

Informe Laboratorio 1

Sección 3

Sebastián Alonzo

e-mail: sebastian.alonzo@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	5
3.3. Actividad 3	6

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.

```

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 sfef
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 vkbmhzktybt r lxznkbwtw xg kxw xl
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

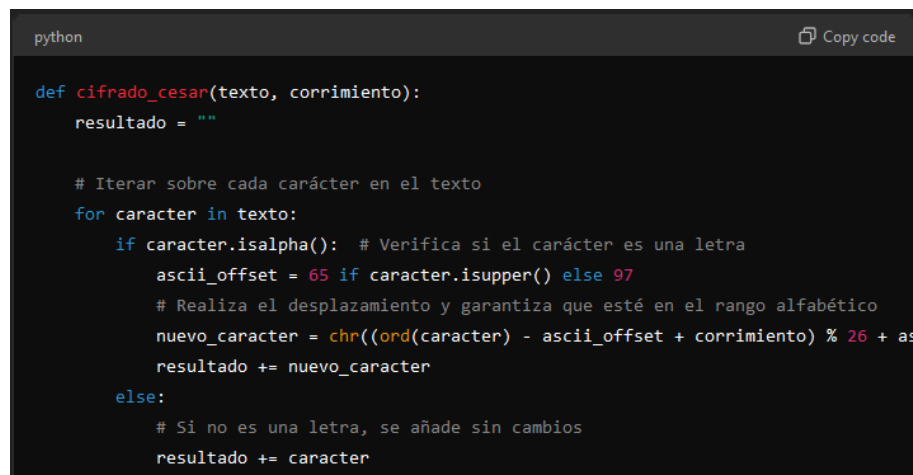
```

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

Como inicio de la actividad se sigue con las instrucciones utilizando la herramienta de Inteligencia Artificial 'Chatgpt' entregándole un 'prompt', mensaje que interpreta para replicar por computadora.



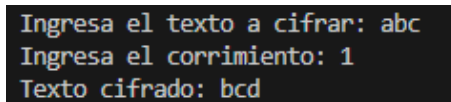
```
python Copy code

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

    # Iterar sobre cada carácter en el texto
    for caracter in texto:
        if caracter.isalpha(): # Verifica si el carácter es una letra
            ascii_offset = 65 if caracter.isupper() else 97
            # Realiza el desplazamiento y garantiza que esté en el rango alfabético
            nuevo_caracter = chr((ord(caracter) - ascii_offset + corrimiento) % 26 + ascii_offset)
            resultado += nuevo_caracter
        else:
            # Si no es una letra, se añade sin cambios
            resultado += caracter
```

Figura 1: ChatGPT Python cifrado cesar Prompt.

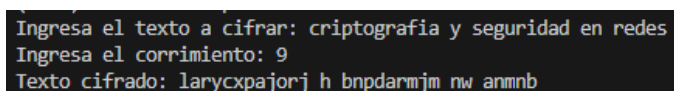
Aquí, en la figura 1, se puede apreciar el código en python para el cifrado cesar, verificando que el carácter sea una letra y no un numero o símbolo para luego darle el correcto valor ASCII y hacer el corrimiento.



```
Ingresa el texto a cifrar: abc
Ingresa el corrimiento: 1
Texto cifrado: bcd
```

Figura 2: Prueba del cifrado con mensaje 'abc'.

Se puede apreciar en la figura 2 el correcto funcionamiento del corrimiento, intentando con el mismo mensaje de la actividad se tiene:



```
Ingresa el texto a cifrar: criptografia y seguridad en redes
Ingresa el corrimiento: 9
Texto cifrado: larycxpajorj h bnpdarmjm nw anmnb
```

Figura 3: Prueba del cifrado con mensaje de actividad 1.

Manteniendo la similitud del mensaje original y cifrado en la primera actividad respaldado por la figura 3.

3.2. Actividad 2

Para el modo sigilo de la segunda actividad se prioriza primero el poder enviar los caracteres en varios paquetes ICMP request. Se utiliza nuevamente chatGPT el cual reutiliza el script o código anterior.

```
def enviar_ping(mensaje, destino):  
    for caracter in mensaje:  
        paquete = IP(dst=destino) / ICMP() / caracter  
        respuesta = sr1(paquete, timeout=1, verbose=0)  
        if respuesta:  
            print("Sent 1 packets.")  
        else:  
            print("No response received.")  
  
if __name__ == "__main__":  
    if len(sys.argv) != 4:  
        print(f"Uso: {sys.argv[0]} <destino> <mensaje> <corrimiento>")  
        sys.exit(1)
```

Figura 4: chatGPT prompt para enviar paquetes con un carácter en cifrado cesar por ICMP.

Como se puede apreciar en la figura 4, configura todo para paquete para que tenga el destino y origen correcto así como enviar el mensaje en caracteres por paquete y para poder ver un resultado se debe entregar el destino, el mensaje y el corrimiento; probando este código se entrega:

```
● (base) PS C:\Users\piesd\OneDrive\Documentos\vscode> python Actividad2.py 1.1.1.1 "abc" 1  
WARNING: Wireshark is installed, but cannot read manuf !  
Sent 1 packets.  
Sent 1 packets.  
Sent 1 packets.
```

Figura 5: Resultado de terminal en prueba de código Actividad2.

Y para ver el valor esperado, que seria el ultimo carácter como 'd', haciendo uso de el Sniffer Wireshak con filtrado en icmp con la ip de destino 1.1.1.1 se puede encontrar el paquete en la siguiente figura 6

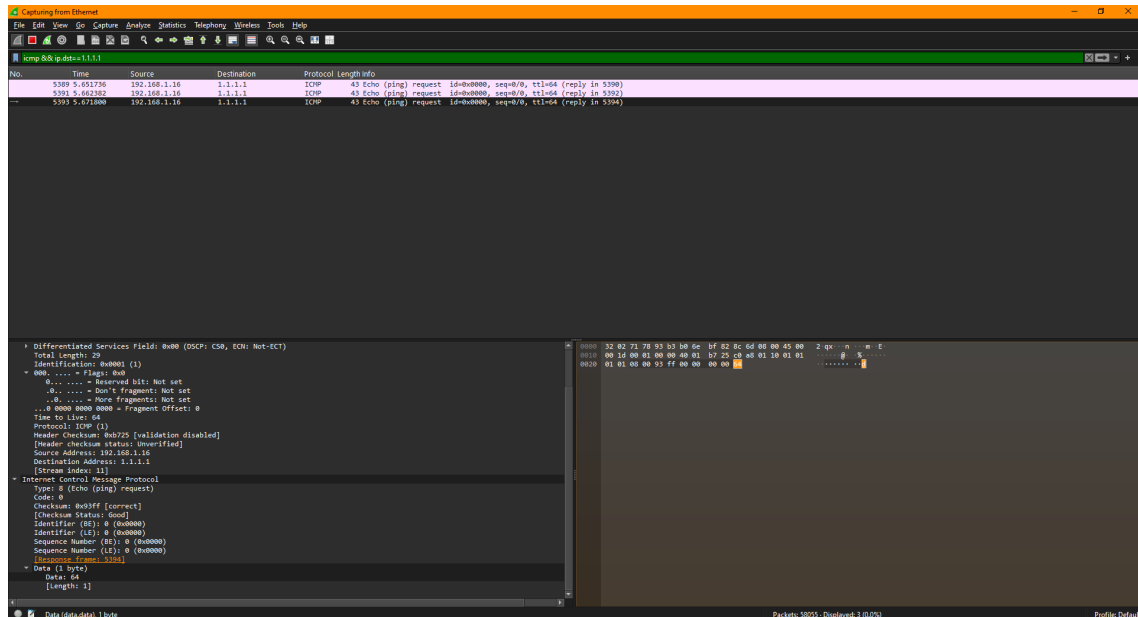


Figura 6: Sniff de paquete creado por el código, entregando el valor esperado 'd' dentro del payload.

3.3. Actividad 3

Entregando el mismo código por la actividad 2, chatGPT arroja un código que interpreta un archivo .pcapng hecho por Wireshark y arroja las combinaciones posibles como se ve en la siguiente figura 7:

```
def obtener_mensaje_cifrado(pcap_file):
    paquetes = rdpcap(pcap_file)
    mensaje_cifrado = ""

    for paquete in paquetes:
        if ICMP in paquete and paquete[ICMP].type == 8: # ICMP echo request (type 8)
            carga_util = str(paquete[ICMP].load, 'utf-8', errors='ignore')
            mensaje_cifrado += carga_util

    return mensaje_cifrado

def es_mensaje_probable(mensaje):
    palabras_comunes = ["the", "and", "el", "la", "que", "de", "y", "en"]
    for palabra in palabras_comunes:
        if palabra in mensaje.lower():
            return True
    return False
```

Figura 7: Prompt con descifrado de carga útil y método de mensaje probable.

Al probar este código con un mensaje 'Hola' con corrimiento 3, guardándolo y entregándolo al script este nos arroja los siguientes posibles mensajes en la figura 8 :

```
Probando todas las combinaciones de corrimiento:
Corrimiento 0: Krod
Corrimiento 1: Jqnc
Corrimiento 2: Ipmb
Corrimiento 3: Hola
Corrimiento 4: Gnkz
Corrimiento 5: Fmjy
Corrimiento 6: Elix
Corrimiento 7: Dkhw
Corrimiento 8: Cjgv
Corrimiento 9: Bifu
Corrimiento 10: Ahet
Corrimiento 11: Zgds
Corrimiento 12: Yfcr
Corrimiento 13: Xebq
Corrimiento 14: Wdap
Corrimiento 15: Vczo
Corrimiento 16: Ubyn
Corrimiento 17: Taxm
Corrimiento 18: Szwl
Corrimiento 19: Ryvk
Corrimiento 20: Qxuj
Corrimiento 21: Pwti
Corrimiento 22: Ovsh
Corrimiento 23: Nurg
Corrimiento 24: Mtqf
Corrimiento 25: Lspe
```

Figura 8: Posibles casos de corrimiento para la palabra hola.

Conclusiones y comentarios

A modo de conclusión he de declarar que el objetivo de hacer el mensaje sigiloso no pudo ser completado en su totalidad, gran desafío fue el utilizar la herramienta de chatGPT para coincidir con un acuerdo respecto a como hacer parecer el mensaje lo mas cercano a un ping pero, por principalmente administración del tiempo no se completo a tiempo esta parte del laboratorio.

Algo a destacar es la diferencia de entrega de chatGPT4 y su versión gratuita lo que también ralentizo el trabajo, poder incluir imágenes para explicar un problema a la inteligencia

artificial hace del uso de esta algo mucho mas fluido y natural, encontrando detalles por los que uno podría pasar desapercibido.

Para finalizar el trabajo fue hecho si con mucha mas agilidad con el uso de chatGPT, si bien a veces se confunde y entrega respuestas repetidas o lo contrario a lo esperado, muchas veces daba con probar y modificar un par de cosas o simplemente mejorar el prompt. No menor es la importancia del prompt que uno le deja a la inteligencia artificial ya que al igual que un computador es una maquina que para evitar respuestas no esperadas se debe ser más y más preciso para obtener lo buscado. Todo lo visto en este laboratorio se puede ver en el github de aquí.