

UNIVERSIDAD DIEGO PORTALES

SANTIAGO, CHILE, MAYO DE 2023



udp

Facultad de Ingeniería y Ciencias

Departamento de Informática y Telecomunicaciones

Criptografía y Ciberseguridad en Redes:

Laboratorio 5

Profesor: Nicolás Boettcher
Ayudantes: Brayan Espina
Mirko Babic
Alumno: Nicolás Chirino

Índice

1. Descripción de actividades	3
2. Estado del arte	3
3. Desarrollo (Parte 1)	3
3.1. Códigos de cada Dockerfile	3
3.1.1. C1	3
3.1.2. C2	4
3.1.3. C3	5
3.1.4. C4	5
3.1.5. S1	6
3.2. Creación de credenciales para S1	7
3.3. Tráfico generado por C1 (detallado)	7
3.4. Tráfico generado por C2 (detallado)	11
3.5. Tráfico generado por C3 (detallado)	13
3.6. Tráfico generado por C4 (iface io) (detallado)	15
3.7. Diferencia entre C1 y C2	17
3.8. Diferencia entre C2 y C3	17
3.9. Diferencia entre C3 y C4	18
4. Desarrollo (Parte 2)	18
4.1. Identificación del cliente SSH	18
4.2. Replicación del tráfico (paso por paso)	18
5. Desarrollo (Parte 3)	19
5.1. Replicación del tráfico (paso por paso)	19
6. Conclusión	20

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:

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. Estado del arte

Antes de comenzar el informe, quisiera aclarar que todos los códigos, capturas de wireshark y capturas de pantallas se encuentran en el repositorio <https://github.com/nicobrhc/cryptography>, específicamente en el laboratorio-5. Menciono esto para que tengan facilidad de probar los códigos utilizados y ver los resultados obtenidos (que puede ser mejor que una foto del informe).

3. Desarrollo (Parte 1)

3.1. Códigos de cada Dockerfile

3.1.1. C1

El dockerfile de este contenedor es el siguiente:

```
FROM ubuntu:14.10

COPY sources.list /etc/apt/sources.list

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y openssh-client

CMD ["bash"]
```

Este contenedor creará una imagen de ubuntu 14.10, copiará el archivo “sources.list” (necesario usar apt con repositorios antiguos) y finalmente abre una terminal bash. Para correr el contenedor se debe usar el siguiente comando:

```
docker build -t lab5c1 .  
docker run -it -rm lab5c1
```

El archivo “sources.list” respectivo es el siguiente:

```
deb http://old-releases.ubuntu.com/ubuntu utopic main multiverse restricted universe  
deb http://old-releases.ubuntu.com/ubuntu utopic-backports main multiverse restricted  
universe  
deb http://old-releases.ubuntu.com/ubuntu utopic-proposed main multiverse restricted  
universe  
deb http://old-releases.ubuntu.com/ubuntu utopic-security main multiverse restricted  
universe  
deb http://old-releases.ubuntu.com/ubuntu utopic-updates main multiverse restricted  
universe
```

3.1.2. C2

El dockerfile de este contenedor es el siguiente:

```
FROM ubuntu:16.10  
  
COPY sources.list /etc/apt/sources.list  
  
ENV DEBIAN_FRONTEND=noninteractive  
  
RUN apt-get update && apt-get install -y openssh-client  
  
CMD ["bash"]
```

Este contenedor creará una imagen de ubuntu 16.10, copiará el archivo “sources.list” (necesario usar apt con repositorios antiguos) y finalmente abre una terminal bash. Para correr el contenedor se debe usar el siguiente comando:

```
docker build -t lab5c2 .  
docker run -it -rm lab5c2
```

El archivo “sources.list” respectivo es el siguiente:

```
deb http://old-releases.ubuntu.com/ubuntu yakkety main multiverse restricted universe  
deb http://old-releases.ubuntu.com/ubuntu yakkety-backports main multiverse restricted  
universe  
deb http://old-releases.ubuntu.com/ubuntu yakkety-proposed main multiverse restricted
```

```
universe
deb http://old-releases.ubuntu.com/ubuntu yakkety-security main multiverse restricted
universe
deb http://old-releases.ubuntu.com/ubuntu yakkety-updates main multiverse restricted
universe
```

3.1.3. C3

```
FROM ubuntu:18.10

COPY sources.list /etc/apt/sources.list

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y openssh-client

CMD ["bash"]
```

Este contenedor creará una imagen de ubuntu 18.10, copiará el archivo “sources.list” (necesario usar apt con repositorios antiguos) y finalmente abre una terminal bash. Para correr el contenedor se debe usar el siguiente comando:

```
docker build -t lab5c2 .
docker run -it --rm lab5c2
```

El archivo “sources.list” respectivo es el siguiente:

```
deb http://old-releases.ubuntu.com/ubuntu cosmic main multiverse restricted universe
deb http://old-releases.ubuntu.com/ubuntu cosmic-backports main multiverse restricted
universe
deb http://old-releases.ubuntu.com/ubuntu cosmic-proposed main multiverse restricted
universe
deb http://old-releases.ubuntu.com/ubuntu cosmic-security main multiverse restricted
universe
deb http://old-releases.ubuntu.com/ubuntu cosmic-updates main multiverse restricted
universe
```

3.1.4. C4

```
FROM ubuntu:20.10

COPY sources.list /etc/apt/sources.list

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y openssh-client
```

```
CMD ["bash"]
```

Este contenedor creará una imagen de ubuntu 18.10, copiará el archivo “sources.list” (necesario usar apt con repositorios antiguos) y finalmente abre una terminal bash. Para correr el contenedor se debe usar el siguiente comando:

```
docker build -t lab5c2 .  
docker run -it -rm lab5c2
```

El archivo “sources.list” respectivo es el siguiente:

```
deb http://old-releases.ubuntu.com/ubuntu groovy main multiverse restricted universe  
deb http://old-releases.ubuntu.com/ubuntu groovy-backports main multiverse restricted  
universe  
deb http://old-releases.ubuntu.com/ubuntu groovy-proposed main multiverse restricted  
universe  
deb http://old-releases.ubuntu.com/ubuntu groovy-security main multiverse restricted  
universe  
deb http://old-releases.ubuntu.com/ubuntu groovy-updates main multiverse restricted  
universe
```

3.1.5. S1

```
FROM ubuntu:20.10  
  
COPY sources.list /etc/apt/sources.list  
  
RUN apt update && apt install openssh-server sudo -y  
  
RUN useradd -rm -d /home/ubuntu -s /bin/bash -g root -G sudo -u 1000 test  
  
RUN echo 'test:test' | chpasswd  
  
RUN service ssh start  
  
EXPOSE 22  
  
CMD ["/usr/sbin/sshd", "-D"]
```

Este contenedor creará una imagen de ubuntu 18.10, copiará el archivo “sources.list” (necesario usar apt con repositorios antiguos) y finalmente abre una terminal bash. Para correr el contenedor se debe usar el siguiente comando:

```
docker build -t lab5c2 .  
docker run -it -rm lab5c2
```

El archivo “sources.list” respectivo es el siguiente:

```
deb http://old-releases.ubuntu.com/ubuntu groovy main multiverse restricted universe
deb http://old-releases.ubuntu.com/ubuntu groovy-backports main multiverse restricted
universe
deb http://old-releases.ubuntu.com/ubuntu groovy-proposed main multiverse restricted
universe
deb http://old-releases.ubuntu.com/ubuntu groovy-security main multiverse restricted
universe
deb http://old-releases.ubuntu.com/ubuntu groovy-updates main multiverse restricted
universe
```

3.2. Creación de credenciales para S1

Tal como se aprecia en el dockerfile del contenedor C4/S1, para crear las credenciales de este se ejecutan los siguientes comandos:

```
RUN useradd -rm -d /home/ubuntu -s /bin/bash -g root -G sudo -u 1000 test
RUN echo 'test:test' | chpasswd
```

De esta forma, se crea el usuario “test” y la contraseña “test”.

3.3. Tráfico generado por C1 (detallado)

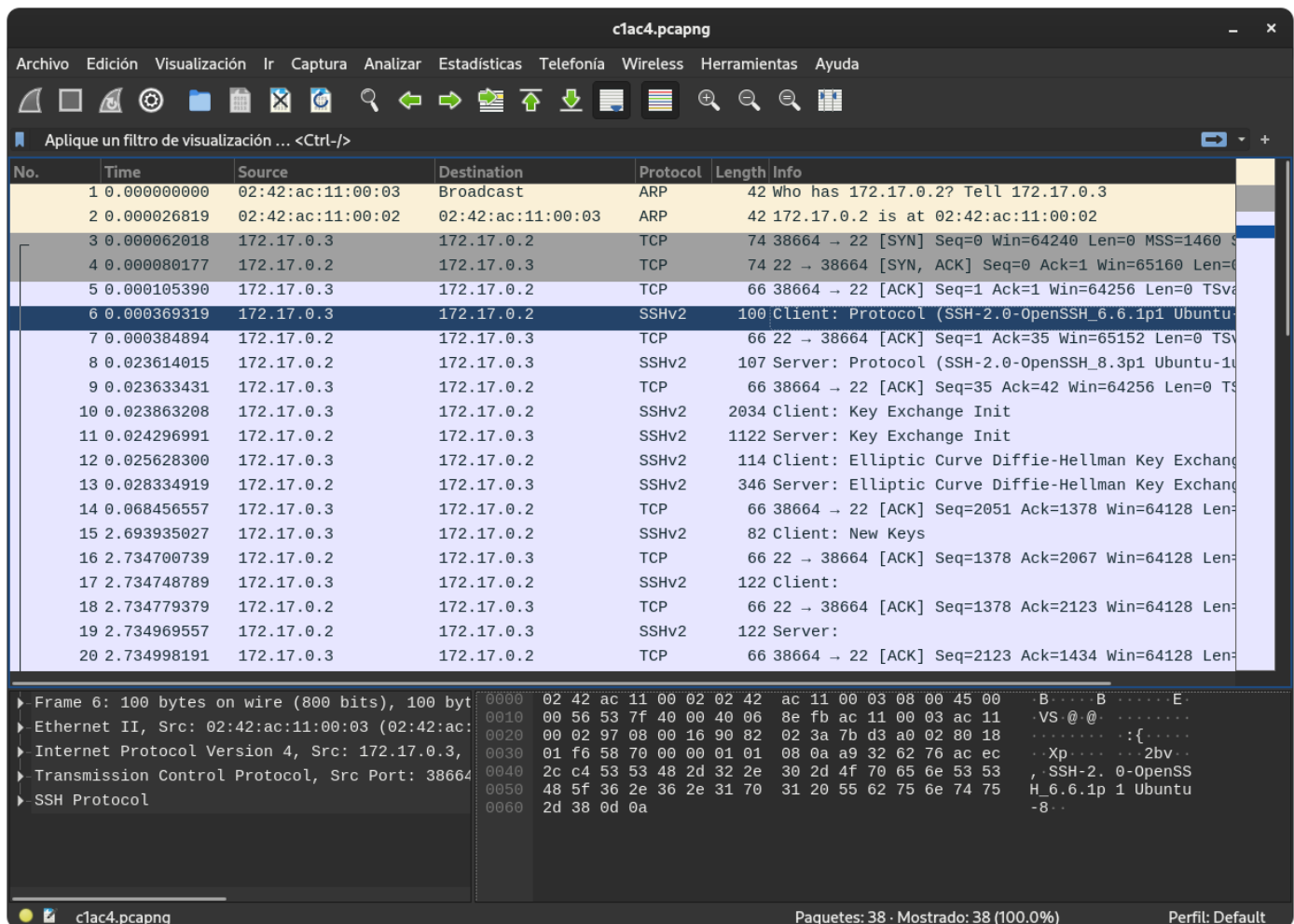
Una vez ejecutado el comando de docker run, se abre la terminal de la imagen ubuntu, en donde se ejecutará el siguiente comando y posteriormente se saldrá sin ejecutar más comandos:

```
ssh test@172.17.0.2
```

Donde 172.17.0.2 es la IP del contenedor S1 (servidor). Esto se aprecia en la siguiente imagen:

```
test@394211c257f1: ~  
docker run -it --rm lab5c4  
test@394211c257f1: ~  
12:16:56  
root@c0b31282243e:/# 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 26:96:64:92:24:25:ef:ef:e4:2e:c2:dc:08:be:78:16.  
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 6.3.6-200.fc38.x86_64 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: Sat Jun 10 16:15:12 2023 from 172.17.0.3  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
test@394211c257f1:~$
```

Al recibir los paquetes del contenedor S1 con wireshark desde nuestra máquina local, se tiene el siguiente resultado:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	02:42:ac:11:00:03	Broadcast	ARP	42	Who has 172.17.0.2? Tell 172.17.0.3
2	0.000026819	02:42:ac:11:00:02	02:42:ac:11:00:03	ARP	42	172.17.0.2 is at 02:42:ac:11:00:02
3	0.000062018	172.17.0.3	172.17.0.2	TCP	74	38664 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
4	0.000080177	172.17.0.2	172.17.0.3	TCP	74	22 → 38664 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
5	0.000105390	172.17.0.3	172.17.0.2	TCP	66	38664 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSv
6	0.000369319	172.17.0.3	172.17.0.2	SSHv2	100	Client: Protocol (SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-
7	0.000384894	172.17.0.2	172.17.0.3	TCP	66	22 → 38664 [ACK] Seq=1 Ack=35 Win=65152 Len=0 TSv
8	0.023614015	172.17.0.2	172.17.0.3	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1u
9	0.023633431	172.17.0.3	172.17.0.2	TCP	66	38664 → 22 [ACK] Seq=35 Ack=42 Win=64256 Len=0 TS
10	0.023863208	172.17.0.3	172.17.0.2	SSHv2	2034	Client: Key Exchange Init
11	0.024296991	172.17.0.2	172.17.0.3	SSHv2	1122	Server: Key Exchange Init
12	0.025628300	172.17.0.3	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchang
13	0.028334919	172.17.0.2	172.17.0.3	SSHv2	346	Server: Elliptic Curve Diffie-Hellman Key Exchang
14	0.068456557	172.17.0.3	172.17.0.2	TCP	66	38664 → 22 [ACK] Seq=2051 Ack=1378 Win=64128 Len=
15	2.693935027	172.17.0.3	172.17.0.2	SSHv2	82	Client: New Keys
16	2.734700739	172.17.0.2	172.17.0.3	TCP	66	22 → 38664 [ACK] Seq=1378 Ack=2067 Win=64128 Len=
17	2.734748789	172.17.0.3	172.17.0.2	SSHv2	122	Client:
18	2.734779379	172.17.0.2	172.17.0.3	TCP	66	22 → 38664 [ACK] Seq=1378 Ack=2123 Win=64128 Len=
19	2.734969557	172.17.0.2	172.17.0.3	SSHv2	122	Server:
20	2.734998191	172.17.0.3	172.17.0.2	TCP	66	38664 → 22 [ACK] Seq=2123 Ack=1434 Win=64128 Len=

Frame 6: 100 bytes on wire (800 bits), 100 byte captured (800 bits) on interface eth0, 100 bytes captured (800 bits) on interface eth0, 100 bytes from 172.17.0.3 to 172.17.0.2 on interface eth0

Ethernet II, Src: 02:42:ac:11:00:03 (02:42:ac:11:00:03), Dst: 02:42:ac:11:00:02 (02:42:ac:11:00:02)

Internet Protocol Version 4, Src: 172.17.0.3, Dst: 172.17.0.2

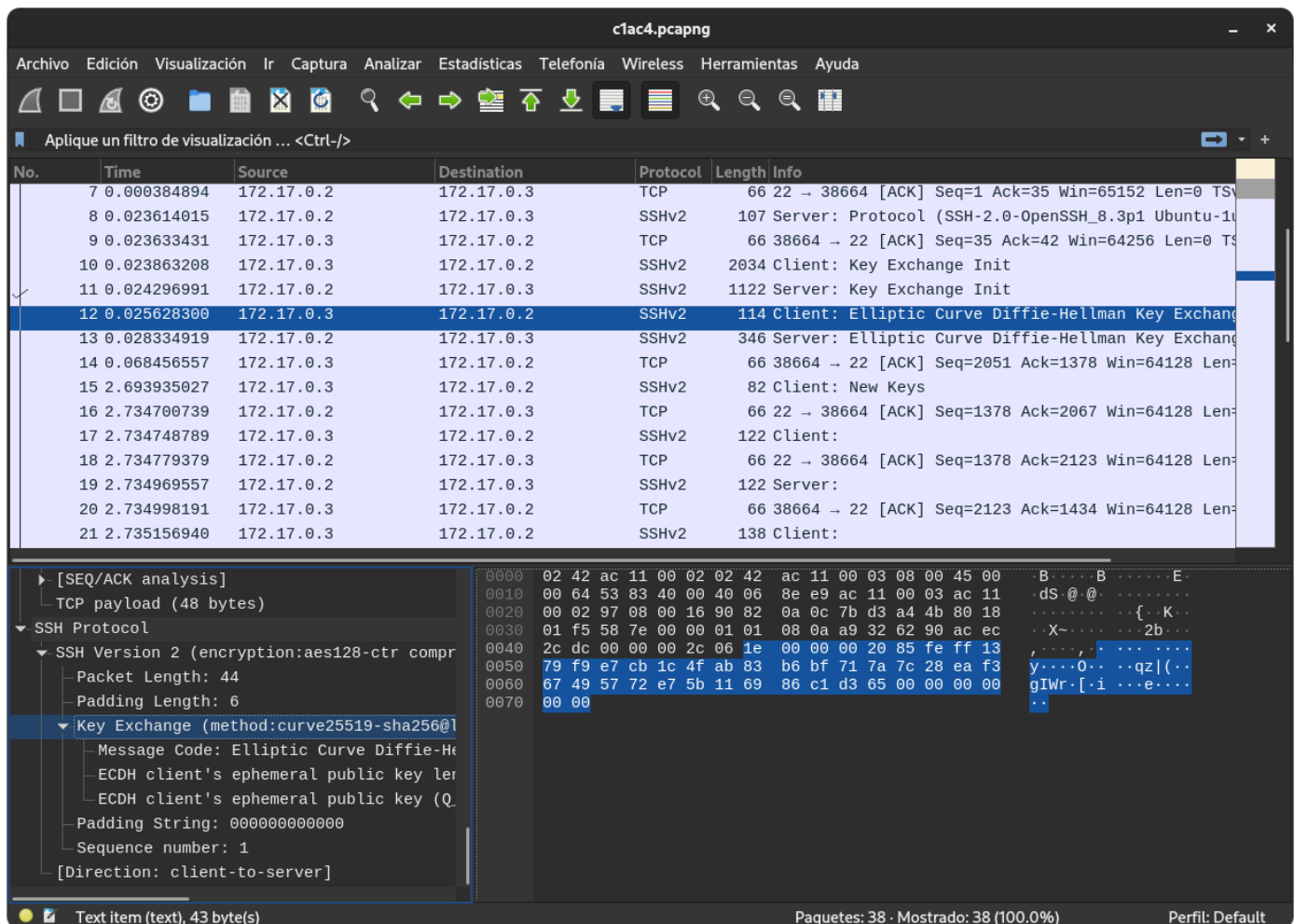
Transmission Control Protocol, Src Port: 38664, Dst Port: 22

SSH Protocol

0000 02 42 ac 11 00 02 02 42 ac 11 00 03 08 00 45 00 B B E
0010 00 56 53 7f 40 00 40 06 8e fb ac 11 00 03 ac 11 VS @ @
0020 00 02 97 08 00 16 90 82 02 3a 7b d3 a0 02 80 18 : {
0030 01 f6 58 70 00 00 01 01 08 0a a9 32 62 76 ac ec . . Xp 2bv
0040 2c c4 53 53 48 2d 32 2e 30 2d 4f 70 65 6e 53 53 , SSH-2. 0-OpenSS
0050 48 5f 36 2e 36 2e 31 70 31 20 55 62 75 6e 74 75 H_6.6.1p 1 Ubuntu
0060 2d 38 0d 0a -8

- El tráfico empieza con 3 paquetes TCP (SYN, SYN ACK, ACK), los cuáles establecen la conexión del cable.
- El primer paquete que envía el cliente es un paquete SSHv2 de tamaño 100 que utiliza la versión 6.6 de OpenSSH (paquete 6).
- El servidor responde con un paquete SSHv2 informando que está utilizando OpenSSH 8.3 (paquete 8)
- Dado que usan distintas versiones de SSH, el cliente envía entonces los algoritmos de cifrado que este soporta con un Key Exchange Init (paquete 10).

- El servidor realiza el siguiente procedimiento, informando al cliente que tipos de algoritmo soporta con un Key Exchange Init (paquete 11).
- Luego, el cliente informa los parámetros que utilizará para construir su parte del algoritmo Diffie Hellman con otro Key Exchange Init (paquete 12).



Wireshark capture of an SSH session. The packet list shows a sequence of packets including TCP ACKs, SSHv2 protocol version exchange, and key exchange. Packet 12 is selected, showing the SSHv2 Client: Elliptic Curve Diffie-Hellman Key Exchange details. The packet bytes pane shows the raw data for the key exchange.

- El servidor recibe estos parámetros y construye su parte de la llave usando sus parámetros, enviando de vuelta las nuevas llaves y el paquete encriptado. (paquete 13).
- Luego, el cliente envía también sus New Keys (paquete 15), lo cuál es utilizado para darle más complejidad al encriptamiento de los paquetes.
- Aquí finaliza la comunicación, en donde si siguiera comunicándose todos los paquetes se encontrarían ya encriptados.

3.4. Tráfico generado por C2 (detallado)

Una vez ejecutado el comando de docker run, se abre la terminal de la imagen ubuntu, en donde se ejecutará el siguiente comando y posteriormente se saldrá sin ejecutar más comandos:

```
ssh test@172.17.0.2
```

Donde 172.17.0.2 es la IP del contenedor S1 (servidor). Esto se aprecia en la siguiente imagen:

```
nico@fedora:~/Documentos/lab5/C2
docker run -p 2222:22 lab5c4
nico@fedora:~/Documentos/lab5/C2
~/Documentos/lab5/C2 12:12:15
docker run -it --rm lab5c2
root@1a724e83309e:/# 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:MFK+LlKM0mNfHWe80XxCdDVj4fENoxPVNkRH4M9AZSo.
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 6.3.6-200.fc38.x86_64 x86_64)

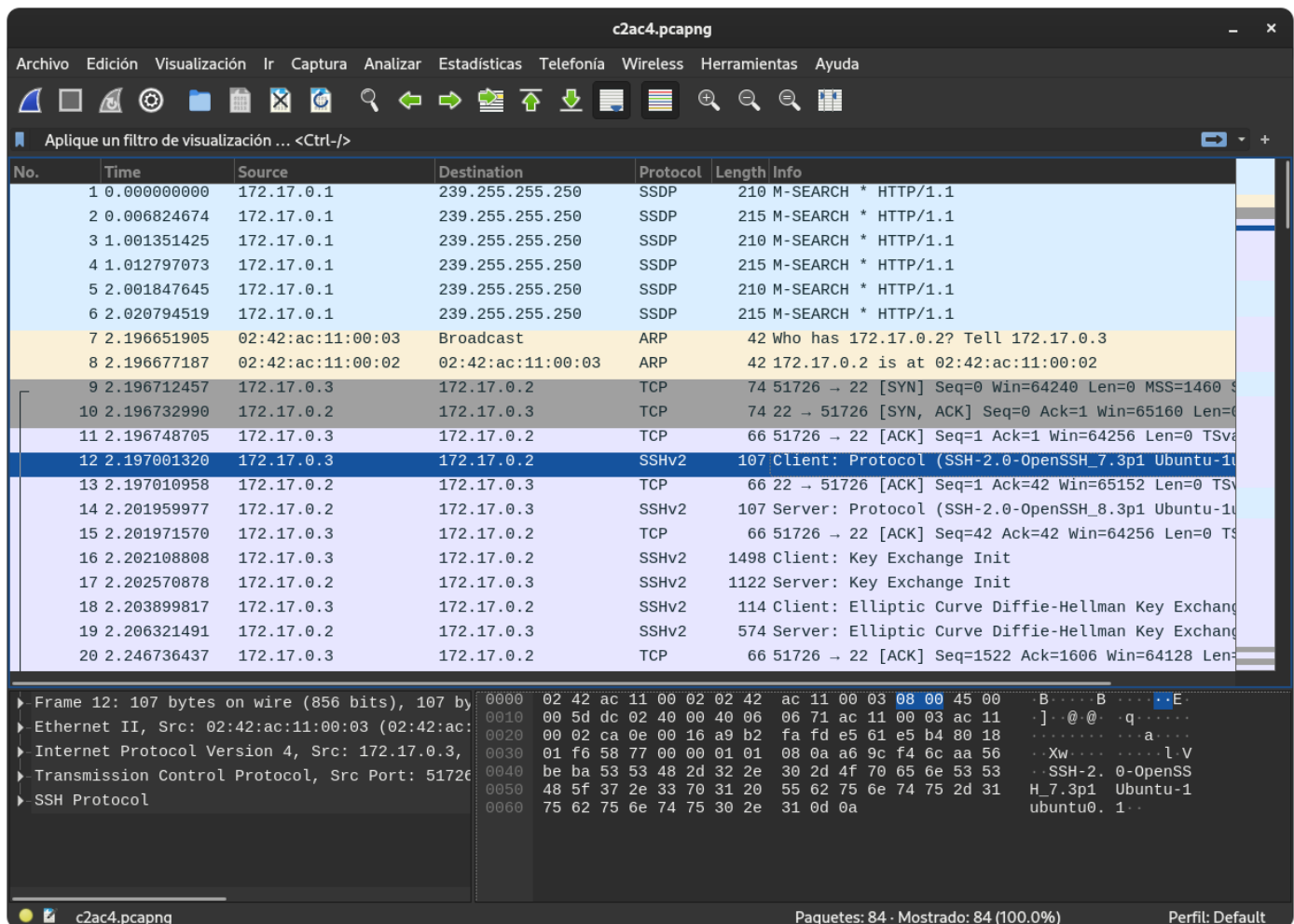
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

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: Sat Jun 10 04:12:11 2023 from 172.17.0.3
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

test@394211c257f1:~$ exit
logout
Connection to 172.17.0.2 closed.
```

Al recibir los paquetes del contenedor S1 con wireshark desde la máquina local, se tiene la siguiente captura:



El detalle de este tráfico es idéntico al tráfico generado por C1, con la diferencia que ahora el cliente utiliza una versión de OpenSSH 7.3, lo cual cambia el contenido transmitido en el Key Exchange Init del cliente. Por parte del servidor, este sigue siendo igual.

3.5. Tráfico generado por C3 (detallado)

Una vez ejecutado el comando de docker run, se abre la terminal de la imagen ubuntu, en donde se ejecutará el siguiente comando y posteriormente se saldrá sin ejecutar más comandos:

```
ssh test@172.17.0.2
```

Donde 172.17.0.2 es la IP del contenedor S1 (servidor). Esto se aprecia en la siguiente imagen:

```
nico@fedora:~/Documentos/lab5/C3
docker run -p 2222:22 lab5c4
nico@fedora:~/Documentos/lab5/C3
~/Documentos/lab5/C3 12:06:02
docker run -it --rm lab5c3
root@59999eca477c:/# ssh test@172.17.0.2
bash: ssh: command not found
root@59999eca477c:/# 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:MFK+LlKM0mNfHWe80XxCdDVj4fENoxPVNKRH4M9AZSo.
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 6.3.6-200.fc38.x86_64 x86_64)

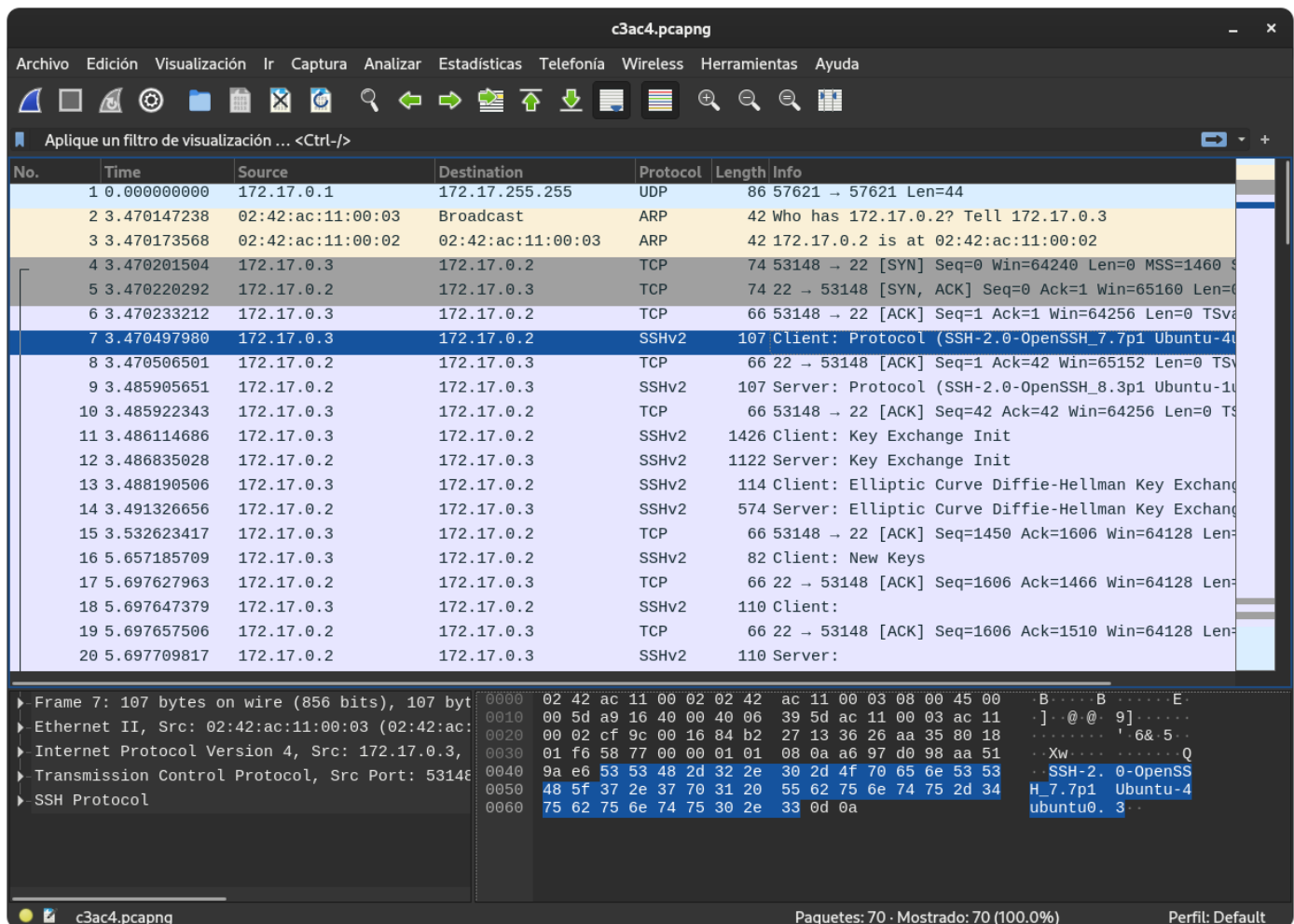
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

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: Sat Jun 10 03:58:16 2023 from 172.17.0.1
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

test@394211c257f1:~$ exit
```

Al recibir los paquetes del contenedor S1 con wireshark desde la máquina local, se tiene la siguiente captura:



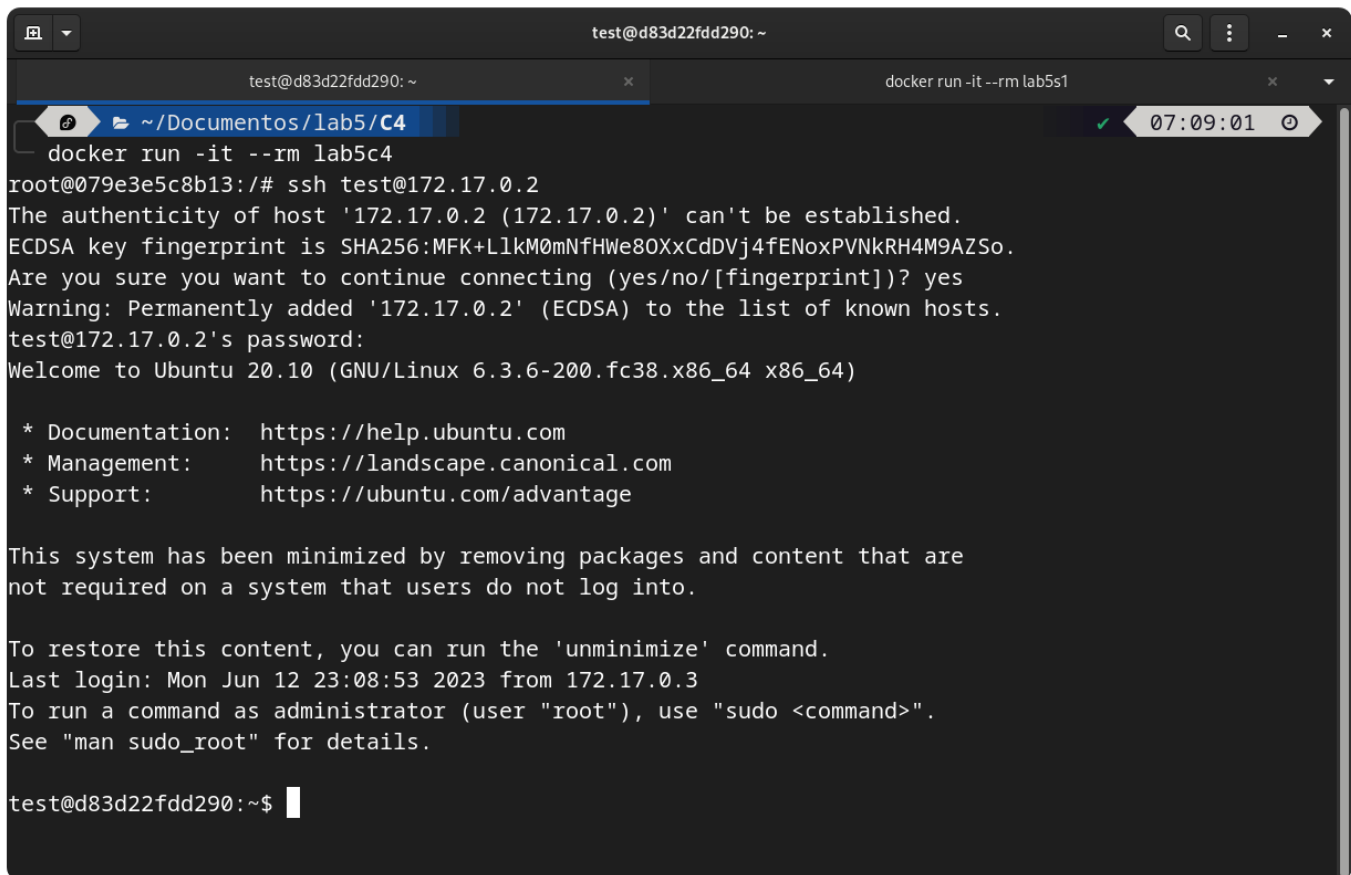
El detalle de este tráfico es idéntico al tráfico generado por C1 y C2, con la diferencia que ahora el cliente utiliza una versión de OpenSSH 7.7, lo cual cambia el contenido transmitido en el Key Exchange Init del cliente. Por parte del servidor, este sigue siendo igual.

3.6. Tráfico generado por C4 (iface io) (detallado)

Una vez ejecutado el comando de docker run, se abre la terminal de la imagen ubuntu, en donde se ejecutará el siguiente comando y posteriormente se saldrá sin ejecutar más comandos:

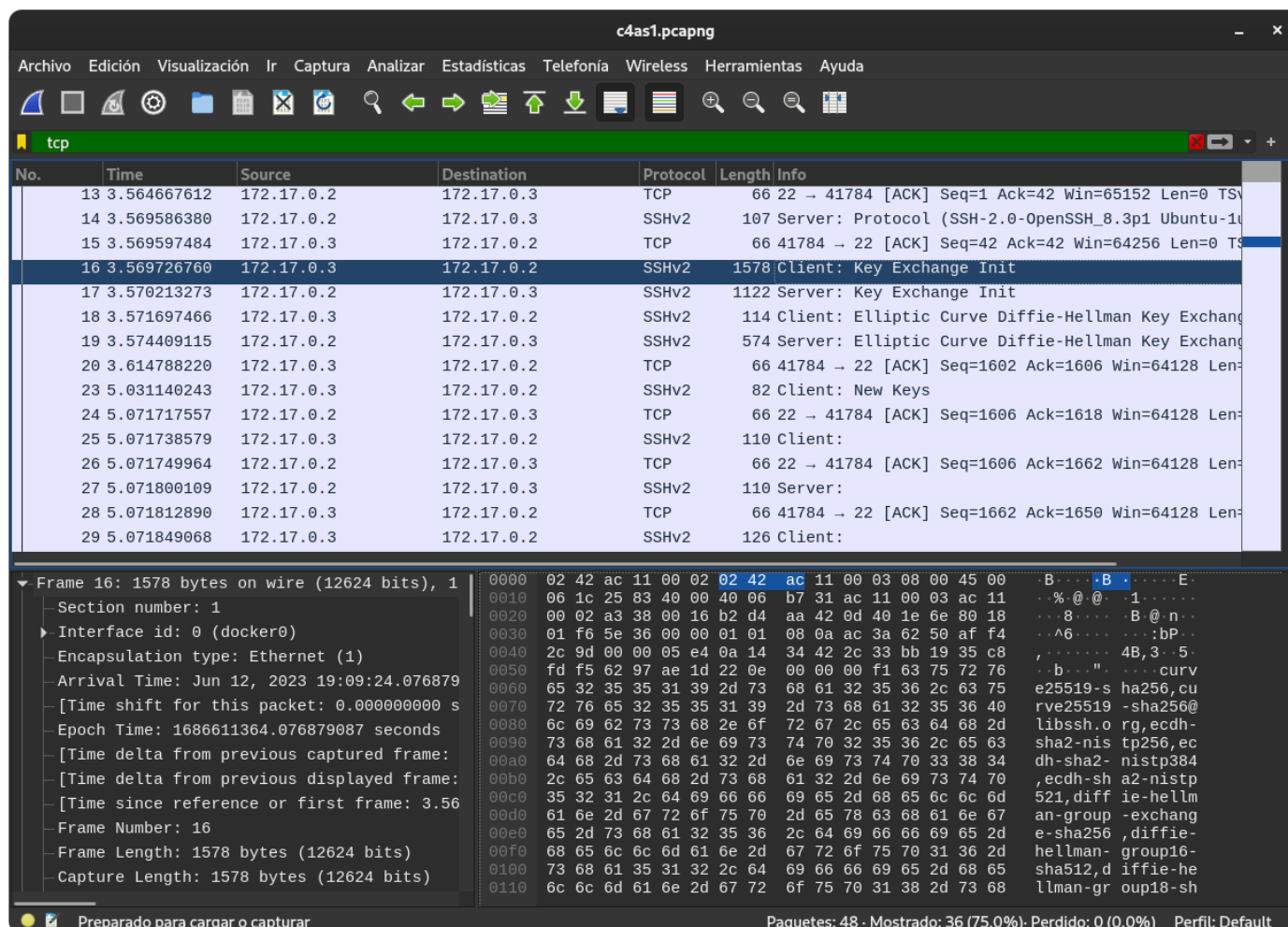
```
ssh test@172.17.0.2
```

Donde 172.17.0.2 es la IP del contenedor S1 (servidor). Esto se aprecia en la siguiente imagen:



```
test@d83d22fdd290: ~  
test@d83d22fdd290: ~  
~/Documentos/lab5/C4  
docker run -it --rm lab5c4  
root@079e3e5c8b13:/# 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:MFK+LlKM0mNfHWe80XxCdDVj4fENoxPVNkRH4M9AZSo.  
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:  
Welcome to Ubuntu 20.10 (GNU/Linux 6.3.6-200.fc38.x86_64 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: Mon Jun 12 23:08:53 2023 from 172.17.0.3  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
test@d83d22fdd290:~$
```

Al recibir los paquetes del contenedor S1 con wireshark desde la máquina local, se tiene la siguiente captura:



Packet 16: 1578 bytes on wire (12624 bits), 1

- Section number: 1
- Interface id: 0 (docker0)
- Encapsulation type: Ethernet (1)
- Arrival Time: Jun 12, 2023 19:09:24.076879
- [Time shift for this packet: 0.000000000 s]
- Epoch Time: 1686611364.076879087 seconds
- [Time delta from previous captured frame: 0.000000000 s]
- [Time delta from previous displayed frame: 0.000000000 s]
- [Time since reference or first frame: 3.56 s]
- Frame Number: 16
- Frame Length: 1578 bytes (12624 bits)
- Capture Length: 1578 bytes (12624 bits)

SSHv2 Client: Key Exchange Init

0000 02 42 ac 11 00 02 02 42 ac 11 00 03 08 00 45 00 B...B...E...
 0010 06 1c 25 83 40 00 40 06 b7 31 ac 11 00 03 ac 11 ..%...@...1...
 0020 00 02 a3 38 00 16 b2 d4 aa 42 0d 40 1e 6e 80 18 ...8...B...n...
 0030 01 f6 5e 36 00 00 01 01 08 0a ac 3a 62 50 af f4 ...^6...bP...
 0040 2c 9d 00 00 05 e4 0a 14 34 42 2c 33 bb 19 35 c8 ...4B,3...5...
 0050 fd f5 62 97 ae 1d 22 0e 00 00 00 f1 63 75 72 76 ...b..."....curv...
 0060 65 32 35 35 31 39 2d 73 68 61 32 35 36 2c 63 75 e25519-s ha256,cu...
 0070 72 76 65 32 35 35 31 39 2d 73 68 61 32 35 36 40 rve25519 -sha256@...
 0080 6c 69 62 73 73 68 2e 6f 72 67 2c 65 63 64 68 2d libssh.o rg,ecdh-...
 0090 73 68 61 32 2d 6e 69 73 74 70 32 35 36 2c 65 63 sha2-nis tp256,ec...
 00a0 64 68 2d 73 68 61 32 2d 6e 69 73 74 70 33 38 34 dh-sha2- nistp384...
 00b0 2c 65 63 64 68 2d 73 68 61 32 2d 6e 69 73 74 70 ,ecdh-sh a2-nistp...
 00c0 35 32 31 2c 64 69 66 66 69 65 2d 68 65 6c 6c 6d 521,diff ie-hellm...
 00d0 61 6e 2d 67 72 6f 75 70 2d 65 78 63 68 61 6e 67 an-group -exchang...
 00e0 65 2d 73 68 61 32 35 36 2c 64 69 66 66 69 65 2d e-sha256 ,diffie-...
 00f0 68 65 6c 6c 6d 61 6e 2d 67 72 6f 75 70 31 36 2d hellman- group16-...
 0100 73 68 61 35 31 32 2c 64 69 66 66 69 65 2d 68 65 sha512,d iffie-he...
 0110 6c 6c 6d 61 6e 2d 67 72 6f 75 70 31 38 2d 73 68 llman-gr oup18-sh...

El detalle de este tráfico es idéntico al tráfico generado por C1, C2 y C3, con la diferencia que ahora el cliente utiliza una versión de OpenSSH 8.3, lo cual cambia el contenido transmitido en el Key Exchange Init del cliente. Por parte del servidor, este sigue siendo igual.

3.7. Diferencia entre C1 y C2

La diferencia principal entre el tráfico de C1 y C2 es que C1 incluye un paquete de Key Exchange mucho más grande que C2, siendo el primero de largo 2034 y el segundo de 1498. Esto significa que al momento de intercambiar información respecto a los algoritmos de cifrado que pueden utilizar para encriptar el mensaje, se tuvo que enviar mayor información para poder encontrar la intersección de estas.

3.8. Diferencia entre C2 y C3

La diferencia principal entre el tráfico de C2 y C3 es que C2 incluye un paquete de Key Exchange un poco más grande que C3, siendo el primero de largo 1498 y el segundo de 1426. Esto significa que al momento de intercambiar información respecto a los algoritmos de cifrado que pueden utilizar para encriptar el mensaje, se tuvo que enviar mayor información para poder encontrar la intersección de estas.

3.9. Diferencia entre C3 y C4

La diferencia principal entre el tráfico de C3 y C4 es que C3 incluye un paquete de Key Exchange más pequeño que C4, siendo el primero de largo 1426 y el segundo de 1578. Esto significa que al momento de intercambiar información respecto a los algoritmos de cifrado que pueden utilizar para encriptar el mensaje, se tuvo que enviar mayor información para poder encontrar la intersección de estas dado que estos nuevos sistemas soportan también más algoritmos de cifrado.

4. Desarrollo (Parte 2)

4.1. Identificación del cliente SSH

Dado que se utiliza un cliente de SSH que envía un paquete de largo 1578 en su Key Exchange, se puede asumir que este cliente utiliza la versión 8.3 perteneciente a ubuntu 20.10. Para lograr que este paquete sea enviado con un signo de interrogación "?" En la parte de versión, habría que meterse dentro del código fuente del cliente de forma que se altere la parte en dónde se muestra la versión del cliente y así lograr un paquete cuya versión es "?".

4.2. Replicación del tráfico (paso por paso)

Para replicar este tráfico, se creó un nuevo dockerfile denominado "Incognito". Este dockerfile lo que hace es descargar OpenSSH Portable en la versión 8.3 desde su código source, de forma que se siga teniendo un Key Exchange de 1578. Posteriormente, se buildeará utilizando MAKE, pero antes de eso se modificará la línea 3 del código "version.h", el cuál incluye la versión del software que estamos utilizando, quedando finalmente el siguiente archivo version.h:

```
/* $OpenBSD: version.h,v 1.97 2023/03/15 21:19:57 djm Exp $ */

#define SSH_VERSION      "OpenSSH_?"

#define SSH_PORTABLE     "p1"
#define SSH_RELEASE      SSH_VERSION SSH_PORTABLE
```

El dockerfile que utilizaremos será el siguiente:

```
FROM ubuntu:20.10

COPY sources.list /etc/apt/sources.list

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get install -y build-essential wget zlib1g-dev libssl-dev

# Instalar OpenSSH Portable 8.3
RUN wget https://cdn.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-8.3p1.tar.gz && \
    tar zxvf openssh-8.3p1.tar.gz

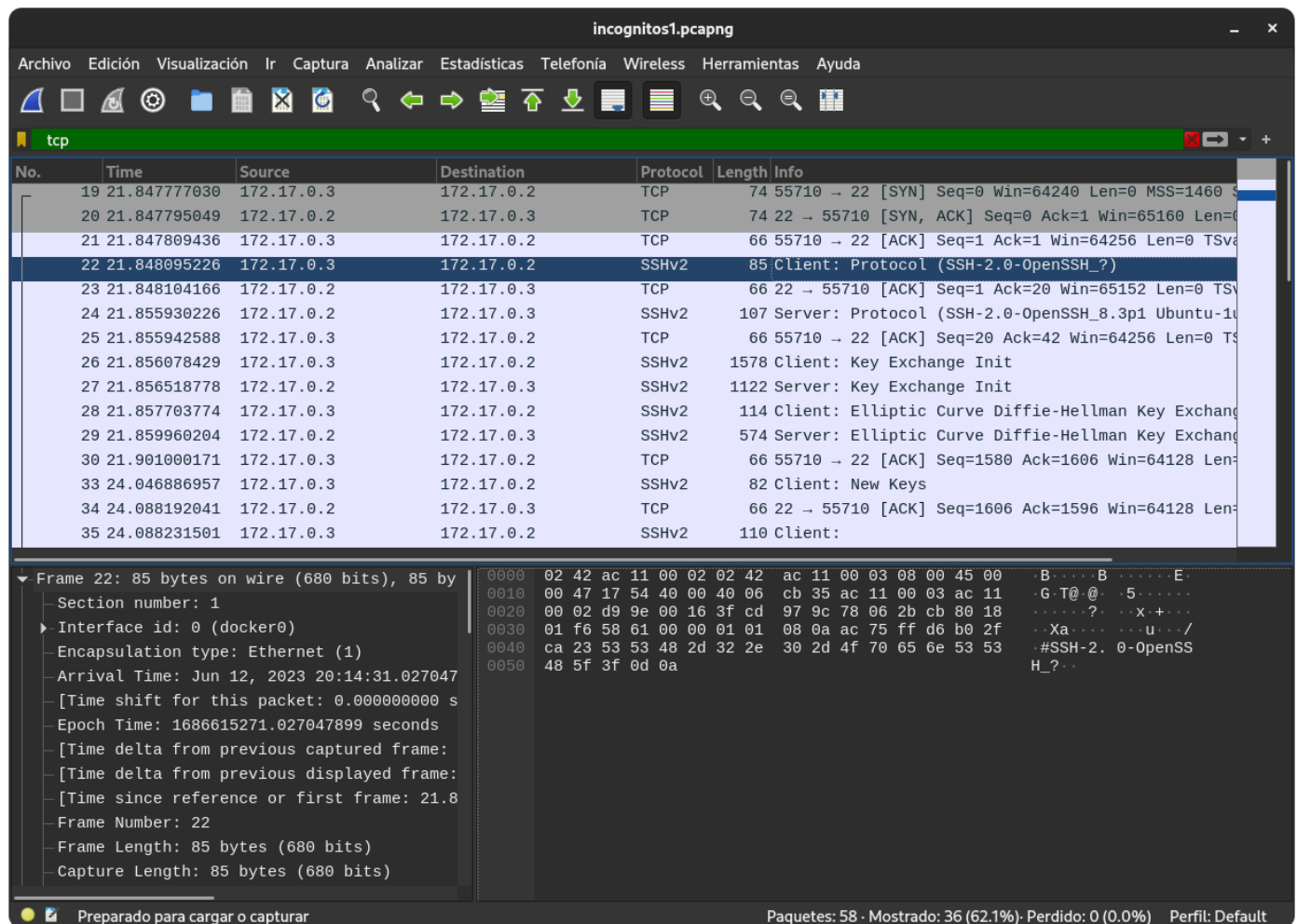
# Modificar la version a ?
RUN sed -i '3s/8.3/?/' /openssh-8.3p1/version.h

WORKDIR /openssh-8.3p1
```

```
# Buildear desde source
RUN ./configure && \
    make && \
    make install

CMD ["bash"]
```

De esta forma, al hacer el procedimiento normal de buildear el dockerfile, ejecutarlo y posteriormente conectarse con ssh al contenedor S1, se tiene el siguiente resultado:



incognitos1.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

tcp

No.	Time	Source	Destination	Protocol	Length	Info
19	21.847777030	172.17.0.3	172.17.0.2	TCP	74	55710 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
20	21.847795049	172.17.0.2	172.17.0.3	TCP	74	22 → 55710 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
21	21.847809436	172.17.0.3	172.17.0.2	TCP	66	55710 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSv
22	21.848095226	172.17.0.3	172.17.0.2	SSHv2	85	Client: Protocol (SSH-2.0-OpenSSH_?)
23	21.848104166	172.17.0.2	172.17.0.3	TCP	66	22 → 55710 [ACK] Seq=1 Ack=20 Win=65152 Len=0 TSv
24	21.855930226	172.17.0.2	172.17.0.3	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1u
25	21.855942588	172.17.0.3	172.17.0.2	TCP	66	55710 → 22 [ACK] Seq=20 Ack=42 Win=64256 Len=0 TS
26	21.856078429	172.17.0.3	172.17.0.2	SSHv2	1578	Client: Key Exchange Init
27	21.856518778	172.17.0.2	172.17.0.3	SSHv2	1122	Server: Key Exchange Init
28	21.857703774	172.17.0.3	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchang
29	21.859960204	172.17.0.2	172.17.0.3	SSHv2	574	Server: Elliptic Curve Diffie-Hellman Key Exchang
30	21.901000171	172.17.0.3	172.17.0.2	TCP	66	55710 → 22 [ACK] Seq=1580 Ack=1606 Win=64128 Len=
33	24.046886957	172.17.0.3	172.17.0.2	SSHv2	82	Client: New Keys
34	24.088192041	172.17.0.2	172.17.0.3	TCP	66	22 → 55710 [ACK] Seq=1606 Ack=1596 Win=64128 Len=
35	24.088231501	172.17.0.3	172.17.0.2	SSHv2	110	Client:

Frame 22: 85 bytes on wire (680 bits), 85 by
 Section number: 1
 Interface id: 0 (docker0)
 Encapsulation type: Ethernet (1)
 Arrival Time: Jun 12, 2023 20:14:31.027047
 [Time shift for this packet: 0.000000000 s
 Epoch Time: 1686615271.027047899 seconds
 [Time delta from previous captured frame:
 [Time delta from previous displayed frame:
 [Time since reference or first frame: 21.8
 Frame Number: 22
 Frame Length: 85 bytes (680 bits)
 Capture Length: 85 bytes (680 bits)

Preparado para cargar o capturar Paquetes: 58 · Mostrado: 36 (62.1%) · Perdido: 0 (0.0%) · Perfil: Default

Donde se evidencia que se consiguió mandar un paquete SSH el cuál tiene versión “?”.

5. Desarrollo (Parte 3)

5.1. Replicación del tráfico (paso por paso)

Intenté con todas las herramientas por cielo mar y tierra y no pude :(

6. Conclusión

En conclusión, se logró la mayoría de los objetivos de este laboratorio, el cuál era diferenciar los paquetes de tráfico generados por el protocolo SSH según la versión que se esté utilizando, así también poder identificar los valores importantes de estos paquetes para reconocer que un cliente el cuál se conecta a nuestro servidor es un cliente deseado y no un intruso. La única experiencia que no se logró fue la última, lamentablemente por falta de tiempo e información al respecto. En general muy agotadora experiencia, ojalá no se repita.