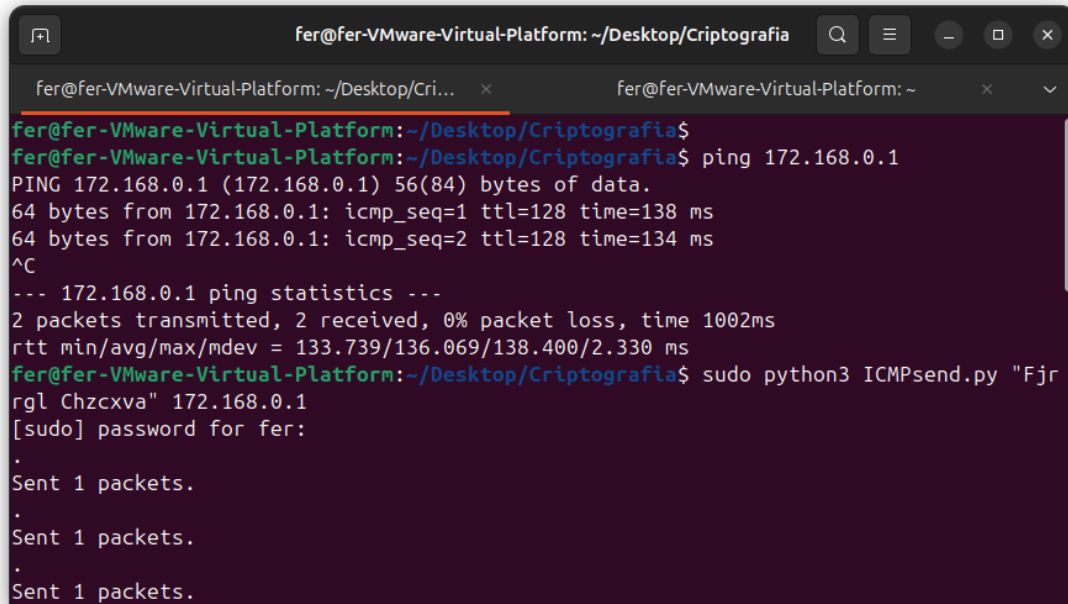


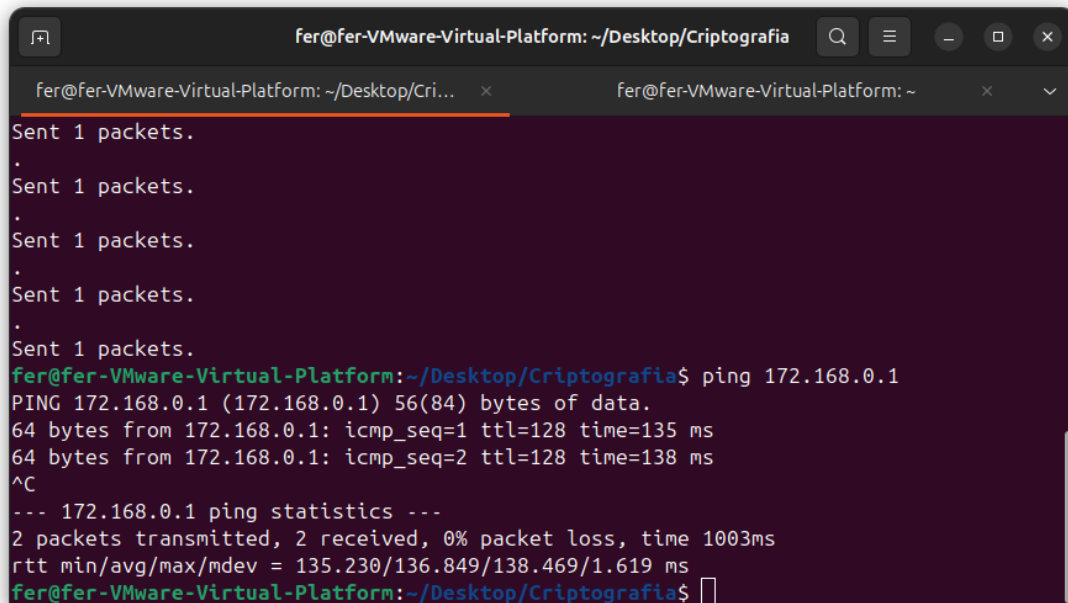
Figura 3: Prompt para el código del modo Stealth.

Primero ejecutamos *Wireshark*, realizamos la captura *any* y aplicamos el filtro *icmp* para ver solo los paquetes que nos interesan.



```
fer@fer-VMware-Virtual-Platform: ~/Desktop/Criptografia
fer@fer-VMware-Virtual-Platform:~/Desktop/Criptografia$ ping 172.168.0.1
PING 172.168.0.1 (172.168.0.1) 56(84) bytes of data.
64 bytes from 172.168.0.1: icmp_seq=1 ttl=128 time=138 ms
64 bytes from 172.168.0.1: icmp_seq=2 ttl=128 time=134 ms
^C
--- 172.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 133.739/136.069/138.400/2.330 ms
fer@fer-VMware-Virtual-Platform:~/Desktop/Criptografia$ sudo python3 ICMPsend.py "Fjrgl Chzcva" 172.168.0.1
[sudo] password for fer:
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

Figura 4: Ping inicial y envio de paquetes de letras iniciales.



```
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
fer@fer-VMware-Virtual-Platform:~/Desktop/Criptografia$ ping 172.168.0.1
PING 172.168.0.1 (172.168.0.1) 56(84) bytes of data.
64 bytes from 172.168.0.1: icmp_seq=1 ttl=128 time=135 ms
64 bytes from 172.168.0.1: icmp_seq=2 ttl=128 time=138 ms
^C
--- 172.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 135.230/136.849/138.469/1.619 ms
fer@fer-VMware-Virtual-Platform:~/Desktop/Criptografia$
```

Figura 5: Ping final y envio de paquetes de letras finales.

Luego con un *ping* a la IP: 168.172.0.1 podemos ver la estructura de un paquete ICMP Request, tras esto, ejecutamos el código que reciben como parámetros la palabra cifrada y la

IP de destino, para luego esta ser fragmentada y enviada en cada paquete, tal como enseña la figura 4. Vemos los últimos paquetes llegar y volvemos a hacer un *ping* a la IP: 172.168.0.1, para ver si el tráfico ha cambiado o se mantiene igual forma, como se ve en la figura 5, además guardamos la captura para hacer el análisis correspondiente.

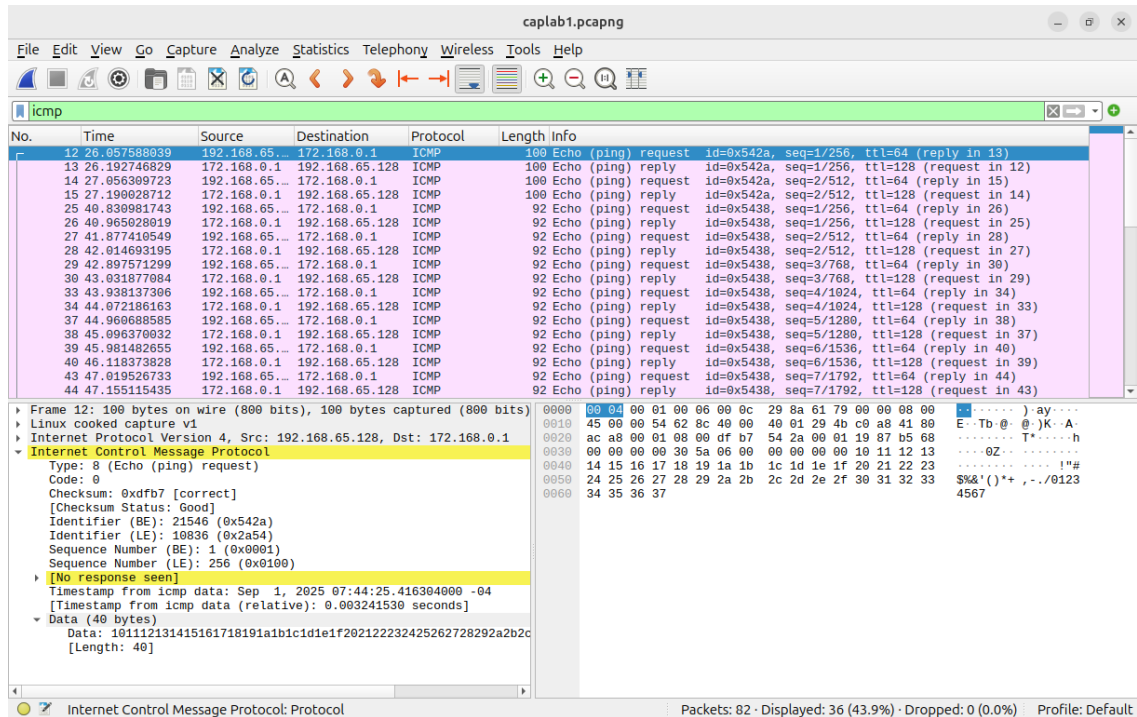


Figura 6: Estructura paquete ICMP Request inicial.

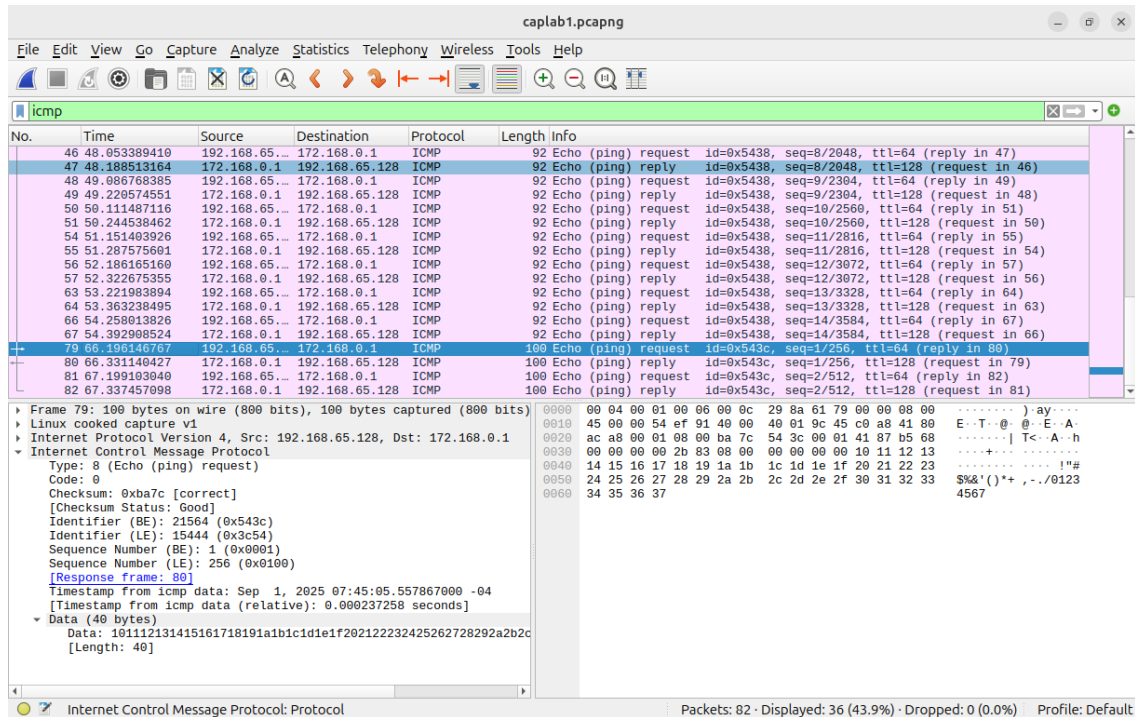


Figura 7: Estructura paquete ICMP Request final.

Notar que un paquete *ping* ICMP Request tiene 100 bytes totales de longitud, un payload de 40 bytes y una frecuencia entre cada mensaje de 1 segundo. Y para un paquete ICMP Request construido por nosotros tiene 92 bytes totales de longitud, un payload de 48 bytes y una frecuencia entre cada mensaje de 1 segundo, ver figura 6 y 7.

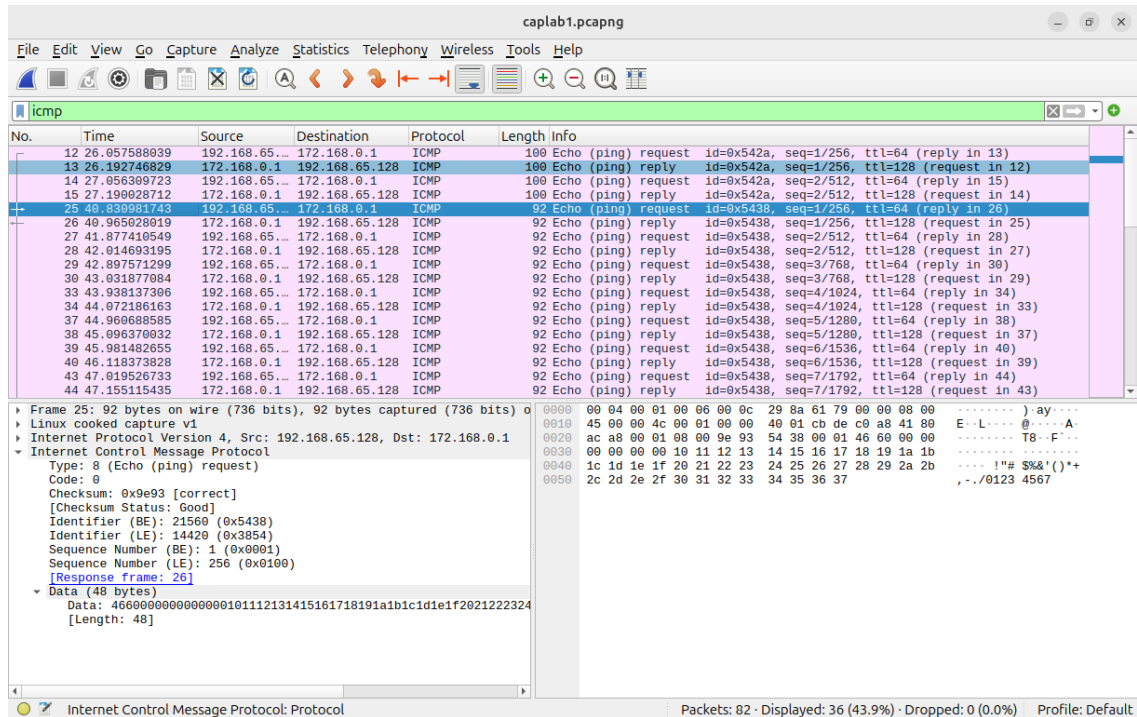


Figura 8: Letra “F” en paquete ICMP Request inicial.

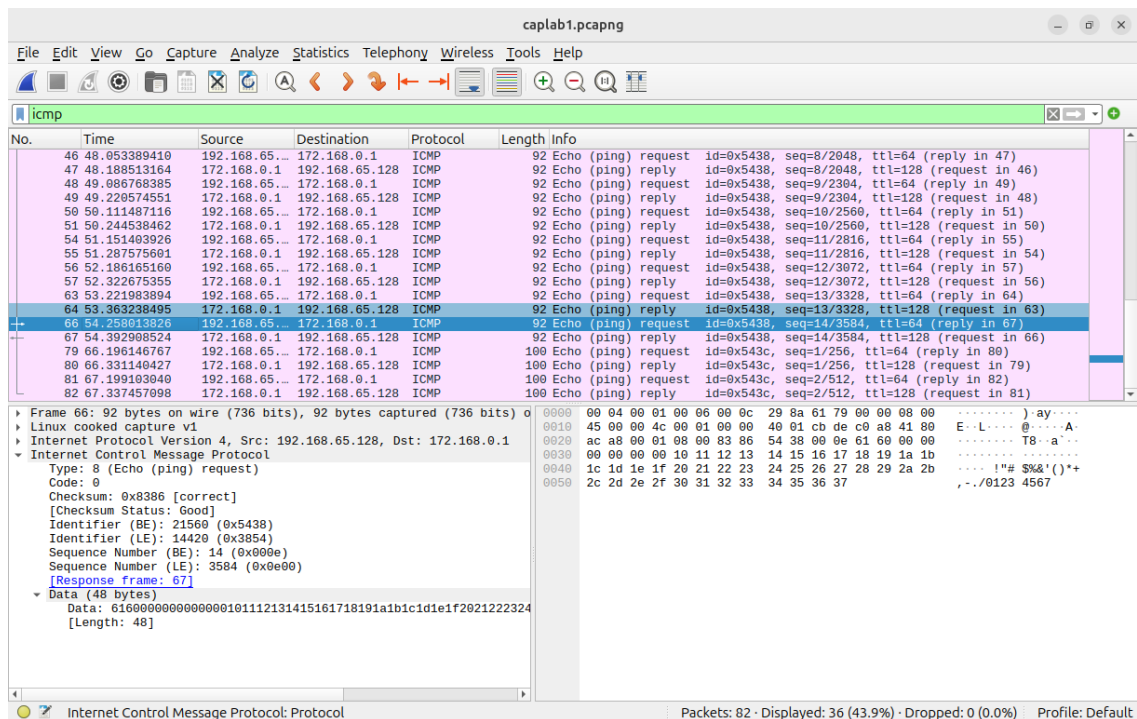


Figura 9: Letra “a” en paquete ICMP Request final.

Notar que la primera letra enviada es una “F” y la última una “a”, tal como se muestra en la figura 8 y 9 respectivamente, por lo que el mensaje cifrado se ha enviado correctamente.

3.3. Actividad 3

Para esta última actividad tenemos que construir el mensaje cifrado registrado en el paso anterior, juntando así todos los caracteres y aplicando cifrado César a la inversa, es decir, ver todas las combinaciones posibles para determinar el mensaje original. Una última vez nos apoyaremos de ChatGPT para crear un código en `python3` mencionando todos los detalles.

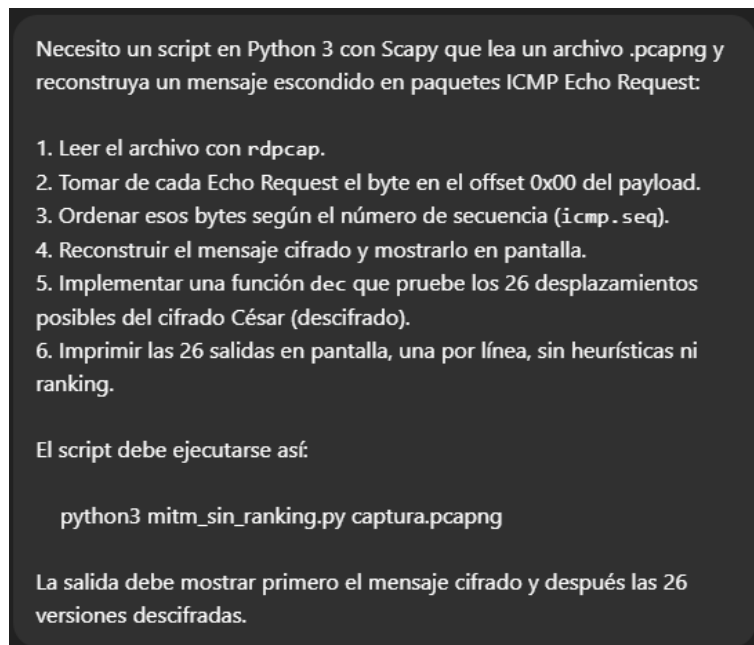
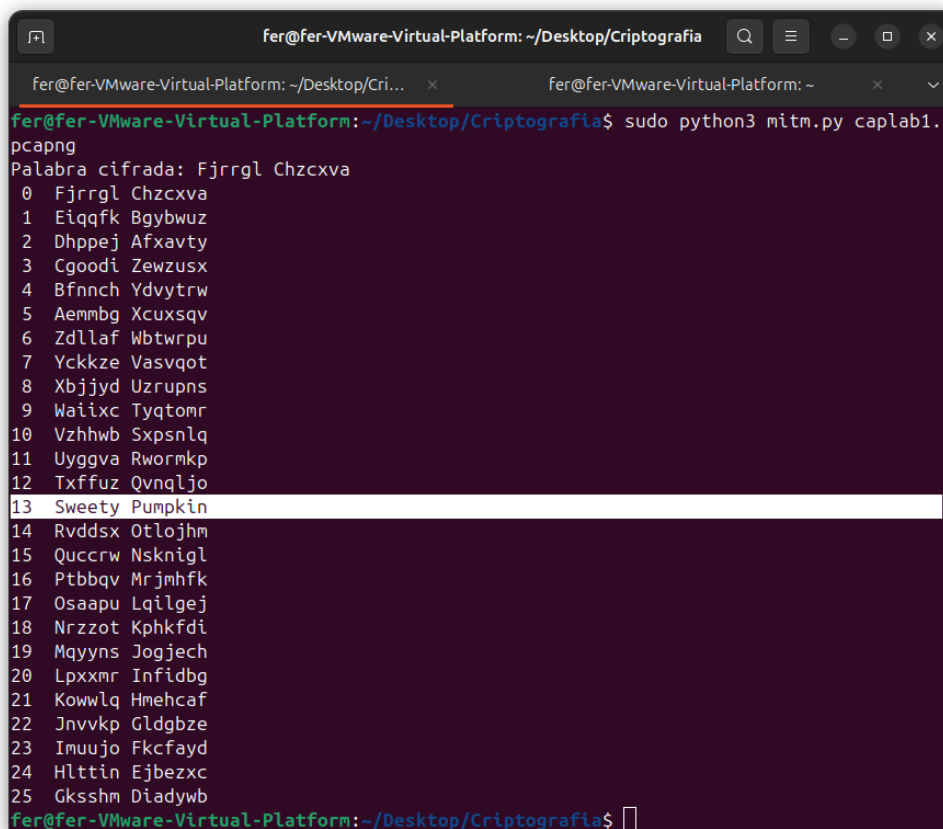


Figura 10: Prompt para el código del MitM.



```
fer@fer-VMware-Virtual-Platform: ~/Desktop/Criptografia
fer@fer-VMware-Virtual-Platform: ~/Desktop/Cri... x
fer@fer-VMware-Virtual-Platform: ~/Desktop/Criptografia$ sudo python3 mitm.py caplab1.pcapng
Palabra cifrada: Fjrrgl Chzcxa
0 Fjrrgl Chzcxa
1 Eiqqfk Bgybwuz
2 Dhppej Afxavty
3 Cgoodi Zewzusx
4 Bfnnch Ydvyrw
5 Aemmbg Xcuxsqv
6 ZdllaF Wbtwrpu
7 Yckkze Vasvqot
8 Xbjjyd Uzrupns
9 Waiixc Tyqtomr
10 Vzhhwb Sxpsnlq
11 Uyggva Rwormkp
12 Txffuz Qvnqljo
13 Sweety Pumpkin
14 Rvddsx Otlojhm
15 Quccrw Nsknigl
16 Ptbbaq Mrjmhfk
17 Osaapu Lqilgej
18 Nrzzot Kphkfdi
19 Mqyyns Jogjech
20 Lpxxmr Infidbg
21 Kowwlq Hmehcaf
22 Jnvvkp Gldgbze
23 Imuujo Fkcfayd
24 Hlttin Ejbezxc
25 Gksshm Diadywb
fer@fer-VMware-Virtual-Platform: ~/Desktop/Criptografia$
```

Figura 11: Mensaje cifrado decodificado

Al ejecutar el programa por consola este recibe la captura registrada en **Wireshark** y retorna todas las combinaciones posibles del cifrado, y en este caso, el desplazamiento 13 corresponde al mensaje original.

Conclusiones y comentarios

El cifrado César es una de las técnicas de cifrado más antiguas del mundo, que se remonta al siglo I a.C., y que hasta el día de hoy sigue siendo útilizando para sistemas de comunicaciones y redes de datos al momento de enviar mensajes.

Durante este laboratorio logramos identificar todos los pasos de este cifrado, de como conectarlo a un ejemplo de comunicación, hasta obtener el mensaje original descifrado, todo esto con el uso de herramientas como **ChatGPT** para la creación de códigos que ayudan a ejecutar todas las actividades correspondientes, y el análisis de tráfico por medio de **Wireshark**, el cual ayudo a comprobar que cada uno de los paquetes enviados cumplan con los requisitos previamente vistos, para así no mantener sospecha frente alguna filtración de datos en la comunicación.

Aunque **ChatGPT** es una herramienta genial para agilizar todo el código de trabajo, al momento de utilizar los prompts seleccionados este cumplía con ciertos requisitos, sin embargo, había que pedir modificaciones para mantener consistencia con las actividades planteadas.

Los códigos utilizados en la actividad pueden ser encontrados en el siguiente link: <https://github.com/f>