Informe Laboratorio 5

Sección 2

Vania Vergara
e-mail: vania.vergara@mail.udp.cl

Junio de 2023

Índice

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 $\rightarrow$ S1
  - C2 $\rightarrow$ S1
  - C3 $\rightarrow$ S1
  - C4 $\rightarrow$ 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?

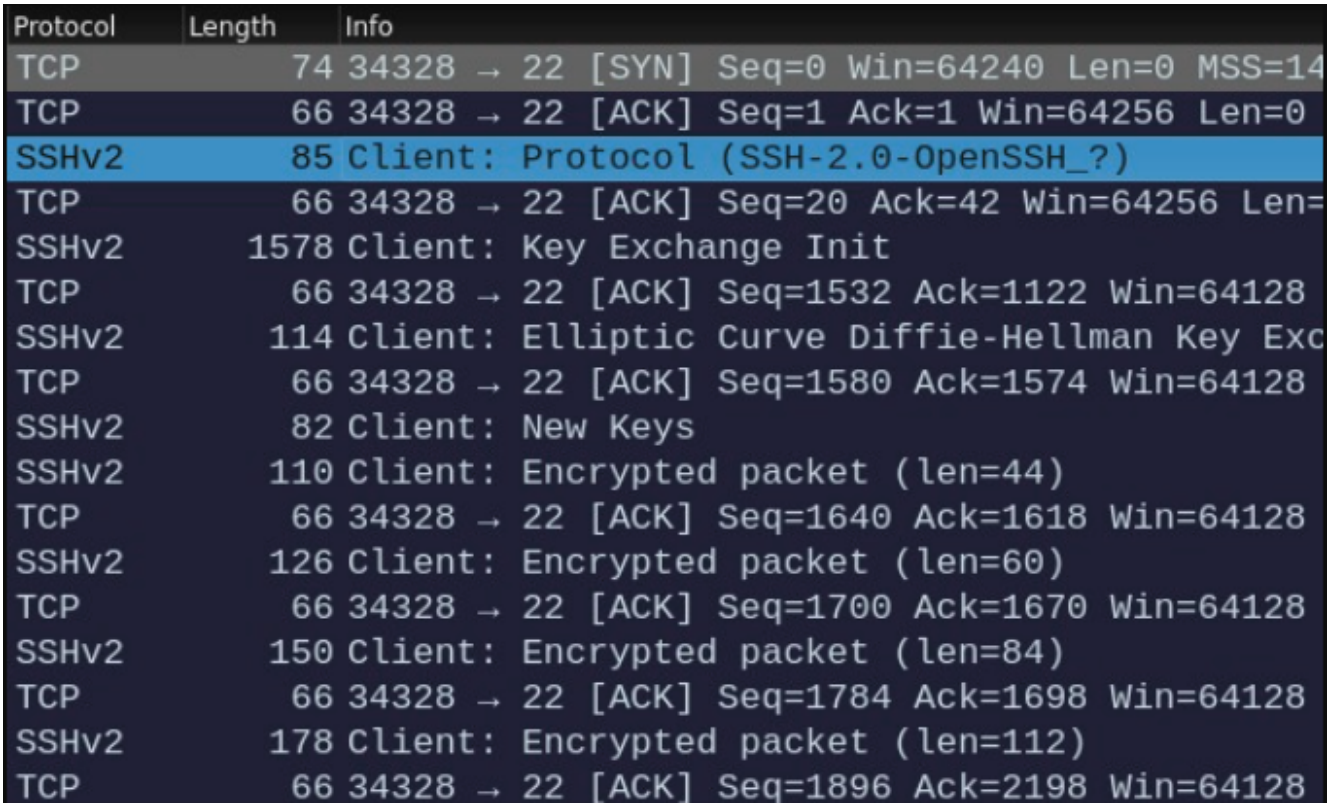

Figura 1: Tráfico generado del informante

Replique 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.

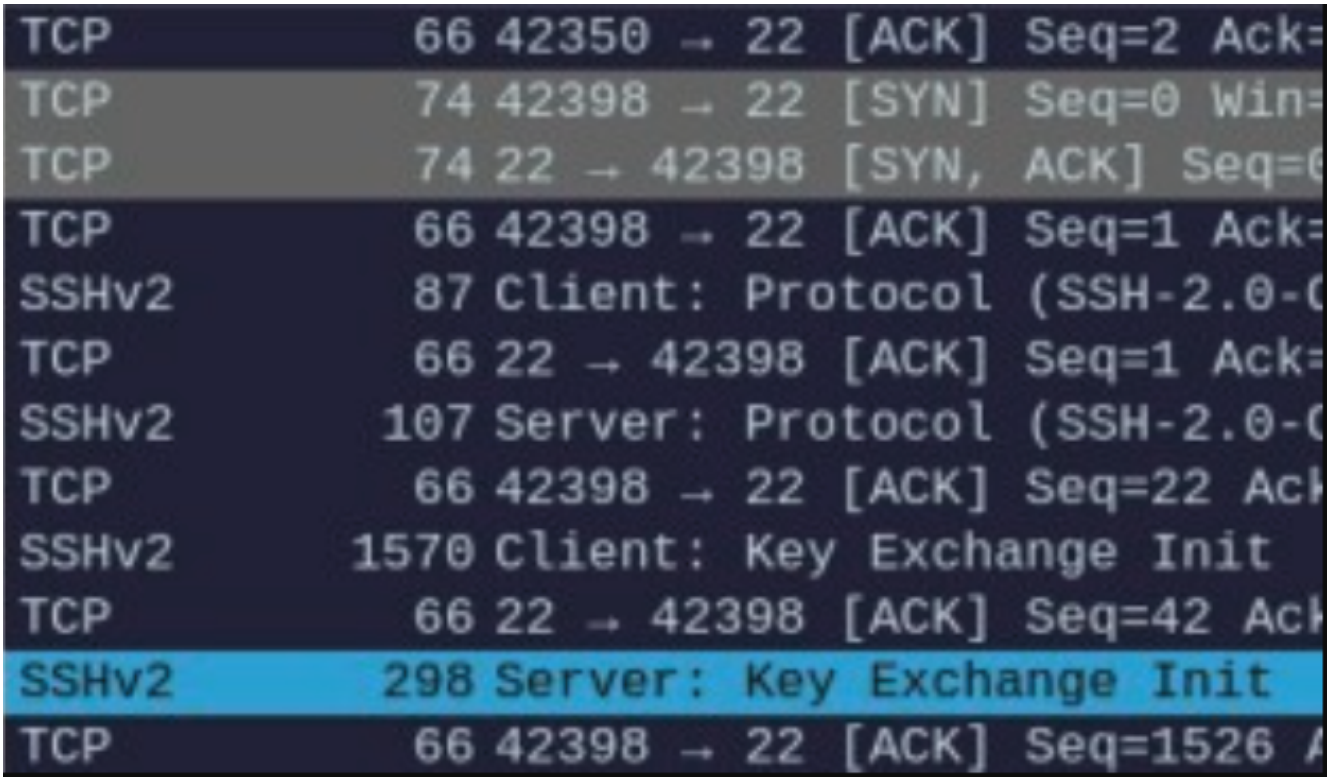

Figura 2: Captura del Key Exchange

2. Desarrollo (Parte 1)

2.1. Códigos de cada Dockerfile

2.1.1. S1

```
FROM ubuntu:20.10

RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ groovy main restricted universe multiverse" > /etc/apt/sources.list
RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ groovy-updates main restricted universe multiverse" >> /etc/apt/sources.list
RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ groovy-security main restricted universe multiverse" >> /etc/apt/sources.list
RUN apt-get update && apt-get upgrade -y
RUN apt-get install -y openssh-client openssh-server sshpass

CMD ["/bin/bash"]
```

Figura 3: Dockerfile S1

2.1.2. C1

```
FROM ubuntu:14.10

RUN sed --in-place='.bak' \
  's/\(archive\|security\)\.ubuntu\.com/old-releases\.ubuntu\.com/g' \
  /etc/apt/sources.list
RUN apt-get update && apt-get upgrade -y
RUN apt-get install -y openssh-client openssh-server

CMD ["/bin/bash"]
```

Figura 4: Dockerfile C1

2.1.3. C2

```
FROM ubuntu:16.10

RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ yakkety main restricted universe multiverse" > /etc/apt/sources.list
RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ yakkety-updates main restricted universe multiverse" >> /etc/apt/sources.list
RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ yakkety-security main restricted universe multiverse" >> /etc/apt/sources.list
RUN apt-get update && apt-get upgrade -y
RUN apt-get install -y openssh-client openssh-server sshpass

CMD ["/bin/bash"]
```

Figura 5: Dockerfile C2

2.1.4. C3

```
FROM ubuntu:18.10

RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ cosmic main restricted universe multiverse" > /etc/apt/sources.list
RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ cosmic-updates main restricted universe multiverse" >> /etc/apt/sources.list
RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ cosmic-security main restricted universe multiverse" >> /etc/apt/sources.list
RUN apt-get update && apt-get upgrade -y
RUN apt-get install -y openssh-client openssh-server sshpass

CMD ["/bin/bash"]
```

Figura 6: Dockerfile C3

2.1.5. C4

```
FROM ubuntu:20.10

RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ groovy main restricted universe multiverse" > /etc/apt/sources.list
RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ groovy-updates main restricted universe multiverse" >> /etc/apt/sources.list
RUN echo "deb http://old-releases.ubuntu.com/ubuntu/ groovy-security main restricted universe multiverse" >> /etc/apt/sources.list
RUN apt-get update && apt-get upgrade -y
RUN apt-get install -y openssh-client openssh-server sshpass

CMD ["/bin/bash"]
```

Figura 7: Dockerfile C3

2.2. Creación de las credenciales para S1

```
root@672d7c2ef0d5:/# adduser test
Adding user `test' ...
Adding new group `test' (1000) ...
Adding new user `test' (1000) with group `test' ...
The home directory `/home/test' already exists.  Not copying from `/etc/skel'.
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for test
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

Figura 8: Creación usuario test

2.3. Tráfico generado por C1 (detallado)

Se observa que el primer paquete generado por el cliente tiene un tamaño de 100 bytes y utiliza la versión 6.6.1p1 del protocolo. Durante el inicio del intercambio de llaves, se observa que esta versión del protocolo utiliza el cifrado aes128-ctr y el algoritmo hmac-sha1-etm, con un tamaño de 1964 bytes más 8 bytes de padding. Además, el intercambio de llaves se realiza mediante curve25519-sha256, con un algoritmo de longitud 212.

En el siguiente paquete, se indica el inicio del intercambio de claves de difie-hellman elíptico, donde se proporciona la clave pública efímera del cliente ECDH, que consta de 32 bytes. El protocolo tiene un paquete de 44 bytes más 6 de padding.

Posteriormente, en el siguiente paquete se menciona que se están utilizando nuevas llaves, donde el protocolo tiene un tamaño de 12 bytes y un padding de 10 bytes.

Por último, los cinco paquetes restantes representan el intercambio de paquetes cifrados, cuyos tamaños son respectivamente de 56, 72, 152, 128 y 400 bytes.

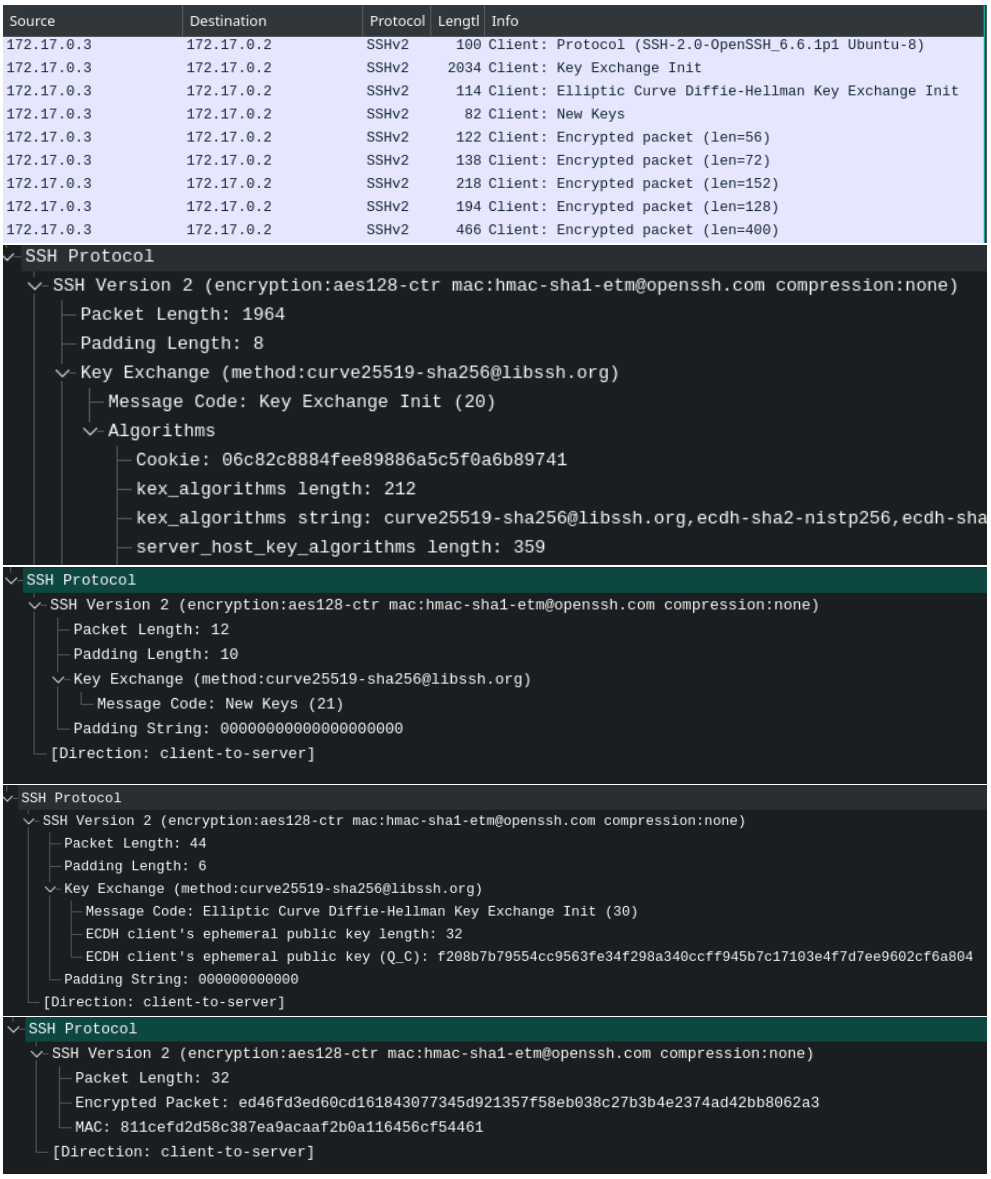

Figura 9: Tráfico del C1

2.4. Tráfico generado por C2 (detallado)

En el primer paquete generado por el cliente, se ve que utiliza la versión 7.3p1. En el siguiente paquete se observa durante el inicio de intercambio de llaves, el protocolo utiliza el cifrado chacha20-poly1305, donde el tamaño de paquetes es de 1428 bytes más 11 bytes de padding, además se utiliza el método curve25519-sha256 el cual tiene longitud 286.

En el siguiente paquete, se indica el inicio del intercambio de claves de difie-hellman elíptico, donde se proporciona la clave pública efímera del cliente ECDH, que consta de 32 bytes, donde el paquete es de 44 bytes más 6 de padding.

Después se menciona que se están utilizando nuevas llaves, donde el protocolo tiene un tamaño de 12 bytes y un padding de 10 bytes.

Por último, se inicia el intercambio de mensajes cifrados, donde los paquetes tienen un tamaño de 44, 60, 84, 112 y 376 btyes respectivamente.

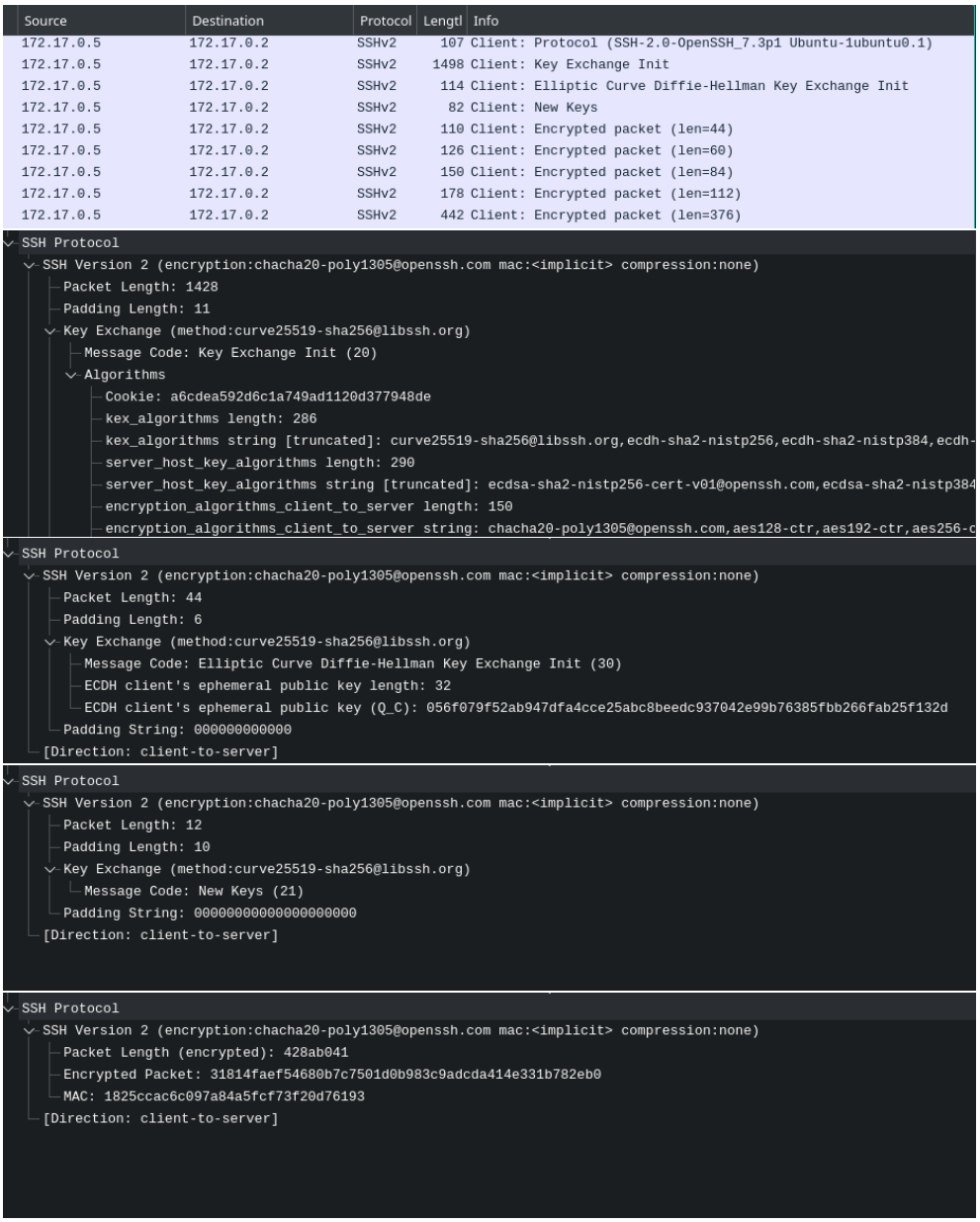

Figura 10: Tráfico del C2

2.5. Tráfico generado por C3 (detallado)

En el primer paquete generado por el cliente, se ve que utiliza la versión 7.7p1. En el siguiente paquete se observa durante el inicio de intercambio de llaves, el protocolo utiliza el cifrado chacha20-poly1305, donde el tamaño de paquetes es de 1356 bytes más 5 bytes de padding, además se utiliza el método curve25519-sha256 el cual tiene longitud 286.

En el siguiente paquete, se indica el inicio del intercambio de claves de difie-hellman elíptico, donde se proporciona la clave pública efímera del cliente ECDH, que consta de 32 bytes, donde el paquete es de 44 bytes más 6 de padding.

Después se menciona que se están utilizando nuevas llaves, donde el protocolo tiene un tamaño de 12 bytes y un padding de 10 bytes.

Por último, se inicia el intercambio de mensajes cifrados, donde los paquetes tienen un tamaño de 44, 60, 84, 112 y 376 btyes respectivamente.

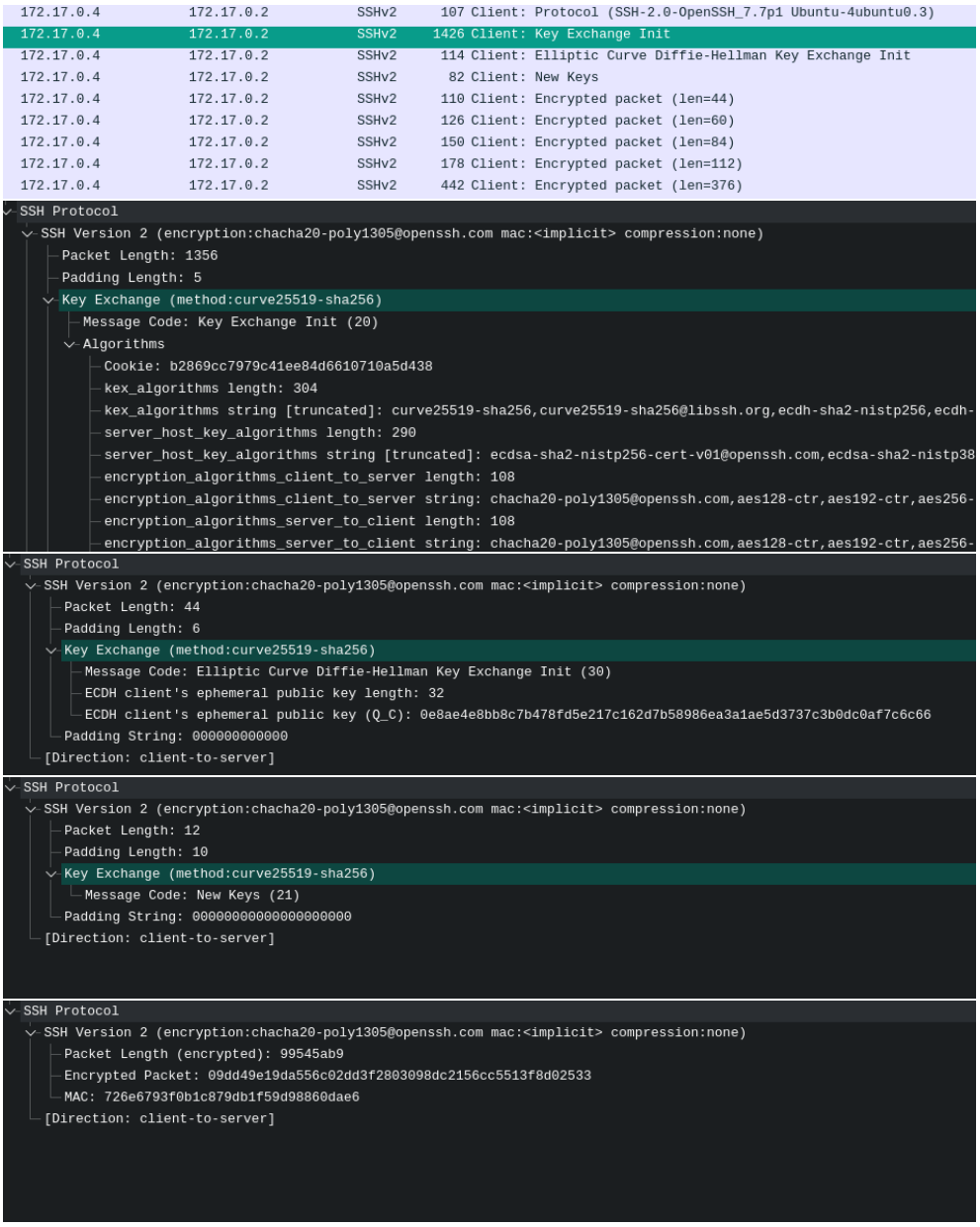

Figura 11: Tráfico del C3

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

En el primer paquete generado por el cliente, se ve que utiliza la versión 8.3p1. En el siguiente paquete se observa durante el inicio de intercambio de llaves, el protocolo utiliza el cifrado chacha20-poly1305, donde el tamaño de paquetes es de 1508 bytes más 10 bytes de padding, además se utiliza el método curve25519-sha256 el cual tiene longitud 241.

En el siguiente paquete, se indica el inicio del intercambio de claves de difie-hellman elípti-

co, donde se proporciona la clave pública efímera del cliente ECDH, que consta de 32 bytes, donde el paquete es de 44 bytes más 6 de padding.

Después se menciona que se están utilizando nuevas llaves, donde el protocolo tiene un tamaño de 12 bytes y un padding de 10 bytes.

Por último, se inicia el intercambio de mensajes cifrados, donde los paquetes tienen un tamaño de 44, 60, 84, 112 y 376 btyes respectivamente.

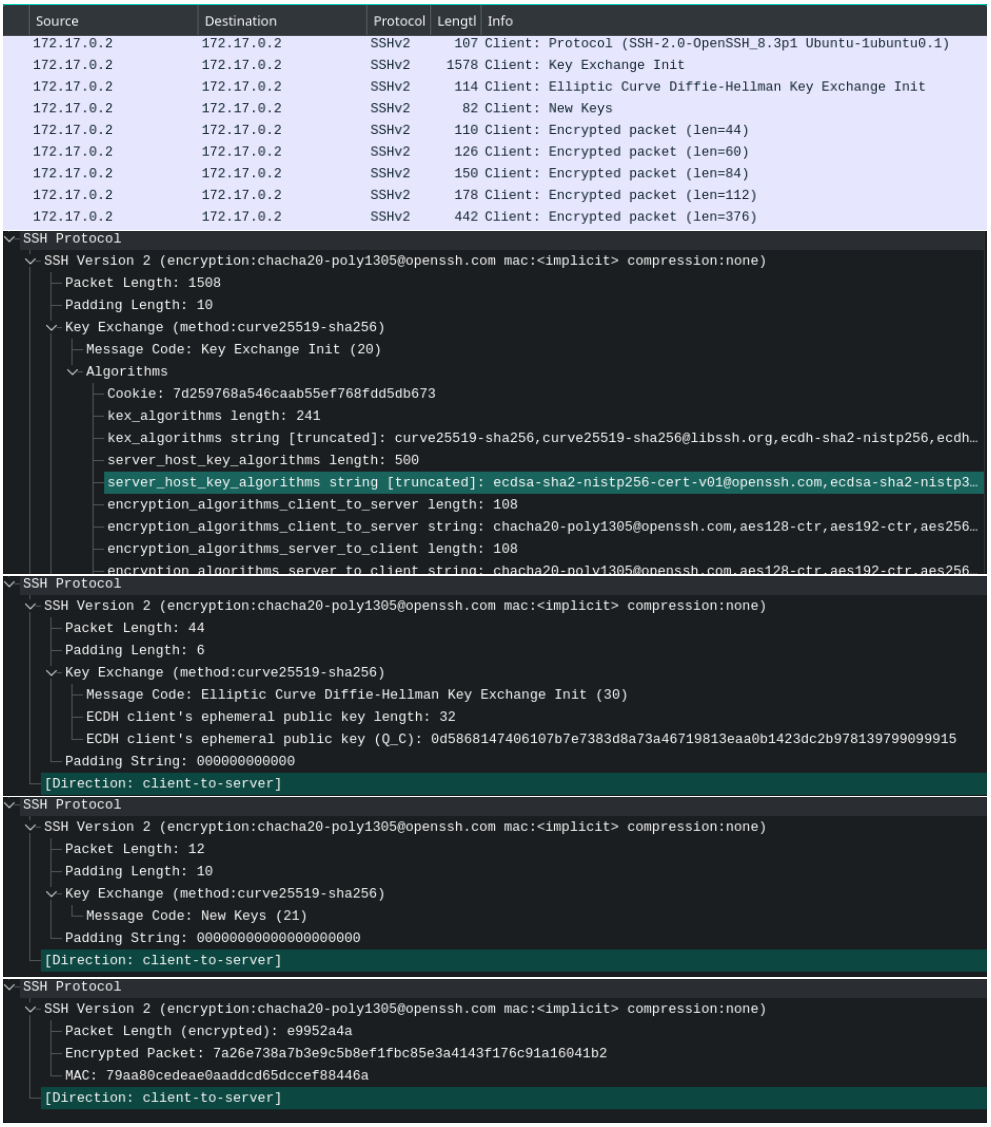

Figura 12: Tráfico del C2

2.7. Diferencia entre C1 y C2

Existen varias diferencias entre los clientes al iniciar el proceso de intercambio de llaves. En primer lugar, los paquetes presentan diferentes tamaños, la versión del protocolo son distintas (6.6p1 y 7.3.p1) y también se emplea un cifrado distinto. En el caso de C1, se utiliza el cifrado aes128-ctr, mientras que en C2 se utiliza chacha20-poly1305. Por último, al intercambiar paquetes encriptados, todos ellos poseen tamaños distintos.

2.8. Diferencia entre C2 y C3

Las diferencias que existen entre los paquetes es primeramente las versiones utilizadas por los clientes, 7.3p1 (C2) y 7.7p1 (C3). Después cuando se inicia el intercambio de llaves, los tamaños de los paquetes son distintos: 1498 bytes de C2 y 1426 bytes de C3, donde los padding también son distintos, con 11 y 5 respectivamente. También se ve que cambia en los tamaños que ejecuta el algoritmo (kex_algorithms_length).

2.9. Diferencia entre C3 y C4

Las diferencias que existen entre los paquetes es primeramente las versiones utilizadas por los clientes, 7.7p1 (C2) y 8.3p1 (C3). Después cuando se inicia el intercambio de llaves, los tamaños de los paquetes son distintos: 1426 bytes de C3 y 1578 bytes de C4, donde los padding también son distintos, con 5 y 10 respectivamente. También se ve que cambia en los tamaños que ejecuta el algoritmo (kex_algorithms_length).

3. Desarrollo (Parte 2)

3.1. Identificación del cliente ssh

El cliente corresponde a C4, debido al tamaño del paquete cuando se inicia el proceso de intercambio de llaves.

3.2. Replicación de tráfico (paso por paso)

4. Desarrollo (Parte 3)

4.1. Replicación de tráfico (paso por paso)

Conclusiones y comentarios