

Informe Laboratorio 5

Sección 1

Pablo Alejandro Díaz Chamorro
e-mail: pablo.diaz_c@mail.udp.cl

Junio de 2023

Índice

1. Descripción de actividades	2
2. Desarrollo (Parte 1)	4
2.1. Códigos de cada Dockerfile	4
2.1.1. S1	5
2.1.2. C1	5
2.1.3. C2	5
2.1.4. C3	6
2.1.5. C4	6
2.2. Creación de las credenciales para S1	6
2.3. Tráfico generado por C1 (detallado)	6
2.4. Tráfico generado por C2 (detallado)	14
2.5. Tráfico generado por C3 (detallado)	18
2.6. Tráfico generado por C4 (4 (iface lo) (detallado)	22
2.7. Diferencia entre C1 y C2	26
2.8. Diferencia entre C2 y C3	26
2.9. Diferencia entre C3 y C4	26
3. Desarrollo (Parte 2)	26
3.1. Identificación del cliente ssh	26
3.2. Replicación de tráfico (paso por paso)	27
4. Desarrollo (Parte 3)	28
4.1. Replicación de tráfico (paso por paso)	28

1. Descripción de actividades

Para este último laboratorio, nuestro informante ya sabe que puede establecer un medio seguro sin un intercambio previo de una contraseña, gracias al protocolo diffie-hellman. El problema es que ahora no sabe si confiar en el equipo con el cual establezca comunicación, ya que las credenciales de usuario pueden haber sido divulgadas por algún soplón.

Para el presente laboratorio deberá:

- Crear 4 contenedores en Docker, donde cada uno tendrá el siguiente SO: Ubuntu 14.10, Ubuntu 16.10, Ubuntu 18.10 y Ubuntu 20.10, a los cuales llamaremos C1,C2,C3,C4/S1 respectivamente.
- Para cada uno de ellos, deberá instalar la última versión, disponible en sus repositorios, del cliente y servidor openssh.
- En S1 deberá crear el usuario test con contraseña test, para acceder a él desde los otros contenedores.
- En total serán 4 escenarios, donde cada uno corresponderá a los siguientes equipos:
 - C1 → S1
 - C2 → S1
 - C3 → S1
 - C4 → S1

Pasos:

1. Para cada uno de los 4 escenarios, solo deberá establecer la conexión y no realizar ningún otro comando que pueda generar tráfico (como muestra la Figura). Deberá capturar el tráfico de red generado y analizar el patrón de tráfico generado por cada cliente. De esta forma podrá obtener una huella digital para cada cliente a partir de su tráfico.

Indique el tamaño de los paquetes del flujo generados por el cliente y el contenido asociado a cada uno de ellos. Luego, indique qué información distinta contiene el escenario siguiente (diff incremental). El objetivo de esta tarea es identificar claramente los cambios entre las distintas versiones de ssh.

2. Para poder identificar que el usuario efectivamente es el informante, éste utilizará una versión única de cliente. ¿Con qué cliente SSH se habrá generado el siguiente tráfico?

Protocol	Length	Info
TCP	74	34328 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=14
TCP	66	34328 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0
SSHv2	85	Client: Protocol (SSH-2.0-OpenSSH_?)
TCP	66	34328 → 22 [ACK] Seq=20 Ack=42 Win=64256 Len=
SSHv2	1578	Client: Key Exchange Init
TCP	66	34328 → 22 [ACK] Seq=1532 Ack=1122 Win=64128
SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exc
TCP	66	34328 → 22 [ACK] Seq=1580 Ack=1574 Win=64128
SSHv2	82	Client: New Keys
SSHv2	110	Client: Encrypted packet (len=44)
TCP	66	34328 → 22 [ACK] Seq=1640 Ack=1618 Win=64128
SSHv2	126	Client: Encrypted packet (len=60)
TCP	66	34328 → 22 [ACK] Seq=1700 Ack=1670 Win=64128
SSHv2	150	Client: Encrypted packet (len=84)
TCP	66	34328 → 22 [ACK] Seq=1784 Ack=1698 Win=64128
SSHv2	178	Client: Encrypted packet (len=112)
TCP	66	34328 → 22 [ACK] Seq=1896 Ack=2198 Win=64128

Figura 1: Tráfico generado del informante

Repite este tráfico generado en la imagen. Debe generar el tráfico con la misma versión resaltada en azul.

3. Para que el informante esté seguro de nuestra identidad, nos pide que el patrón del tráfico de nuestro server también sea modificado, hasta que el Key Exchange Init del server sea menor a 300 bytes. Indique qué pasos realizó para lograr esto.

TCP	66 42350 → 22 [ACK] Seq=2 Ack=
TCP	74 42398 → 22 [SYN] Seq=0 Win=
TCP	74 22 → 42398 [SYN, ACK] Seq=0
TCP	66 42398 → 22 [ACK] Seq=1 Ack=
SSHv2	87 Client: Protocol (SSH-2.0-0)
TCP	66 22 → 42398 [ACK] Seq=1 Ack=
SSHv2	107 Server: Protocol (SSH-2.0-0)
TCP	66 42398 → 22 [ACK] Seq=22 Ack=
SSHv2	1570 Client: Key Exchange Init
TCP	66 22 → 42398 [ACK] Seq=42 Ack=
SSHv2	298 Server: Key Exchange Init
TCP	66 42398 → 22 [ACK] Seq=1526 Ack=

Figura 2: Captura del Key Exchange

2. Desarrollo (Parte 1)

2.1. Códigos de cada Dockerfile

Primeramente, es necesario saber los comando a usar para crear y correr los contenedores, a través de los archivos dockerfile correspondiente, se hace de la siguiente forma:

```
sudo docker build -t nombrecontenedor -f archivodockerfile
sudo docker run -it nombrecontenedor bash
```

Con el fin de poder crear cuatro contenedores, los cuales son clientes y un servidor OpenSSH, se crean los códigos Dockerfile de cada uno de los escenarios, los cuales son los siguientes:

2.1.1. S1

El dockerfile para el servidor OpenSSH es:

```

1 FROM ubuntu:20.10
2
3 RUN sed -i 's/http://\archive.ubuntu.com/ubuntu/http://\old-releases.ubuntu.com/ubuntu/' /etc/apt/sources.list
4 RUN sed -i '/^deb.*security.ubuntu.com/s/^#/' /etc/apt/sources.list
5 RUN mkdir -p /run/sshd && chmod 755 /run/sshd
6 RUN apt update && apt install -y sudo net-tools openssh-client openssh-server
7 RUN apt-get install -y wireshark
8 RUN apt-get install -y tshark
9 RUN useradd -m -s /bin/bash test && echo 'test:test' | chpasswd
10 RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
11
12 EXPOSE 22
13 CMD ["/usr/sbin/sshd", "-D"]

```

Figura 3: Código Dockerfile S1.

La dirección ip del servidor que se pudo obtener es la 172.17.0.2, a través del comando en el bash del servidor:

```
ifconfig
```

2.1.2. C1

El dockerfile para el cliente 1 es:

```

1 FROM ubuntu:14.10
2
3 RUN sed -i 's/http://\archive.ubuntu.com/ubuntu/http://\old-releases.ubuntu.com/ubuntu/' /etc/apt/sources.list
4 RUN apt update && apt install -y sudo net-tools openssh-client
5
6
7

```

Figura 4: Código Dockerfile C1.

2.1.3. C2

El dockerfile para el cliente 2 es:

```

1 FROM ubuntu:16.10
2
3 RUN sed -i 's/http://\archive.ubuntu.com/ubuntu/http://\old-releases.ubuntu.com/ubuntu/' /etc/apt/sources.list
4 RUN sed -i '/^deb.*security.ubuntu.com/s/^#/' /etc/apt/sources.list
5 RUN apt update && apt install -y sudo net-tools openssh-client

```

Figura 5: Código Dockerfile C2.

2.1.4. C3

El dockerfile para el cliente 3 es:

```

1 FROM ubuntu:18.10
2 RUN sed -i 's/http:\/\/archive.ubuntu.com\/ubuntu/http:\/\/old-releases.ubuntu.com\/ubuntu/' /etc/apt/sources.list
3 RUN sed -i '/^deb.*security.ubuntu.com/s/^#/' /etc/apt/sources.list
4 RUN apt update && apt install -y sudo net-tools openssh-client

```

Figura 6: Código Dockerfile C3.

2.1.5. C4

El dockerfile para el cliente 4 es:

```

1 FROM ubuntu:20.10
2
3 RUN sed -i 's/http:\/\/archive.ubuntu.com\/ubuntu/http:\/\/old-releases.ubuntu.com\/ubuntu/' /etc/apt/sources.list
4 RUN sed -i '/^deb.*security.ubuntu.com/s/^#/' /etc/apt/sources.list
5 RUN mkdir -p /run/sshd && chmod 755 /run/sshd
6 RUN apt update && apt install -y sudo net-tools openssh-client openssh-server
7 RUN apt-get install -y wireshark
8 RUN apt-get install -y tshark
9 RUN useradd -m -s /bin/bash test && echo 'test:test' | chpasswd
10 RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
11
12 EXPOSE 22
13 CMD ["/usr/sbin/sshd", "-D"]

```

Figura 7: Código Dockerfile C4.

2.2. Creación de las credenciales para S1

En este caso, como se puede observar en el dockerfile del servidor OpenSSH (S1), la forma de poder crear credenciales se debe ejecutar lo siguiente:

```
9 RUN useradd -m -s /bin/bash test && echo 'test:test' | chpasswd
```

Figura 8: Código para creación de credenciales desde el dockerfile.

Cabe destacar que las credenciales a usar son:

- User: test
- Password: test

2.3. Tráfico generado por C1 (detallado)

Luego de haber ejecutado el contenedor de C1, en la terminal de ubuntu, se hace la conexión SSH hacia el servidor (S1) con la ip obtenida anteriormente, capturando el tráfico a través de la herramienta wireshark. Esto a través del siguiente comando:

```
root@02832b75b053:/# ssh test@172.17.0.2
The authenticity of host '172.17.0.2 (172.17.0.2)' can't be established.
ECDSA key fingerprint is f5:d5:2b:08:81:d3:82:69:37:f3:ff:06:fd:6a:21:2d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.17.0.2' (ECDSA) to the list of known hosts.
test@172.17.0.2's password:
Welcome to Ubuntu 20.10 (GNU/Linux 6.2.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

test@f08332a4ea87:~$ exit
logout
Connection to 172.17.0.2 closed.
root@02832b75b053:/# █
```

Figura 9: Comando conexión ssh entre C1 y S1.

Ya capturados los paquetes del tráfico generado por la conexión entre C1 y S1, se procede a analizar el tipo de información contenida en los paquetes generados:

- Primeramente, se puede observar que al comenzar la conexión SSH, se envían tres paquetes TCP: SYN, SYN ACK y ACK, los cuales son paquetes típicos que establecen la conexión.

2.3 Tráfico generado por C1 (detallado)

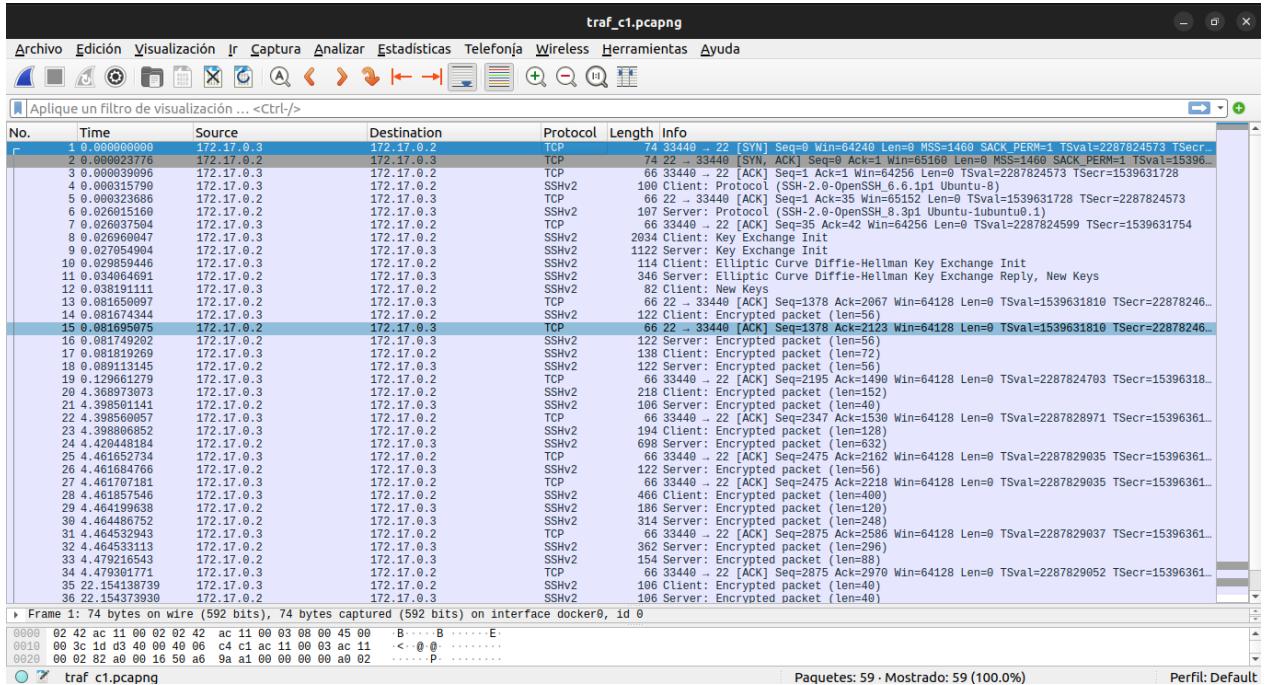


Figura 10: Paquetes TCP entre C1 y S1 capturado desde wireshark.

- En el caso del primer paquete, el cliente envía al servidor un paquete con protocolo SSHv2 de tamaño 100 bytes, en donde se le informa al servidor que esta utilizando OpenSSH con la versión 6.6.

2.3 Tráfico generado por C1 (detallado)

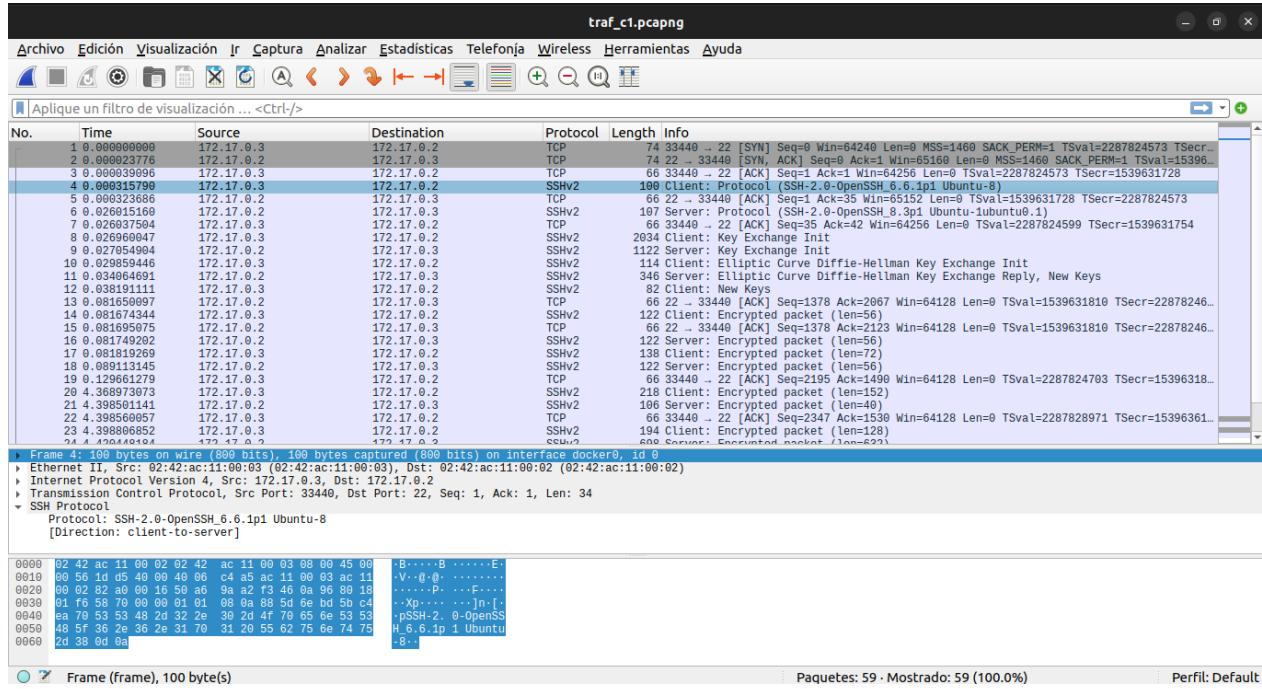


Figura 11: Paquete 1 entre C1 y S1 capturado desde wireshark.

- El segundo paquete, de igual manera el servidor envía al cliente un paquete con protocolo SSHv2 con un tamaño de 107 bytes, informando que utiliza OpenSSH con la versión 8.3.

2.3 Tráfico generado por C1 (detallado)

2 DESARROLLO (PARTE 1)

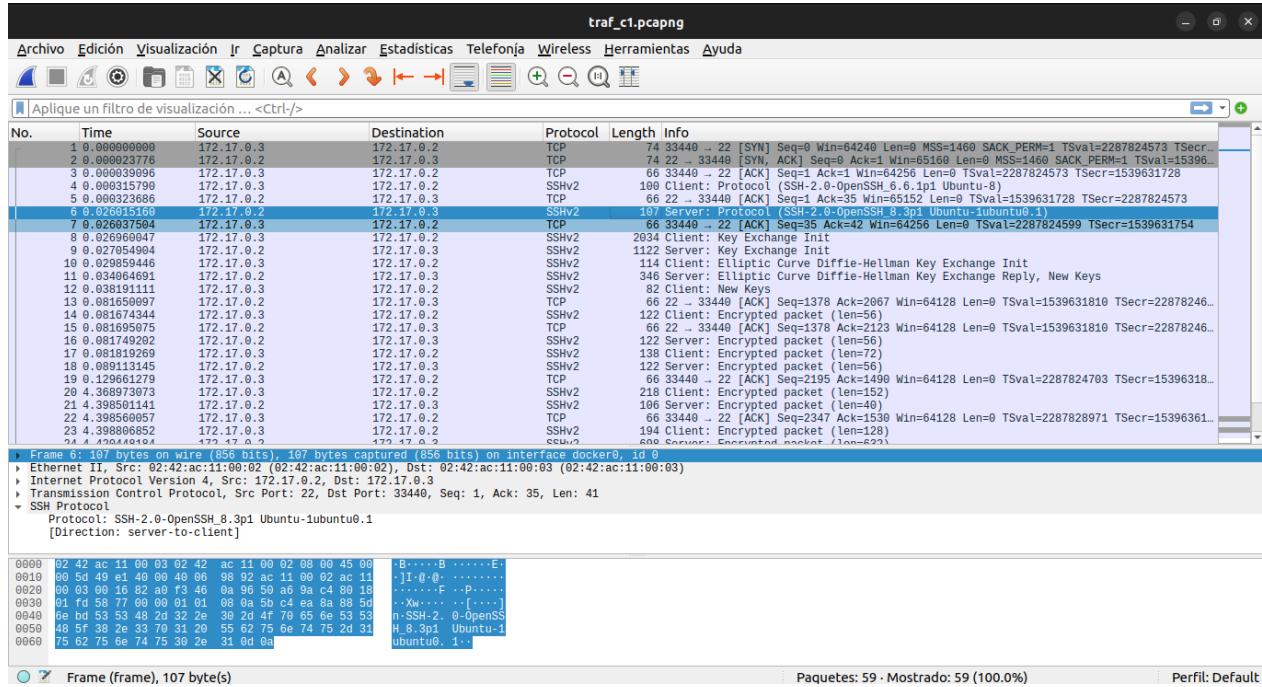


Figura 12: Paquete 2 entre C1 y S1 capturado desde wireshark.

- En este caso, se produce un intercambio de claves entre el cliente (C1) y servidor (S1), en donde primeramente el cliente envía el método a utilizar para el intercambio de claves **curve25519-sha256@libssh.org**, esto con el fin de establecer la conexión con el servidor SSH, con un tamaño de 2034 bytes en el paquete Key Exchange Init del cliente.

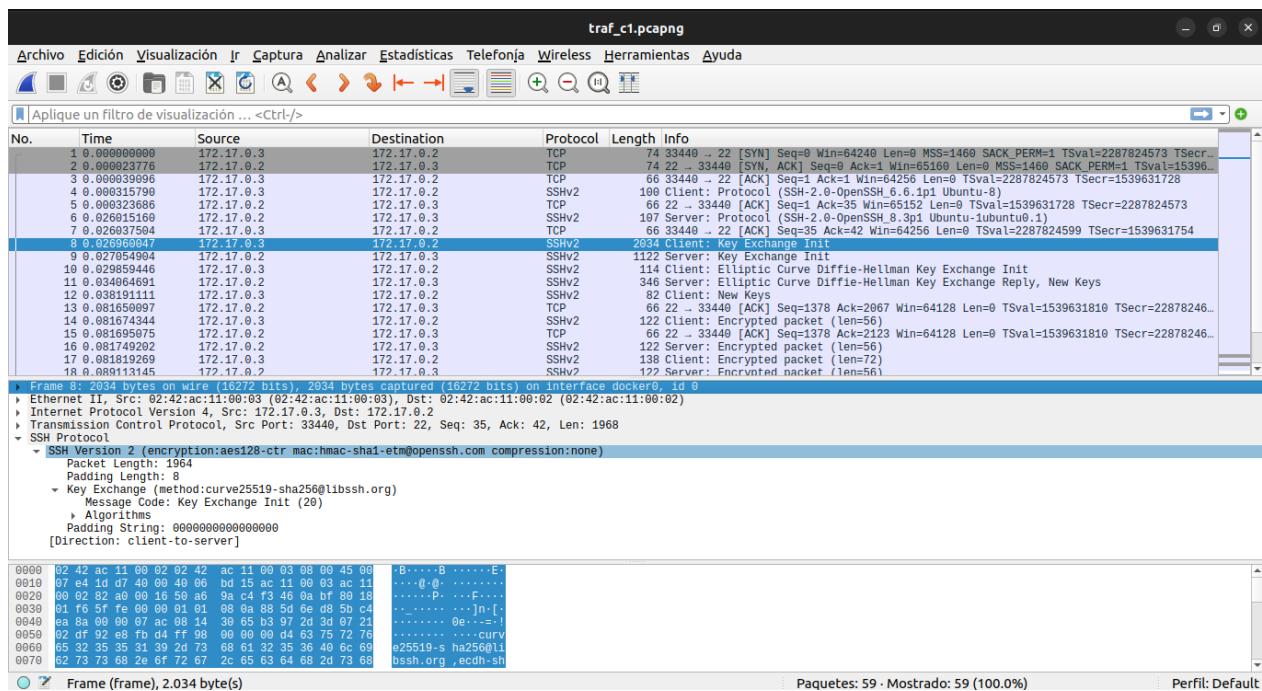


Figura 13: Paquete 3 entre C1 y S1 capturado desde wireshark.

- En este paquete el servidor responde al paquete anterior, confirmando el método de intercambio de claves `curve25519-sha256@libssh.org`, con un tamaño de 1222 bytes en el paquete Key Exchange Init del servidor.

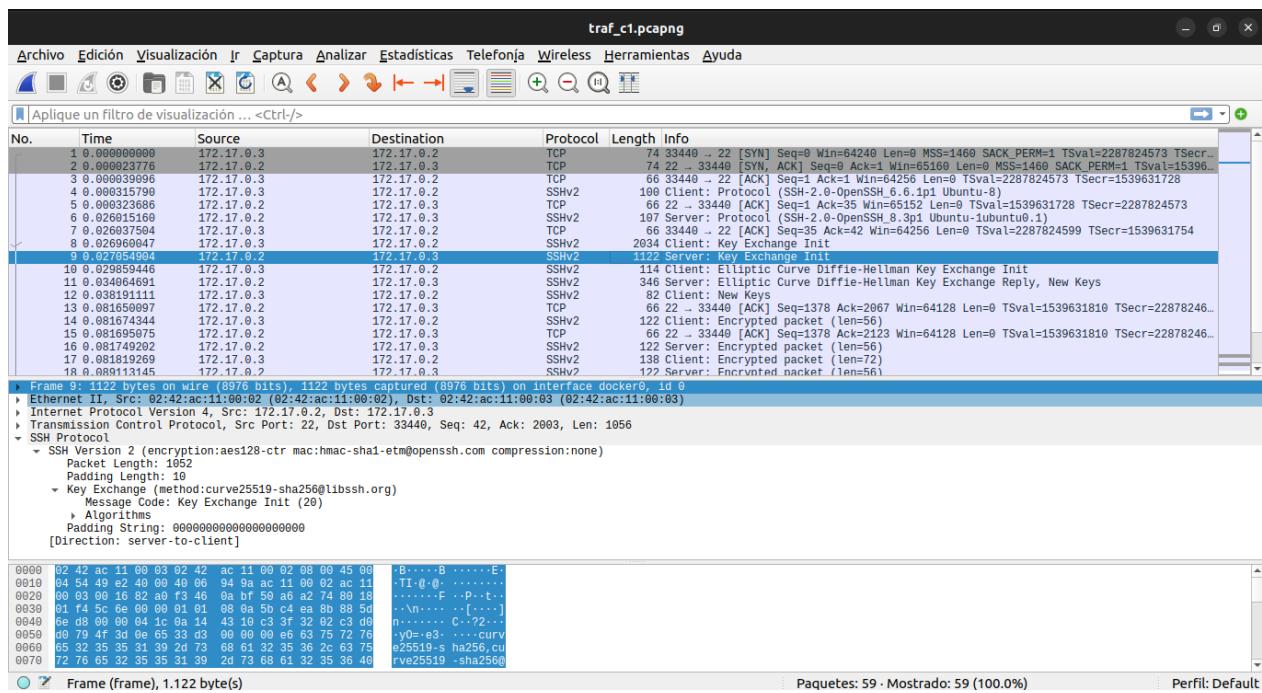


Figura 14: Paquete 4 entre C1 y S1 capturado desde wireshark.

- En las siguientes paquetes, se puede observar que se intercambian las claves públicas y parámetros anteriores entre el cliente y el servidor, en donde el servidor envía claves nuevas y el paquete encriptado.

2 DESARROLLO (PARTE 1)

2.3 Tráfico generado por C1 (detallado)

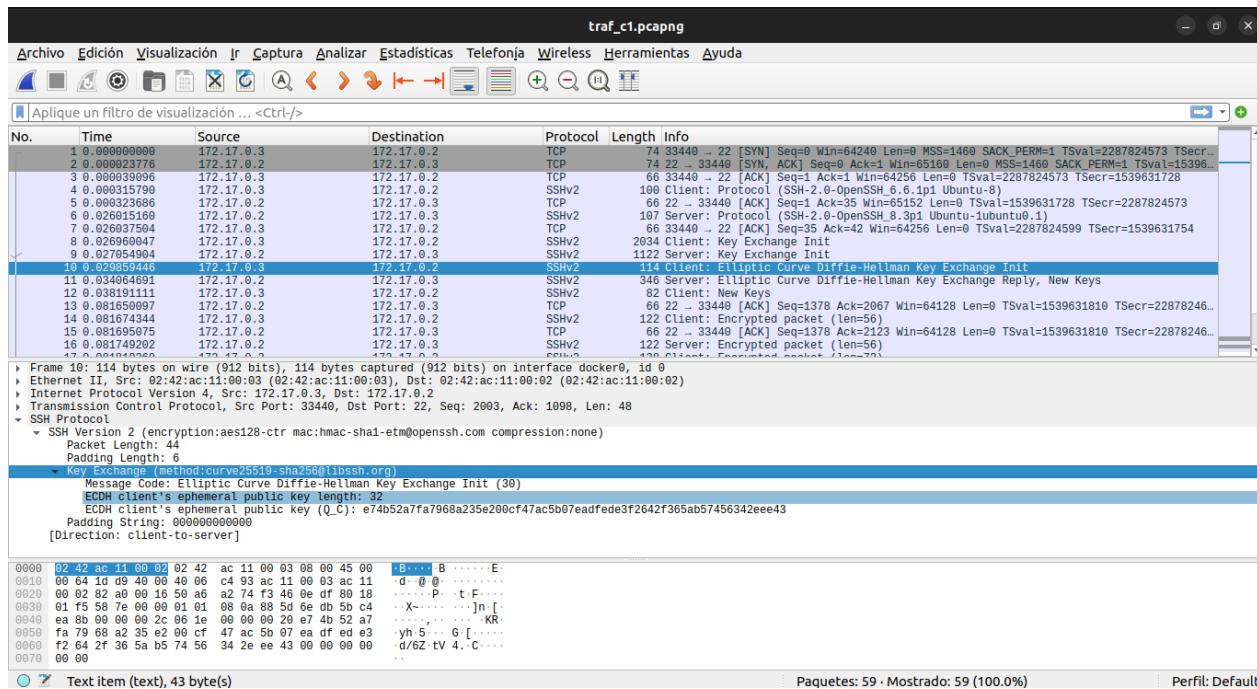


Figura 15: Paquete 5 entre C1 y S1 capturado desde wireshark.

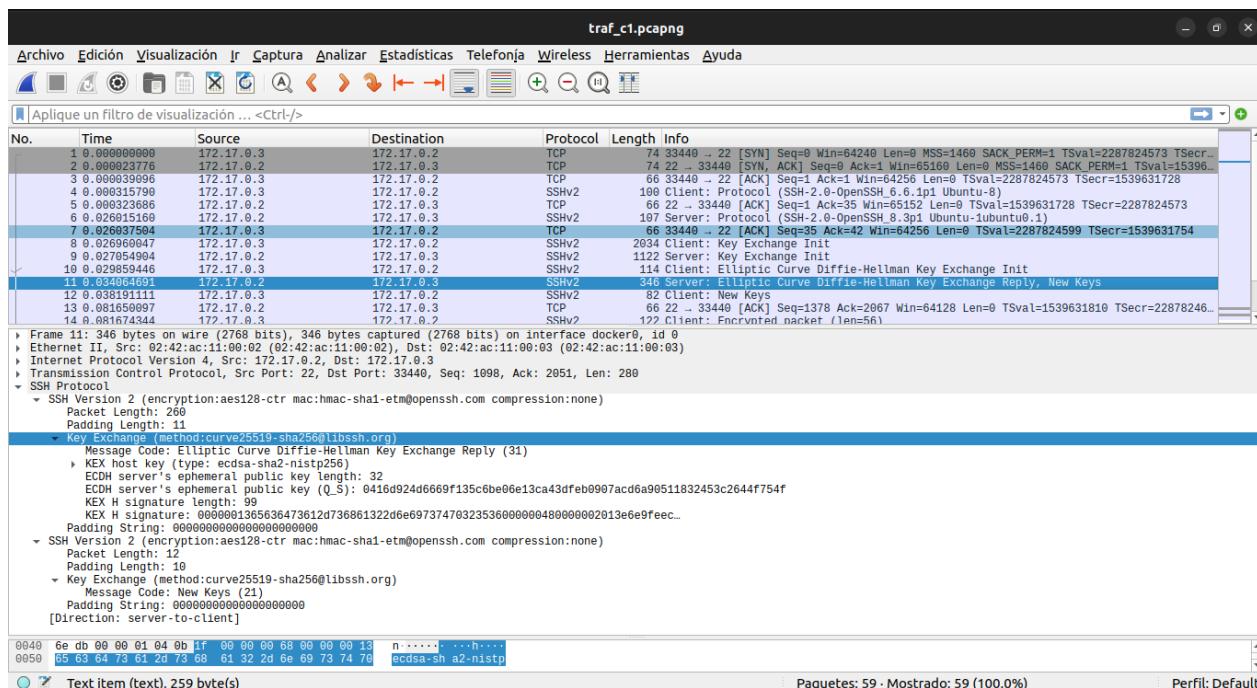


Figura 16: Paquete 6 entre C1 y S1 capturado desde wireshark.

- Por último, con el fin de aumentar la seguridad, el cliente envía claves nuevas (New

2.4 Tráfico generado por C2 (detallado)

Keys).

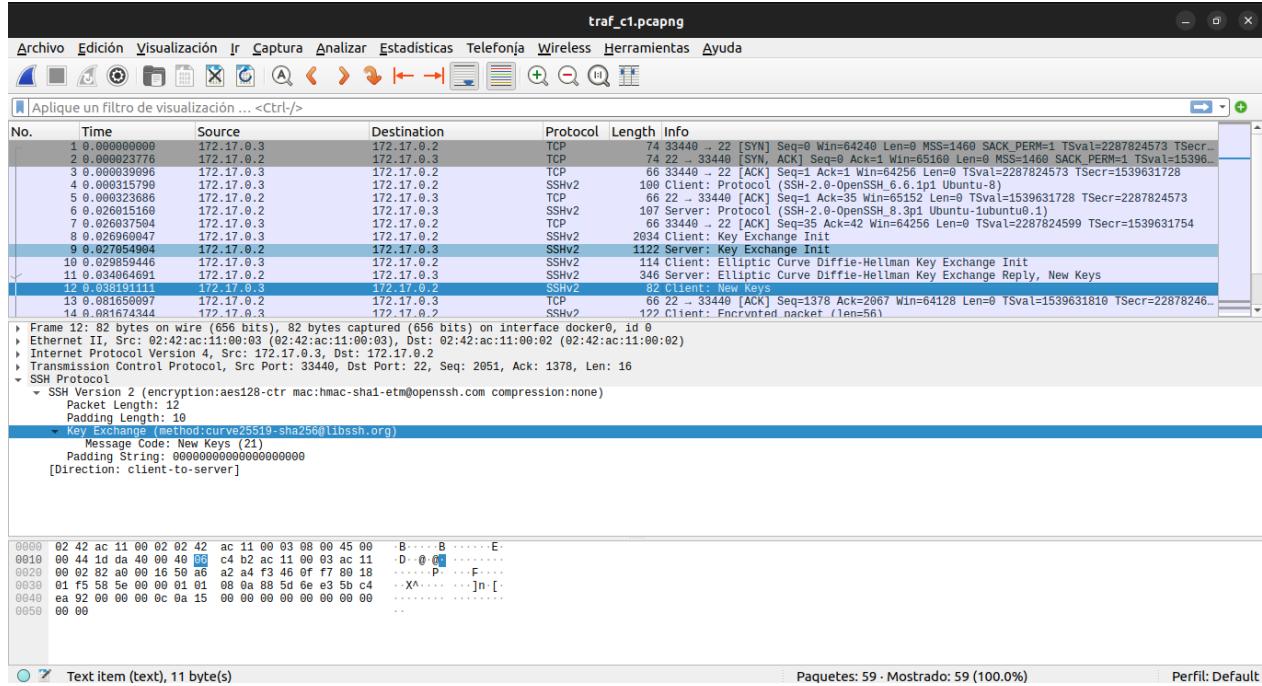


Figura 17: Paquete 7 entre C1 y S1 capturado desde wireshark.

Cabe mencionar que los paquetes siguientes ya están encriptados, con los parámetros y claves anteriormente dichos.

2.4. Tráfico generado por C2 (detallado)

De la misma forma que el cliente anterior, se procede a capturar el tráfico generado por la conexión SSH entre C2 y S1, a través del siguiente comando:

```
root@5b9452c1c99c:/# ssh test@172.17.0.2
The authenticity of host '172.17.0.2 (172.17.0.2)' can't be established.
ECDSA key fingerprint is SHA256:aLSxPCg9VI8cwpLhPAn1brn6K8wCXcTexQff8VcGYDM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.17.0.2' (ECDSA) to the list of known hosts.
test@172.17.0.2's password:
Welcome to Ubuntu 20.10 (GNU/Linux 6.2.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Nov 22 03:46:14 2023 from 172.17.0.3
test@f08332a4ea87:~$ exit
logout
Connection to 172.17.0.2 closed.
root@5b9452c1c99c:/# █
```

Figura 18: Comando conexión SSH entre C2 y S1.

Luego, se analizan la información de los paquetes del tráfico generado, en donde se sabe que el tráfico es **igual** al anterior, cambiando en lo siguiente:

- Para este caso, en el primer paquete el cliente informa al servidor que está utilizando OpenSSH con la versión 7.3. También el servidor informa la versión 8.3 de OpenSSH obtenida anteriormente de igual manera. Los paquetes tanto del cliente como el servidor tienen el mismo tamaño de 107 bytes.

2 DESARROLLO (PARTE 1)

2.4 Tráfico generado por C2 (detallado)

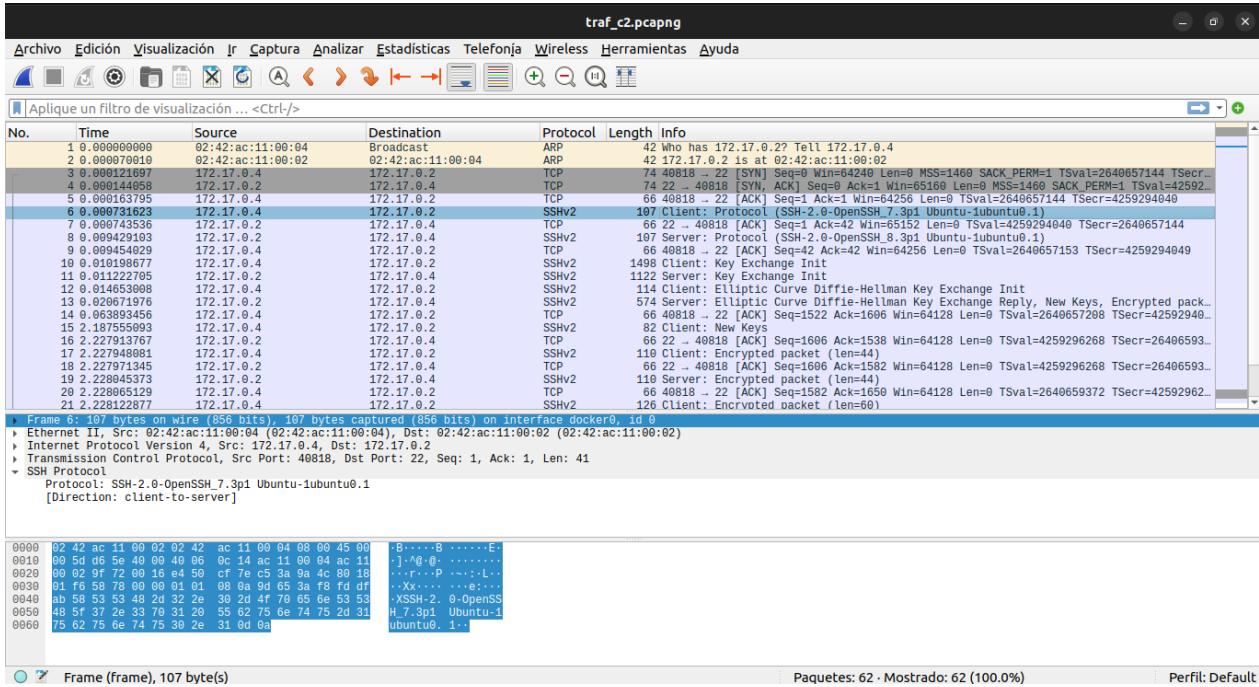


Figura 19: Paquete 1 entre C2 y S1 capturado desde wireshark.

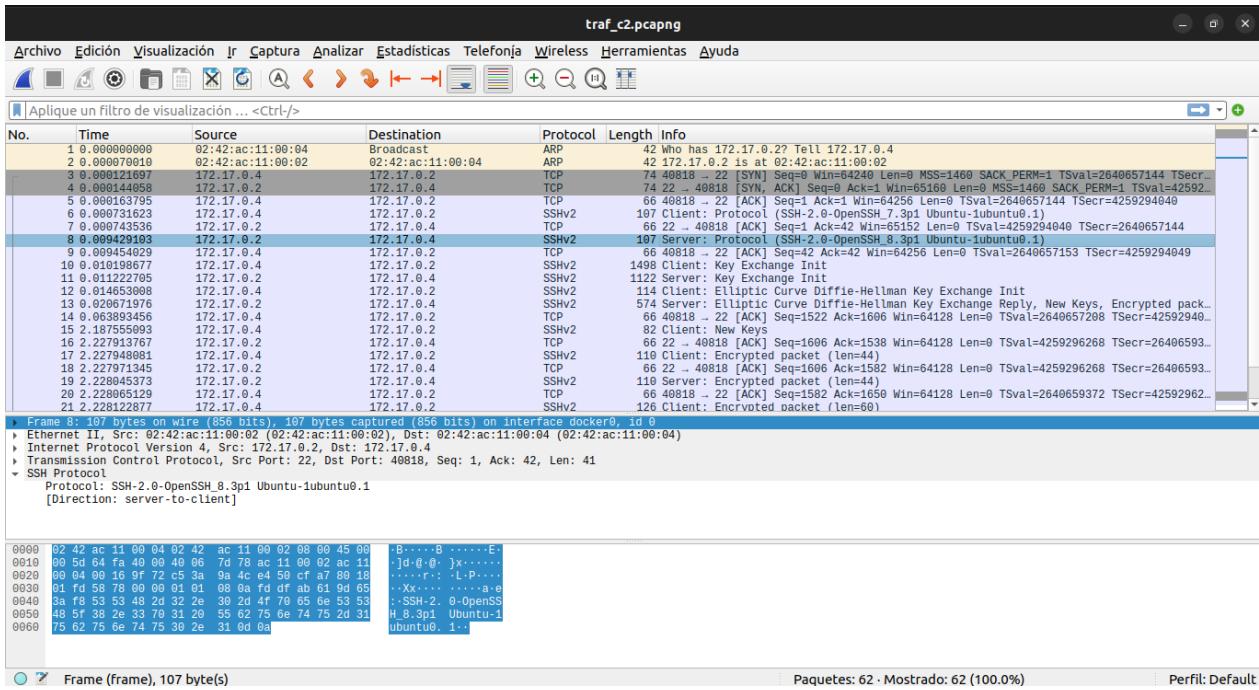


Figura 20: Paquete 2 entre C2 y S1 capturado desde wireshark.

- Por otro lado, debido al cambio del paquete anterior del cliente, los paquetes Key

Exchange Init del cliente y el servidor contienen información acerca del método de intercambio **curve25519-sha256@libssh.org**, utilizando el cifrado **chacha20-poly1305@openssh.com**. El tamaño del paquete del cliente es de 1498 bytes y el paquete del servidor es de 1122 bytes.

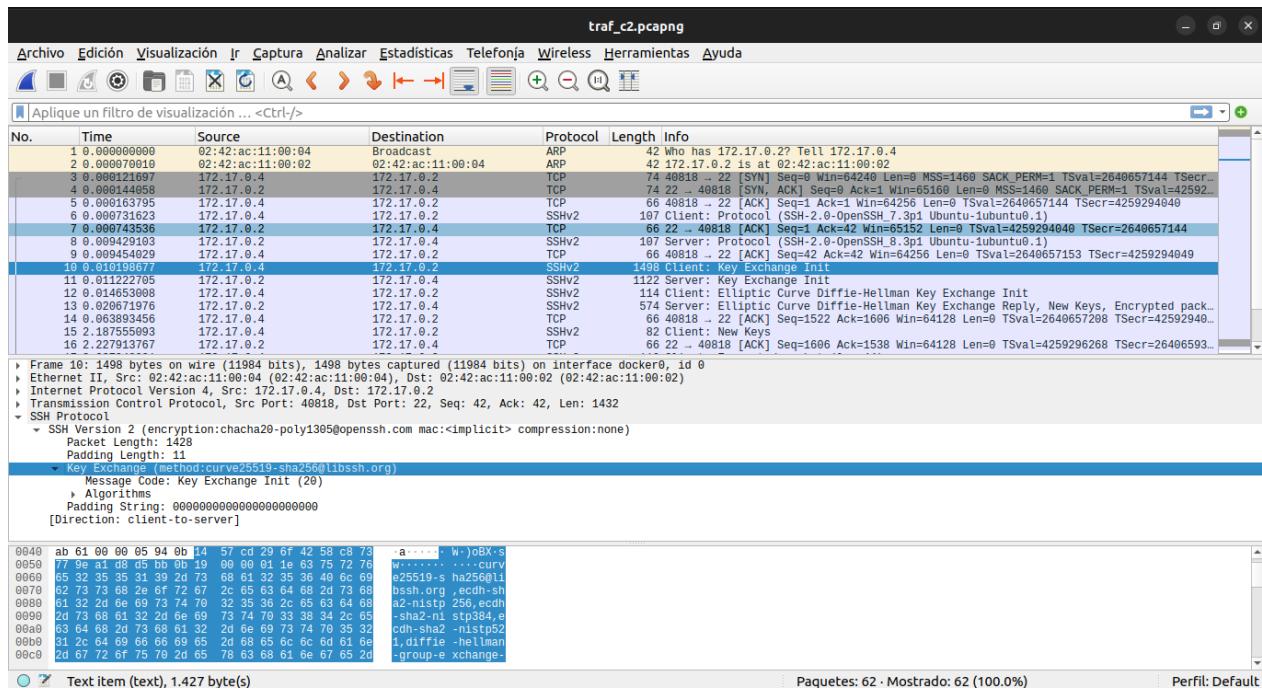


Figura 21: Paquete 3 entre C2 y S1 capturado desde wireshark.

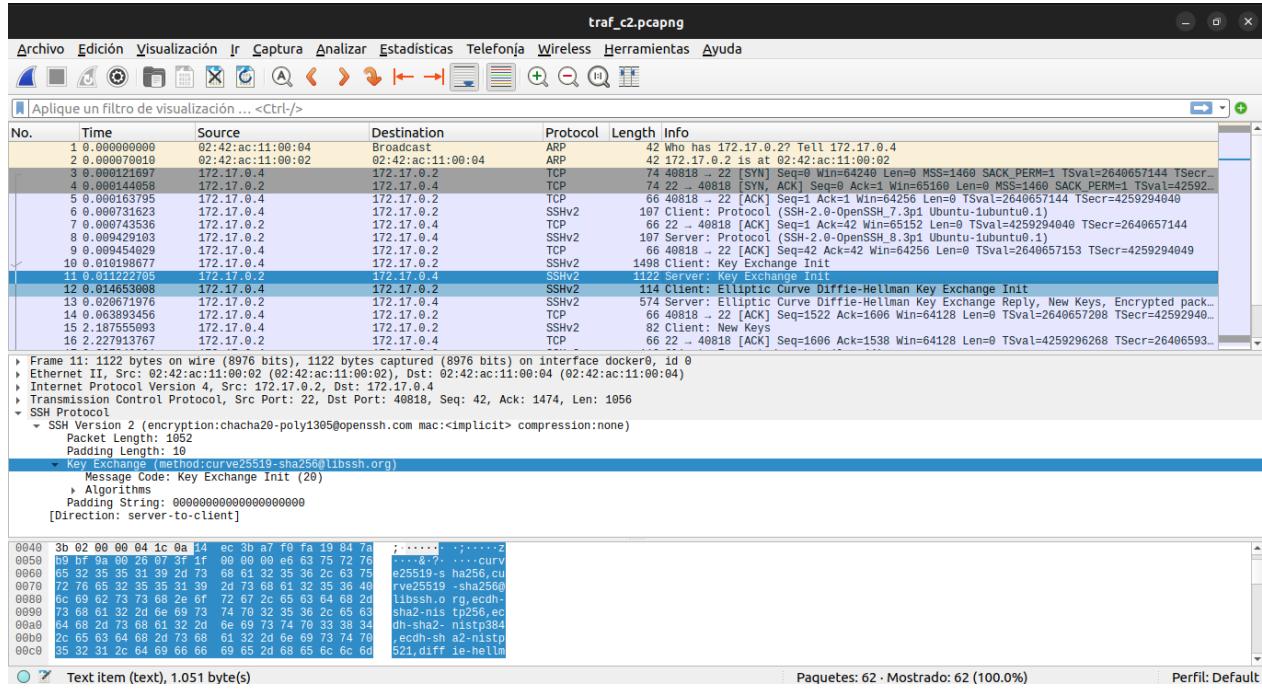


Figura 22: Paquete 4 entre C2 y S1 capturado desde wireshark.

Se sabe que los paquetes siguientes o los no mencionados anteriormente son iguales a los del tráfico entre C1 y S1, y así, para las siguientes capturas de tráfico, en donde claramente cambian los parámetros según las versiones utilizadas en el cliente.

2.5. Tráfico generado por C3 (detallado)

En este caso, se procede a capturar el tráfico generado por la conexión SSH entre el C3 y S1, gracias al siguiente comando:

```
root@7146080eed06:/# ssh test@172.17.0.2
The authenticity of host '172.17.0.2 (172.17.0.2)' can't be established.
ECDSA key fingerprint is SHA256:aLSxPCg9VI8cwpLhPAn1brn6K8wCXcTexQff8VcGYDM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.17.0.2' (ECDSA) to the list of known hosts.
test@172.17.0.2's password:
Welcome to Ubuntu 20.10 (GNU/Linux 6.2.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Nov 22 03:48:03 2023 from 172.17.0.4
test@f08332a4ea87:~$ exit
logout
Connection to 172.17.0.2 closed.
```

Figura 23: Comando conexión SSH entre C3 y S1.

Como bien se sabe, el tráfico generado es el mismo que al capturado por las conexiones anteriores, por lo tanto, se analizan los siguientes paquetes:

- En este caso los primeros paquetes enviados por el cliente contienen información acerca de que utiliza OpenSSH con la versión 7.7. En el caso del paquete del servidor, este informa que utiliza la misma versión de OpenSSH que se utilizó anteriormente en las otras conexiones. El tamaño de los paquetes es de 107 bytes cada uno.

2 DESARROLLO (PARTE 1)

2.5 Tráfico generado por C3 (detallado)

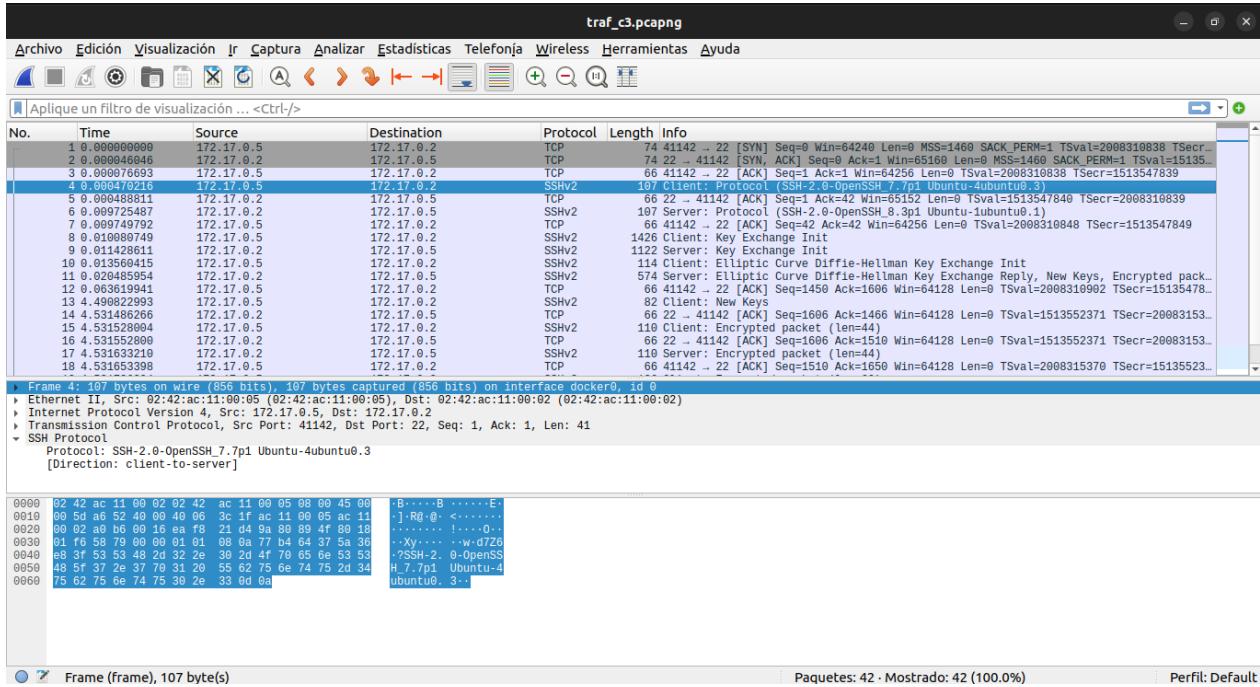


Figura 24: Paquete 1 entre C3 y S1 capturado desde wireshark.

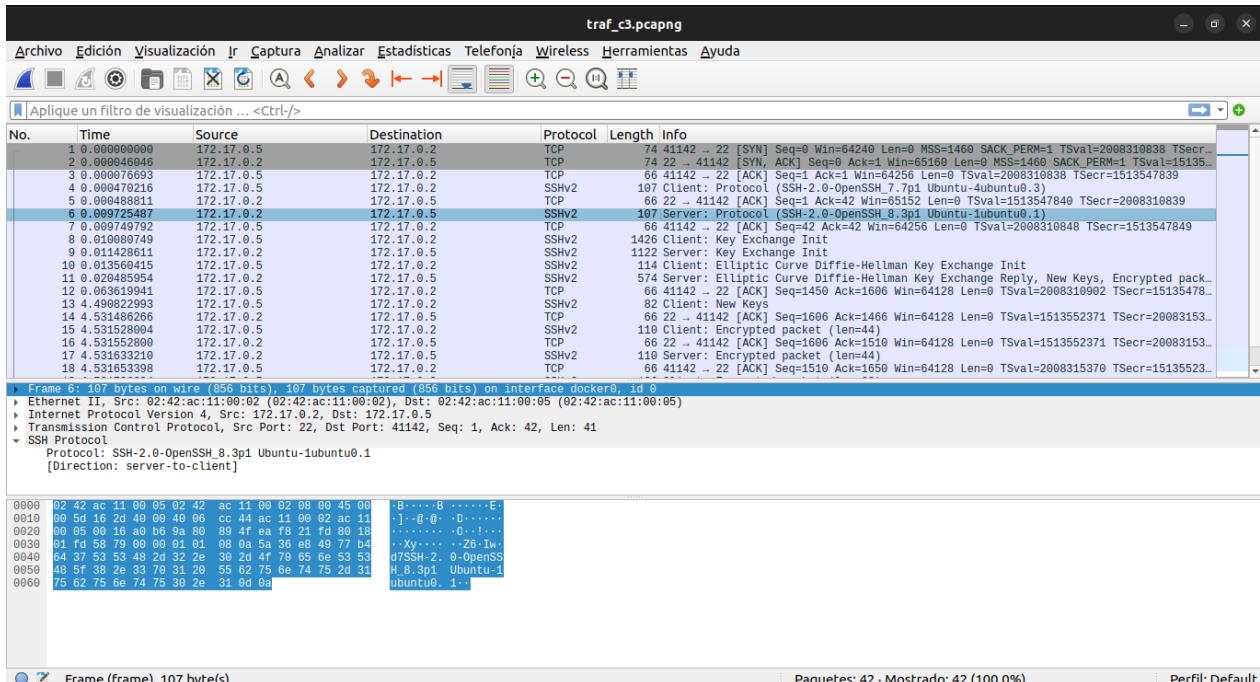


Figura 25: Paquete 2 entre C3 y S1 capturado desde wireshark.

- Los paquetes Key Exchange Init del cliente y el servidor, indican que utilizan el cifrado

chacha20-poly1305@openssh.com. El paquete del cliente es de tamaño 1426 bytes y el paquete del servidor es de tamaño 1122 bytes.

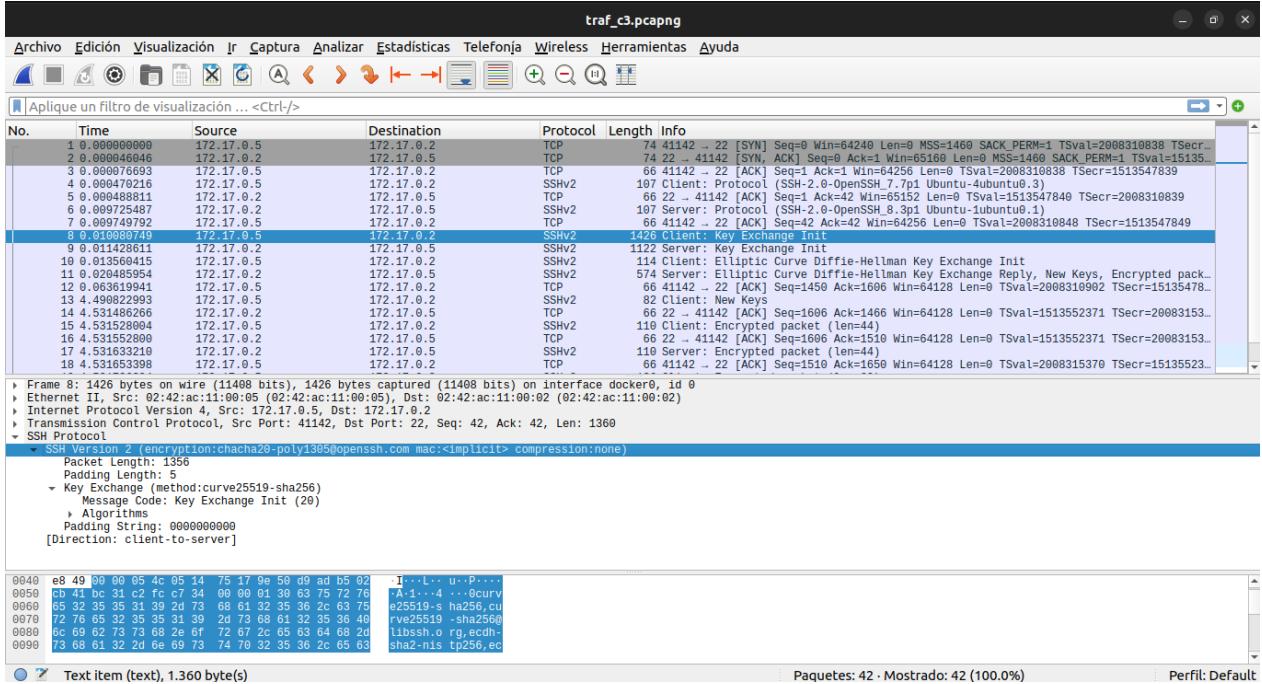


Figura 26: Paquete 3 entre C3 y S1 capturado desde wireshark.

2.6 Tráfico generado por C4 (4 (iface lo) (detallado)

2 DESARROLLO (PARTE 1)

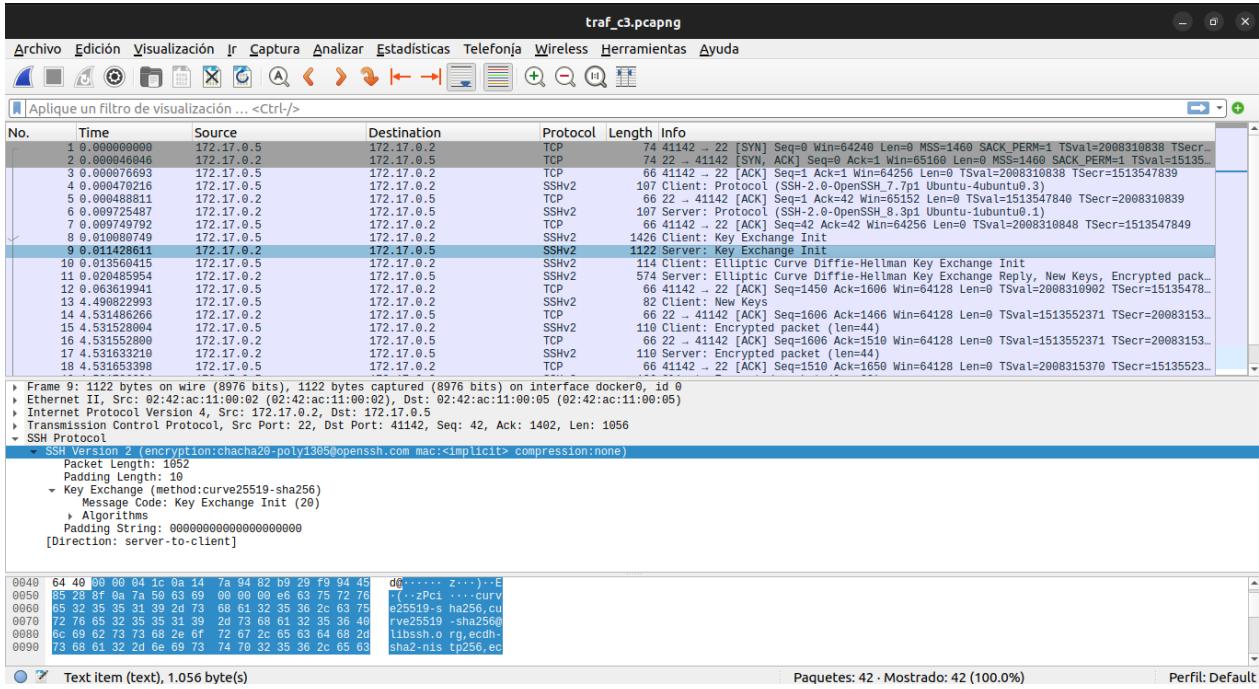


Figura 27: Paquete 4 entre C3 y S1 capturado desde wireshark.

2.6. Tráfico generado por C4 (4 (iface lo) (detallado)

Para el tráfico generado por la conexión SSH entre el C4 y S1, se procede a capturar este tráfico a través de la herramienta de tshark, debido a que el tráfico se está generando del mismo contenedor, por lo que se utiliza dos bash en C4, uno para capturar con tshark y otro para realizar la conexión SSH con el servidor, esto a través del siguiente comando:

```
root@7146080eed06:/# ssh test@172.17.0.2
The authenticity of host '172.17.0.2 (172.17.0.2)' can't be established.
ECDSA key fingerprint is SHA256:alSxPCg9VI8cwpLhAn1brngK8wCXcTexQff8VcGYDM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.17.0.2' (ECDSA) to the list of known hosts.
test@172.17.0.2's password:
Welcome to Ubuntu 20.10 (GNU/Linux 6.2.0-32-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Nov 22 03:48:03 2023 from 172.17.0.4
test@f088332a4ea87:~$ exit
logout
Connection to 172.17.0.2 closed.
```

Figura 28: Comando conexión SSH entre C4 y S1.

De la misma forma que en los tráficos anteriores, se procede a analizar algunos paquetes diferentes:

- El primer paquete del cliente hacia el servidor indica que utiliza la versión 8.3 de

OpenSSH, al igual que el servidor. EL tamaño de los dos paquetes es igual a 107 bytes para cada uno.

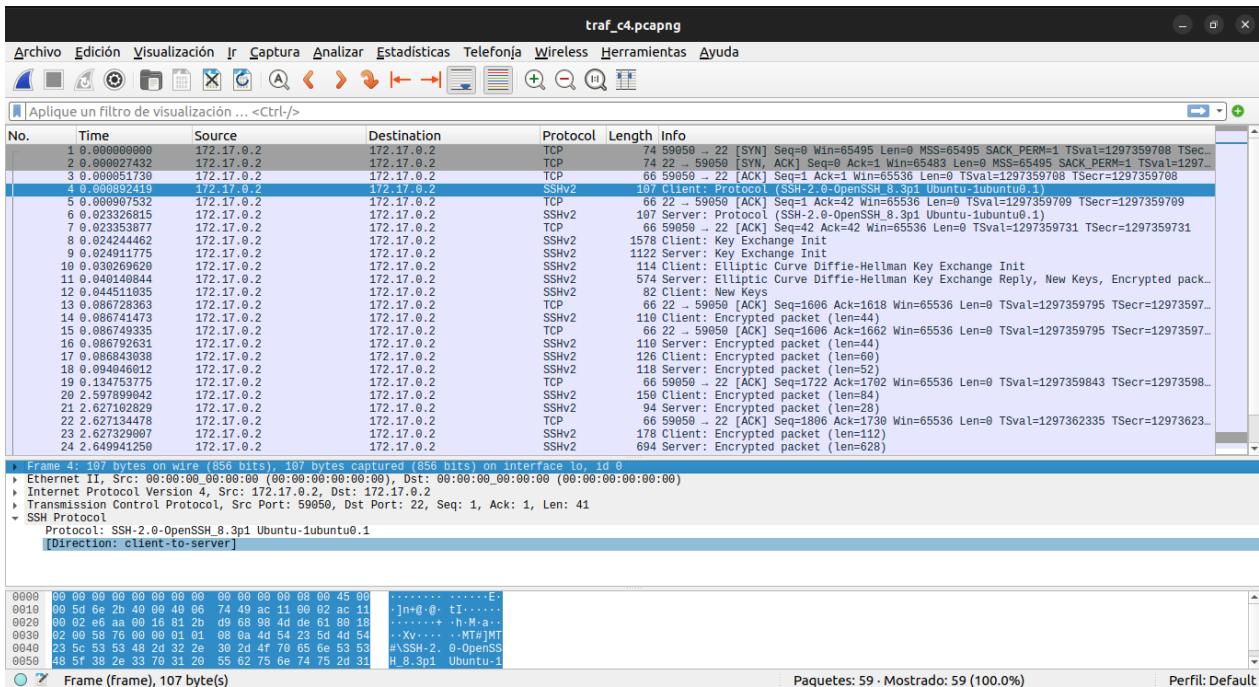


Figura 29: Paquete 1 entre C4 y S1 capturado desde wireshark.

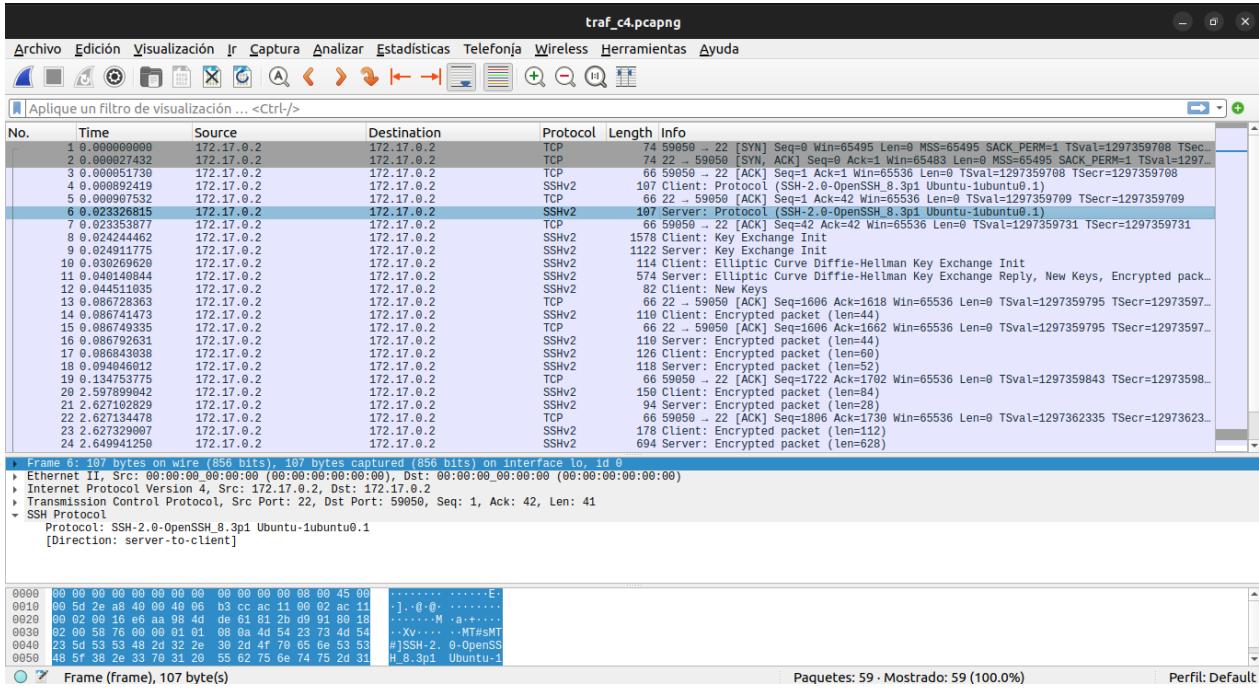


Figura 30: Paquete 2 entre C4 y S1 capturado desde wireshark.

- Por otro lado, en el caso de los paquetes Key Exchange Init, se observa que el cifrado a utilizar es el de **chacha20-poly1305@openssh.com**. En este caso el tamaño del paquete para cliente y el servidor es de 1578 bytes y 1222 bytes respectivamente.

2.6 Tráfico generado por C4 (4 (iface lo) (detallado)

2 DESARROLLO (PARTE 1)

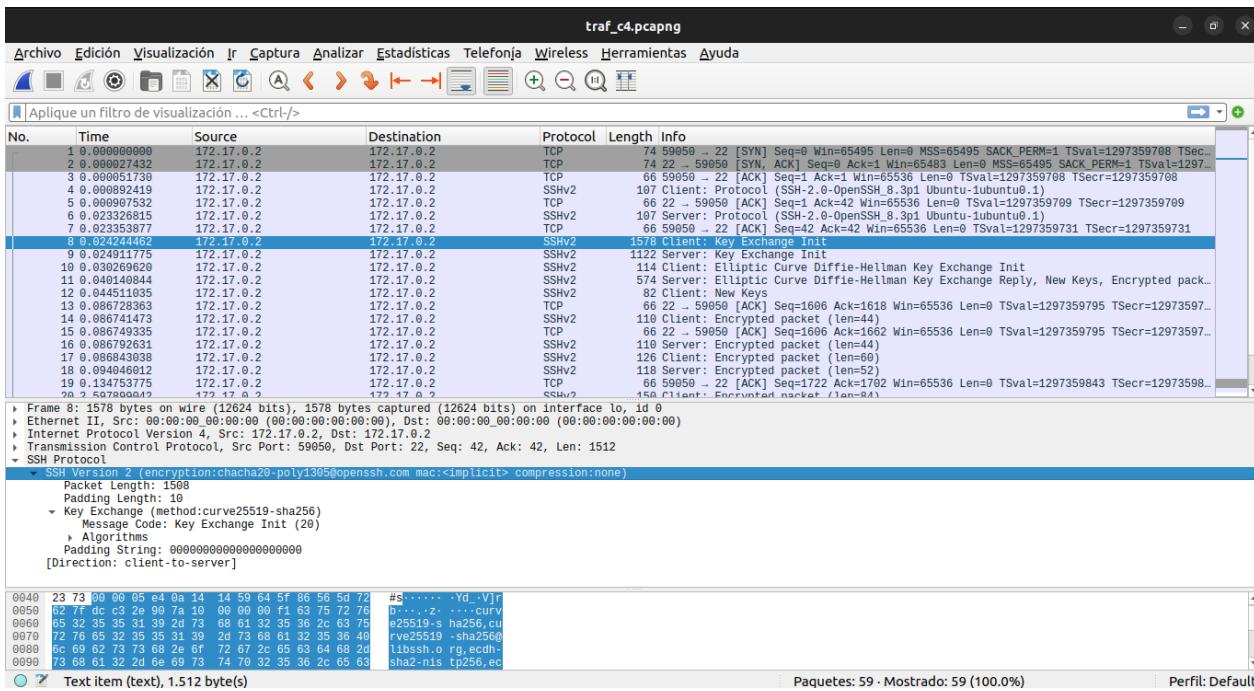


Figura 31: Paquete 3 entre C4 y S1 capturado desde wireshark.

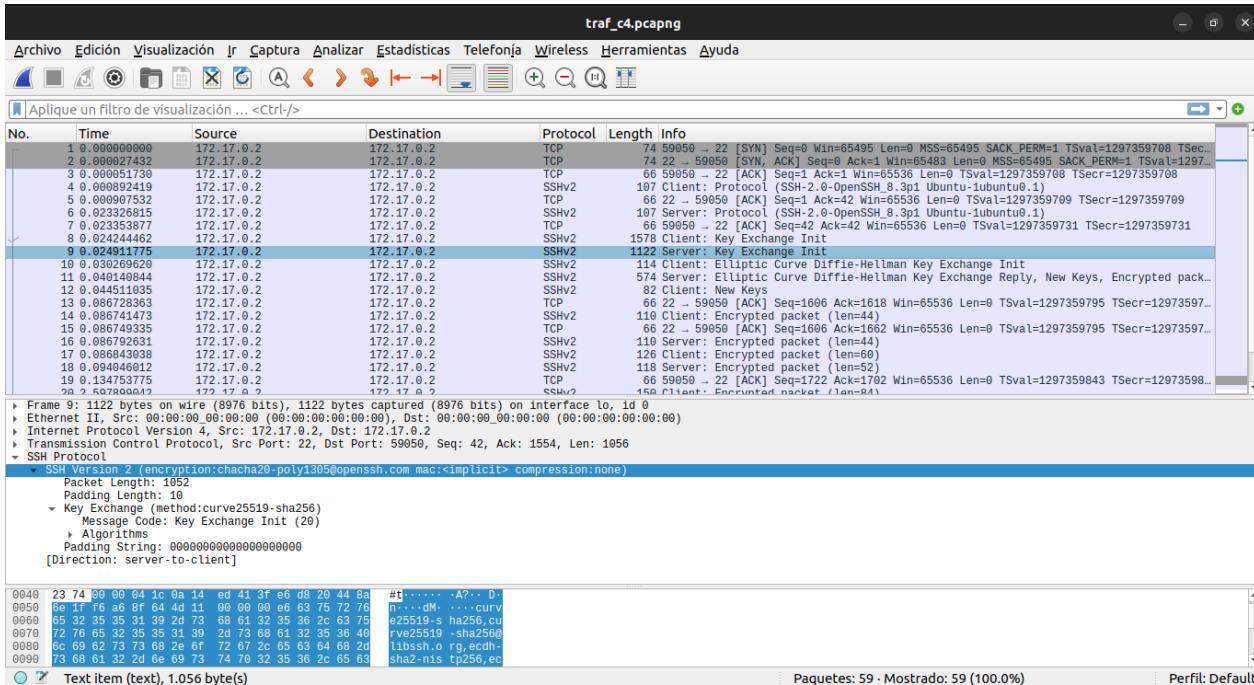


Figura 32: Paquete 4 entre C4 y S1 capturado desde wireshark.

2.7. Diferencia entre C1 y C2

En este caso, existe dos diferencias entre el tráfico de C1 y C2:

- La primera diferencia es que el tamaño de los paquetes Key Exchange Init es distinto, en donde en el caso para C1, tiene un tamaño del paquete de **2034 bytes** y para C2, tiene un tamaño menor a C1 de **1498 bytes**.
- Otra de las diferencias es que la versión del protocolo SSH es distinto, en donde C1 utiliza **SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-8** y C2 utiliza **SSH-2.0-OpenSSH_7.3p1 Ubuntu-1ubuntu0.1**.

2.8. Diferencia entre C2 y C3

Al igual que en el caso anterior, hay dos principales diferencias entre el tráfico de C2 y C3:

- Una de las diferencias es el tamaño de los paquetes Key Exchange Init, donde C2 tiene un tamaño de **1498 bytes**, mayor a la de C3, el cual es de tamaño de **1426 bytes**.
- La segunda diferencia es en las versiones del protocolo SSH, teniendo que C2 utiliza **SSH-2.0-OpenSSH_7.3p1 Ubuntu-1ubuntu0.1** y C3 **SSH-2.0-OpenSSH_7.7p1 Ubuntu-4ubuntu0.3**

2.9. Diferencia entre C3 y C4

Si se diferencian estos tráficos capturados entre C3 y C4, se pueden ver dos diferencias:

- El tamaño de los paquetes Key Exchange Init son distintos, teniendo que el tamaño de C3 es menor a C4, teniendo como tamaño los valores de **1426 bytes** y **1578 bytes** respectivamente.
- Las versiones del protocolo SSH, son distintas entre los dos clientes. En el caso de C3 esta utilizando la versión de **SSH-2.0-OpenSSH_7.7p1 Ubuntu-4ubuntu0.3** y para el caso de C4, utiliza la versión **SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1**

3. Desarrollo (Parte 2)

3.1. Identificación del cliente ssh

Con el fin de identificar el cliente SSH, se procede a analizar la figura 1 entregada, sabiendo que las únicas diferencias entre los clientes, es el tamaño del paquete Key Exchange Init y la versión del protocolo SSH.

Como se puede observar en la figura 1, el tamaño del paquete es de 1578 bytes, por lo que si lo comparamos con las versiones anteriormente analizadas, la única que tiene un tamaño de 1578 bytes es el cliente C4, por lo que se asume que la versión del protocolo SSH es la misma de C4: **SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1**.

3.2. Replicación de tráfico (paso por paso)

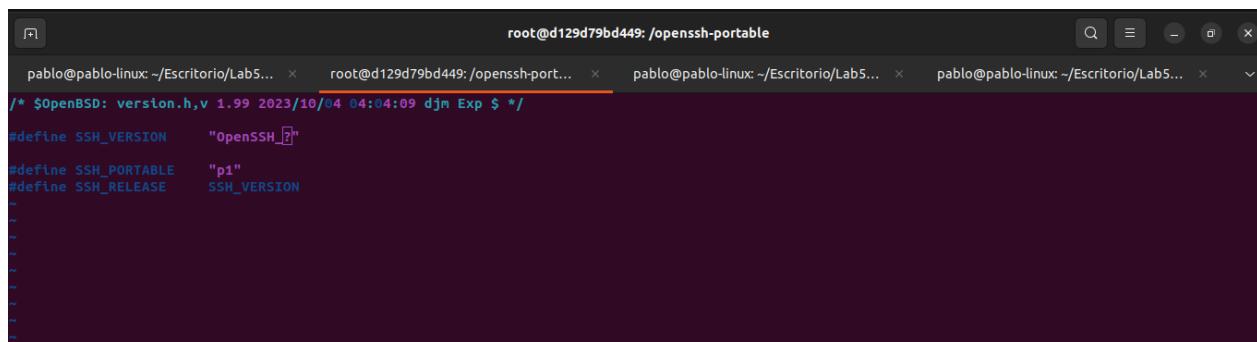
Para poder replicar el tráfico, se procede a utilizar un nuevo contenedor de C4, utilizándolo igualmente como cliente y servidor, para poder modificar la versión del protocolo SSH, capturar el tráfico a través de **tshark** y establecer la conexión con el servidor (S1). Cabe destacar que la ip del servidor cambio a la de 172.17.0.3 por haber creado un nuevo contenedor de C4. El contenedor se ejecuta a través del siguiente comando:

```
sudo docker run --cap-add=NET_RAW --cap-add=NET_ADMIN -it c4 bash
```

Primeramente, se deben instalar herramientas: *autoconf*, *libssl-dev* *zlib1g-dev*, *gcc*, *make*, *git* y *vim*. También, se debe clonar el repositorio <https://github.com/openssh/openssh-portable>, para luego posicionarnos en la carpeta **openssh-portable**, a través de los siguientes comandos:

```
apt update
apt install autoconf libssl-dev zlib1g-dev gcc make git vim
git clone https://github.com/openssh/openssh-portable
cd openssh-portable
```

Luego de haber instalado todo lo necesario, se procede a modificar el archivo **version.h**, desde la herramienta de vim, en donde se modifica el valor del parámetro **SSH_VERSION** por '**OpenSSH_?**', se elimina el valor **SSH_PORTABLE** del parámetro **SSH_RELEASE** y se guarda el archivo.



```
pablo@pablo-linux:~/Escritorio/Lab5... × root@d129d79bd449:/openssh-portable × pablo@pablo-linux:~/Escritorio/Lab5... × pablo@pablo-linux:~/Escritorio/Lab5...
root@d129d79bd449:/openssh-portable
/* $OpenBSD: version.h,v 1.99 2023/10/04 04:04:09 djm Exp $ */
#define SSH_VERSION "OpenSSH_?"
#define SSH_PORTABLE "p1"
#define SSH_RELEASE SSH_VERSION
```

Figura 33: Archivo **versión.h** modificado.

Por último, se ejecutan los siguientes comandos:

```
autoreconf
./configure
make
make install
/usr/local/sbin/sshd
```

Ahora se procede a capturar el tráfico generado por el nuevo C4, a través de **tshark** desde otra bash del mismo contenedor, teniendo como captura de tráfico lo siguiente:

```
root@d129d79bd449:/# ssh test@172.17.0.3
The authenticity of host '172.17.0.3 (172.17.0.3)' can't be established.
ED25519 key fingerprint is SHA256:D0/HKrbjUx44KdTJctDVo5Rzjok+AqIbkqdkdBtPjQk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint]): yes
Warning: Permanently added '172.17.0.3' (ED25519) to the list of known hosts.
test@172.17.0.3's password:
test@d129d79bd449:~$ exit
logout
Connection to 172.17.0.3 closed.
```

Figura 34: Comando conexión SSH entre C4 modificado version.h y S1.

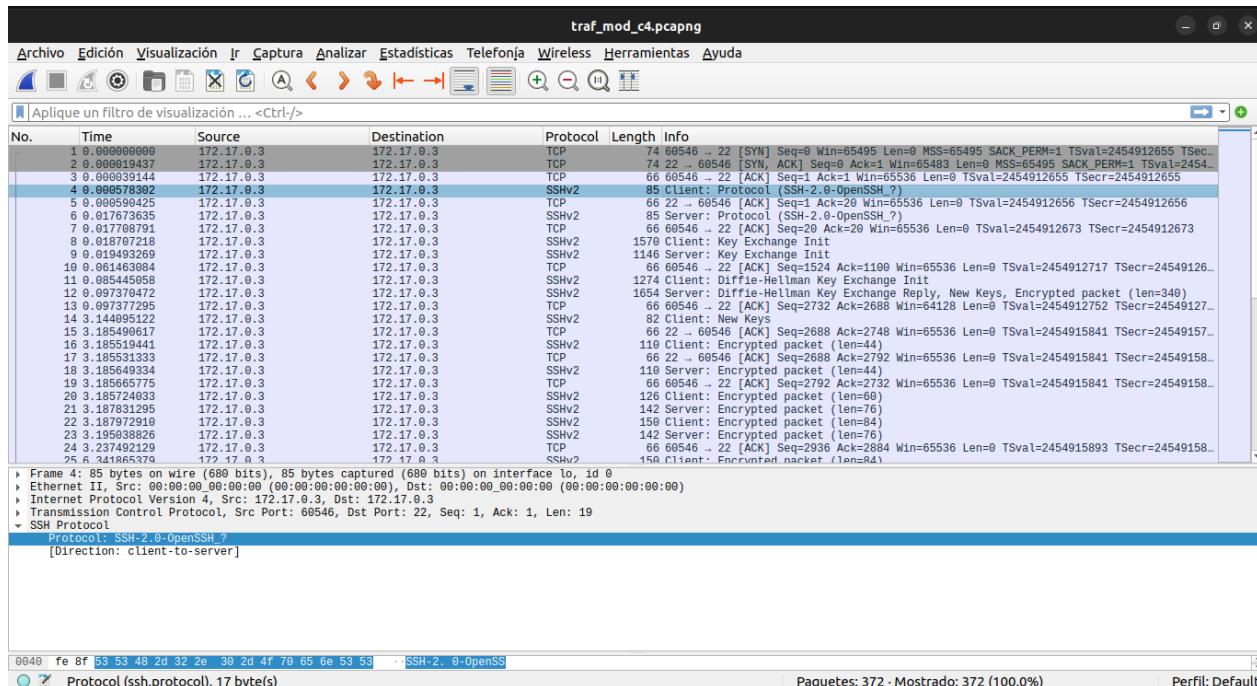


Figura 35: Captura de tráfico con C4 modificado version.h hacia el servidor S1.

Como se puede observar en los primeros paquetes del cliente y el servidor indican que la versión de SSH es la de **SSH-2.0-OpenSSH_?**.

4. Desarrollo (Parte 3)

4.1. Replicación de tráfico (paso por paso)

En este caso, se procede a utilizar el mismo contenedor c4, pero en esta ocasión se modifica el archivo **sshd_config**, en donde se agregan los siguientes parámetros: Ciphers aes128-ctr, HostKeyAlgorithms ecdsa-sha2-nistp256, KexAlgorithms ecdh-sha2-nistp256 y MACs hmac-sha2-256.

```

root@2c6ed43a9988: /openssh-portable
pablo@pablo-linux: ~/Escritorio/Lab5...
pablo@pablo-linux: ~/Escritorio/Lab5...
pablo@pablo-linux: ~/Escritorio/Lab5...

#X11Forwarding no
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
#PrintMotd yes
#PrintLastLog yes
#TCPKeepAlive yes
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none

# override default of no subsystems
Subsystem sftp /usr/libexec/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#   X11Forwarding no
#   AllowTcpForwarding no
#   PermitTTY no
#   ForceCommand cvs server
Ciphers aes128-ctr
HostKeyAlgorithms ecdsa-sha2-nistp256
KexAlgorithms ecdh-sha2-nistp256
MACs hmac-sha2-256
-

```

Figura 36: Archivo sshd_config modificado.

Luego se ejecutan los siguientes comandos, para que los cambios se realicen:

```

autoreconf
./configure
make
make install
/usr/local/sbin/sshd

```

Por último, se procede a capturar el tráfico entre el contenedor C4 modificado y S1, desde otra terminal bash del mismo contenedor a través de **tshark**, obteniendo como resultado la siguiente captura:

```

root@2c6ed43a9988: /openssh-portable# ssh test@172.17.0.2
The authenticity of host '172.17.0.2 (172.17.0.2)' can't be established.
ECDSA key fingerprint is SHA256:3pS8f3rmdrn0TZDWgykm6kjbkVxUWo2aWEha90XD+bA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.17.0.2' (ECDSA) to the list of known hosts.
test@172.17.0.2's password:
test@2c6ed43a9988:~$ exit
logout
Connection to 172.17.0.2 closed.
root@2c6ed43a9988: /openssh-portable# 

```

Figura 37: Comando conexión SSH entre C4 modificado sshd_config y S1.

4.1 Replicación de tráfico (paso por paso)

4 DESARROLLO (PARTE 3)

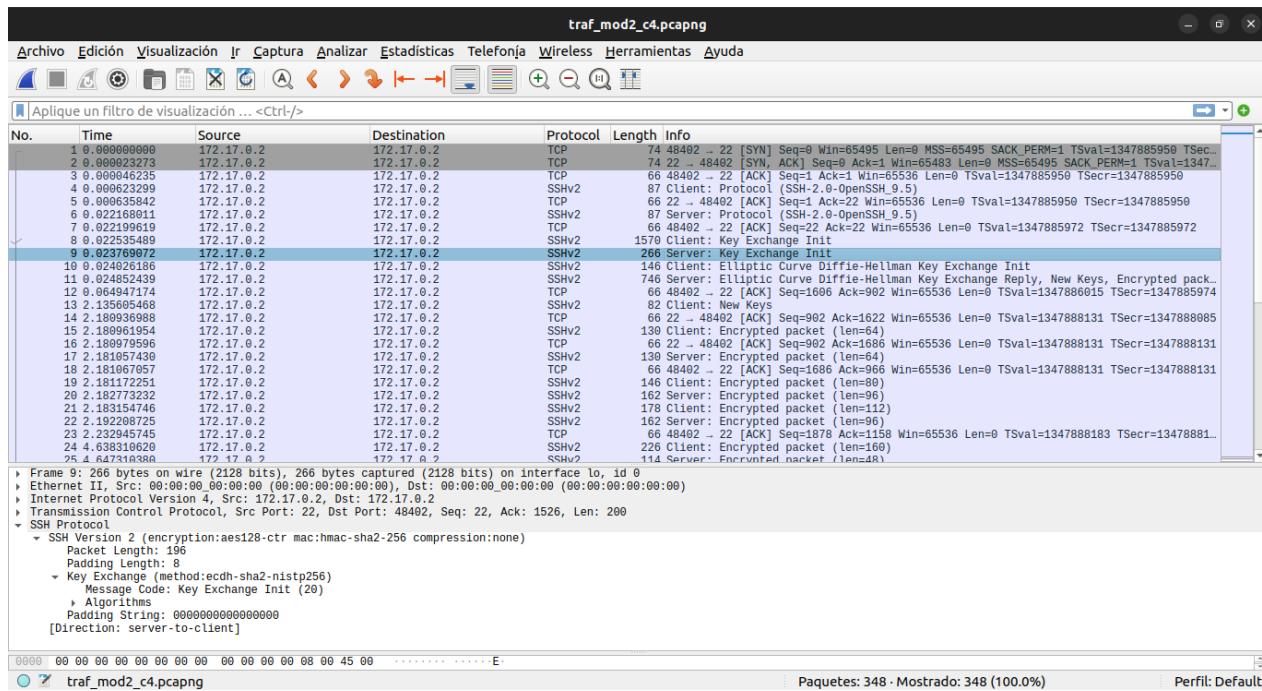


Figura 38: Captura de trafico con C4 modificado sshd, configurado para el servidor S1. @

Como se puede observar, el tamaño del paquete Key Exchange Init del servidor es de **266 bytes**, lo cual es menor a 300 bytes.

Conclusiones y comentarios

En el presente informe de laboratorio, se llevó a cabo una serie de actividades relacionadas con la seguridad en conexiones SSH utilizando contenedores Docker. Se crearon cuatro contenedores, tres como clientes (C1, C2, C3) y uno como servidor (S1), cada uno con diferentes versiones de Ubuntu y OpenSSH. El objetivo principal fue analizar el tráfico SSH generado por cada cliente y realizar modificaciones en la versión del protocolo SSH y en la configuración del servidor para observar sus efectos.

En la primera parte del informe, se detalló la creación de los contenedores y la instalación de OpenSSH en cada uno. Se realizaron conexiones SSH desde los clientes hacia el servidor, capturando el tráfico de red y analizando el intercambio de claves. Se identificaron diferencias en el tráfico generado por cada cliente, como los tamaños de los paquetes Key Exchange Init y las versiones del protocolo SSH utilizadas.

En la segunda parte, se identificó el cliente SSH que generó un tráfico específico proporcionado en una imagen. Se compararon las características del tráfico con los registros obtenidos en la primera parte del informe y se determinó que correspondía al cliente C4.

En la tercera parte, se replicó el tráfico generado por C4, realizando modificaciones en la versión del protocolo SSH a través de la modificación del archivo `version.h`. Se logró replicar el tráfico con éxito, confirmando la identificación del cliente SSH.

Finalmente, en la última parte del informe, se modificó la configuración del servidor SSH (`sshd_config`) para cambiar el patrón de tráfico hasta que el tamaño del paquete Key Exchange Init fuera menor a 300 bytes. Se logró este objetivo mediante la modificación de los algoritmos de cifrado y claves.

En conclusión, el informe abordó de manera detallada la configuración, análisis y modificación de conexiones SSH utilizando contenedores Docker. Se aplicaron técnicas para identificar clientes SSH, replicar tráfico y modificar la configuración del servidor para mejorar la seguridad de las conexiones.

GitHub

- <https://github.com/THELUXE1234/labcripto5>