

# Informe Laboratorio 5

## **Sección 2**

Alumno Alejandro Arratia  
e-mail: alejandro.arratia@mail.udp.cl

Julio de 2024

# Índice

<b>Descripción de actividades</b>	<b>3</b>
<b>1. Desarrollo (Parte 1)</b>	<b>5</b>
1.1. Códigos de cada Dockerfile . . . . .	5
1.1.1. C1 . . . . .	5
1.1.2. C2 . . . . .	6
1.1.3. C3 . . . . .	7
1.1.4. C4/S1 . . . . .	7
1.2. Creación de las credenciales para S1 . . . . .	8
1.3. Tráfico generado por C1, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado) . . . . .	8
1.4. Tráfico generado por C2, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado) . . . . .	10
1.5. Tráfico generado por C3, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado) . . . . .	11
1.6. Tráfico generado por C4 (iface lo), detallando tamaño paquetes del flujo y el HASSH respectivo (detallado) . . . . .	13
1.7. Compara la versión de HASSH obtenida con la base de datos para validar si el cliente corresponde al mismo . . . . .	14
1.8. Tipo de información contenida en cada uno de los paquetes generados en texto plano . . . . .	14
1.8.1. C1 . . . . .	14
1.8.2. C2 . . . . .	15
1.8.3. C3 . . . . .	16
1.8.4. C4/S1 . . . . .	16
1.9. Diferencia entre C1 y C2 . . . . .	17
1.10. Diferencia entre C2 y C3 . . . . .	17
1.11. Diferencia entre C3 y C4 . . . . .	17
<b>2. Desarrollo (Parte 2)</b>	<b>18</b>
2.1. Identificación del cliente SSH con versión “?” . . . . .	18
2.2. Replicación de tráfico al servidor (paso por paso) . . . . .	18
<b>3. Desarrollo (Parte 3)</b>	<b>19</b>
3.1. Replicación del KEI con tamaño menor a 300 bytes (paso por paso) . . . . .	19

## 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 o Podman, donde cada uno tendrá el siguiente SO: Ubuntu 16.10, Ubuntu 18.10, Ubuntu 20.10 y Ubuntu 22.10 a los cuales se llamarán C1, C2, C3 y C4 respectivamente.  
El equipo con Ubuntu 22.10 también será utilizado como S1.
- Para cada uno de ellos, deberá instalar el cliente openSSH disponible en los repositorios de apt, y para el equipo S1 deberá también instalar el servidor openSSH.
- En S1 deberá crear el usuario “**prueba**” con contraseña “**prueba**”, para acceder a él desde los clientes por el protocolo SSH.
- 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, deberá capturar el tráfico generado por cada conexión con el server. A partir de cada handshake, deberá analizar el patrón de tráfico generado por cada cliente y adicionalmente obtener el HASSH que lo identifique. De esta forma podrá obtener una huella digital para cada cliente a partir de su tráfico. Cada HASSH deberá compararlo con la base de datos HASSH disponible en el módulo de TLS, e identificar si el hash obtenido corresponde a la misma versión de su cliente.

Indique el tamaño de los paquetes del flujo generados por el cliente y el contenido asociado a cada uno de ellos. Indique qué información distinta contiene el escenario siguiente (diff incremental). El objetivo de este paso 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

Replique este tráfico generado en la imagen. Debe generar el tráfico con la misma versión resaltada en azul. Recuerde que toda la información generada es parte del sw, por lo tanto usted puede modificar toda la información.

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.

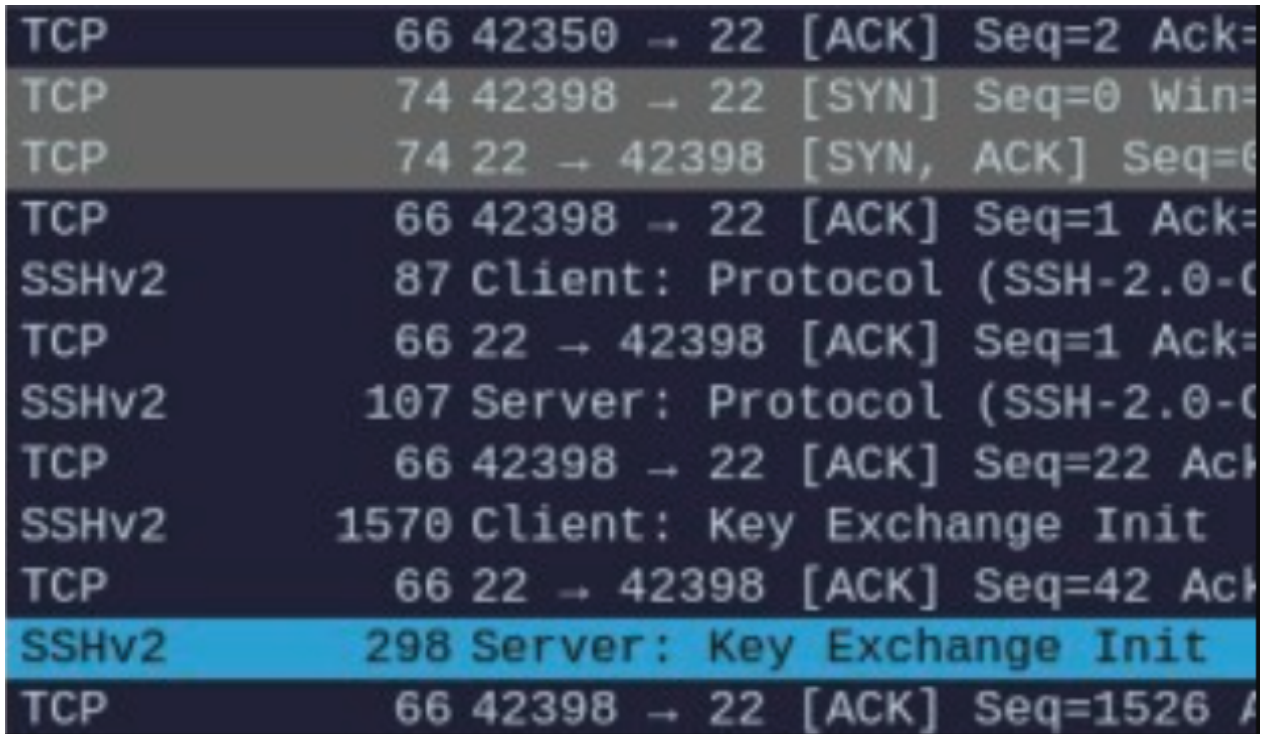


Figura 2: Captura del Key Exchange

## 1. Desarrollo (Parte 1)

### 1.1. Códigos de cada Dockerfile

A cada contenedor creado se le agrega una `sources.list` dado que las versiones utilizadas de Ubuntu se encuentran deprecadas y no existen en los repositorios oficiales.

#### 1.1.1. C1

Se comienza con el contenedor C1, que posee Ubuntu 16.10 y actuará como el primer cliente, se aprovecha de actualizarlo e instalarle `openssh` automáticamente.

```

1 FROM ubuntu:16.10
2
3 RUN sed -i 's|archive.ubuntu.com|old-releases.ubuntu.com|g' /etc/apt/
  sources.list
4 RUN sed -i 's|security.ubuntu.com|old-releases.ubuntu.com|g' /etc/apt/
  sources.list
5 RUN sed -i '/^# deb-src/s/^# //' /etc/apt/sources.list
6
7 RUN apt-get update
8 RUN apt-get install openssh-server openssh-client dpkg-dev nano autoconf
  automake libtool zlib1g-dev libssl-dev --fix-missing -y
9 RUN apt-get source openssh-client

```

```
10
11
12 RUN mkdir /var/run/sshd
13
14 RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc
    /ssh/sshd_config
15
16 EXPOSE 22
17
18 CMD ["/usr/sbin/sshd", "-D"]
```

Código 1: Dockerfile C1

### 1.1.2. C2

Se sigue con el contenedor C2, que posee Ubuntu 18.10 y actuará como el segundo cliente, también se aprovecha de actualizarlo e instalarle openssh automáticamente.

```
1 FROM ubuntu:18.10
2
3 RUN sed -i 's|archive.ubuntu.com|old-releases.ubuntu.com|g' /etc/apt/
    sources.list
4 RUN sed -i 's|security.ubuntu.com|old-releases.ubuntu.com|g' /etc/apt/
    sources.list
5 RUN sed -i '/^# deb-src/s/^# //' /etc/apt/sources.list
6
7 RUN apt-get update
8 RUN apt-get install openssh-server openssh-client dpkg-dev nano autoconf
    automake libtool zlib1g-dev libssl-dev --fix-missing -y
9 RUN apt-get source openssh-client
10
11
12 RUN mkdir /var/run/sshd
13
14 RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc
    /ssh/sshd_config
15
16 EXPOSE 22
17
18 CMD ["/usr/sbin/sshd", "-D"]
```

Código 2: Dockerfile C2

### 1.1.3. C3

Se continúa con el contenedor C3, que posee Ubuntu 20.10 y actuará como el tercer cliente, también se aprovecha de actualizarlo e instalarle openssh automáticamente.

```

1 FROM ubuntu:20.10
2
3 RUN sed -i 's|archive.ubuntu.com|old-releases.ubuntu.com|g' /etc/apt/
  sources.list
4 RUN sed -i 's|security.ubuntu.com|old-releases.ubuntu.com|g' /etc/apt/
  sources.list
5 RUN sed -i '/^# deb-src/s/^# //' /etc/apt/sources.list
6
7 RUN apt-get update
8 RUN apt-get install openssh-server openssh-client dpkg-dev nano autoconf
  automake libtool zlib1g-dev libssl-dev --fix-missing -y
9 RUN apt-get source openssh-client
10
11
12 RUN mkdir /var/run/sshd
13
14 RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc
  /ssh/sshd_config
15
16 EXPOSE 22
17
18 CMD ["/usr/sbin/sshd", "-D"]

```

Código 3: Dockerfile C3

### 1.1.4. C4/S1

Finalmente el último contenedor creado, corresponde al C4 con una versión de Ubuntu 22.10, que a su vez posee el servidor S1. Además de hacer los pasos similares a los demás contenedores, se aprovecha de crear el usuario prueba:prueba que será requerido por todos los clientes al conectarse al servidor.

```

1 FROM ubuntu:22.10
2
3 RUN sed -i 's|archive.ubuntu.com|old-releases.ubuntu.com|g' /etc/apt/
  sources.list
4 RUN sed -i 's|security.ubuntu.com|old-releases.ubuntu.com|g' /etc/apt/
  sources.list
5 RUN sed -i '/^# deb-src/s/^# //' /etc/apt/sources.list
6
7 RUN apt-get update
8 RUN apt-get install -y openssh-server openssh-client zlib1g-dev libssl-dev
  autoconf automake libtool dpkg-dev --fix-missing
9
10 # Ensure sshd_config exists and has proper permissions
11 RUN mkdir -p /var/run/sshd

```

```
12  
13 ENV USER_NAME=prueba  
14 ENV USER_HOME=/home/$USER_NAME  
15  
16 RUN useradd -m -d $USER_HOME $USER_NAME  
17 RUN echo 'prueba:prueba' | chpasswd  
18  
19 RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc  
    /ssh/sshd_config  
20  
21 EXPOSE 22  
22  
23 CMD ["/usr/sbin/sshd", "-D"]
```

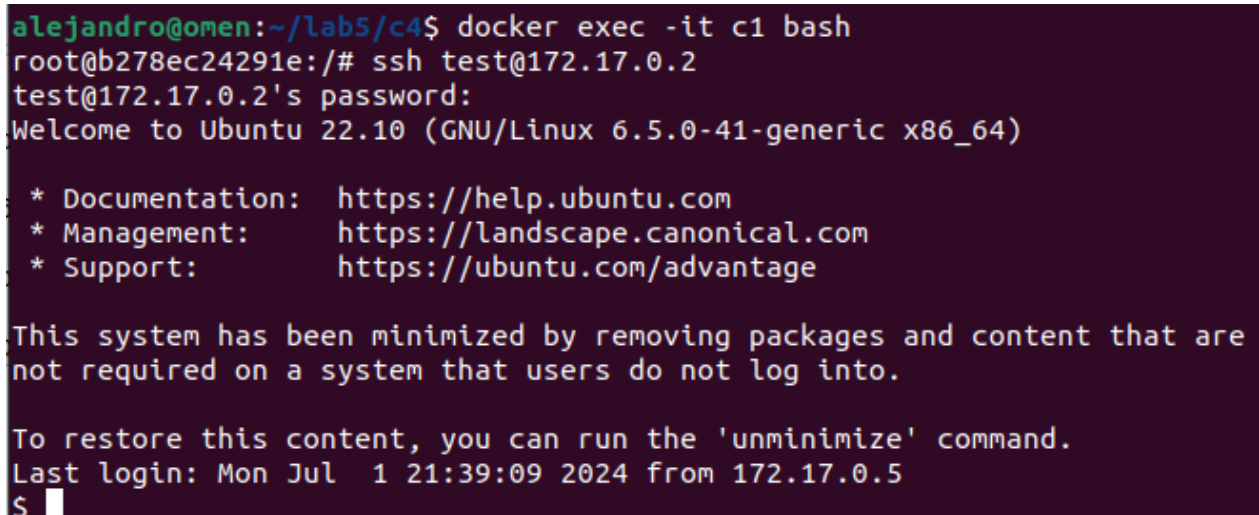
Código 4: Dockerfile C4/S1

## 1.2. Creación de las credenciales para S1

En el paso anterior (Código 4), dentro del Dockerfile correspondiente a C4/S1, se creó el usuario prueba con contraseña prueba, que poseerá los permisos suficientes del sistema operativo para que los clientes puedan conectarse al servidor ssh.

## 1.3. Tráfico generado por C1, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)

Nos conectamos al servidor S1 con dirección IP 172.17.0.2 como se muestra en la Figura 3 utilizando las credenciales creadas previamente.



```
alejandro@omen:~/lab5/c4$ docker exec -it c1 bash  
root@b278ec24291e:/# ssh test@172.17.0.2  
test@172.17.0.2's password:  
Welcome to Ubuntu 22.10 (GNU/Linux 6.5.0-41-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: Mon Jul  1 21:39:09 2024 from 172.17.0.5  
$
```

Figura 3: Conexión C1-C4



### 1.3 Tráfico generado por C1, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)

#### 1 DESARROLLO (PARTE 1)

En la Figura 4 podemos observar el tráfico que se genera al conectar el cliente 1 con el servidor. Se puede observar el handshake de ssh en donde comparten las versiones de ssh que soportan. En el caso del cliente vemos en el paquete correspondiente que posee un tamaño de 74 bytes con una versión de SSH-2.0 OpenSSH 7.3p1.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.5	172.17.0.2	TCP	74	45214 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1495839863 TSecr=
2	0.000017571	172.17.0.2	172.17.0.5	TCP	74	22 → 45214 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=244269
3	0.000026578	172.17.0.5	172.17.0.2	TCP	66	45214 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1495839863 TSecr=2442697795
4	0.000132401	172.17.0.5	172.17.0.2	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.3p1 Ubuntu-1ubuntu0.1)
5	0.000136643	172.17.0.2	172.17.0.5	TCP	66	22 → 45214 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=2442697795 TSecr=1495839863
6	0.000590022	172.17.0.2	172.17.0.5	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
7	0.000596049	172.17.0.5	172.17.0.2	TCP	66	45214 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=1495839864 TSecr=2442697796
8	0.000808093	172.17.0.5	172.17.0.2	SSHv2	1498	Client: Key Exchange Init
9	0.001298092	172.17.0.2	172.17.0.5	SSHv2	1146	Server: Key Exchange Init
10	0.005257692	172.17.0.5	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.012227308	172.17.0.2	172.17.0.5	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packe

Algorithms
Cookie: 8e0aade35dbd596e797165e5e61f7218
kex\_algorithms length: 286
kex\_algorithms string [truncated]: curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-gr
server\_host\_key\_algorithms length: 290
server\_host\_key\_algorithms string [truncated]: ecdsa-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@openssh.com,ecdsa-sha
encryption\_algorithms\_client\_to\_server length: 150
encryption\_algorithms\_client\_to\_server string: chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes25
encryption\_algorithms\_server\_to\_client length: 150
encryption\_algorithms\_server\_to\_client string: chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes25
mac\_algorithms\_client\_to\_server length: 213
mac\_algorithms\_client\_to\_server string [truncated]: umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sh
mac\_algorithms\_server\_to\_client length: 213
mac\_algorithms\_server\_to\_client string [truncated]: umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sh
compression\_algorithms\_client\_to\_server string: none,zlib@openssh.com,zlib
compression\_algorithms\_server\_to\_client length: 26
compression\_algorithms\_server\_to\_client string: none,zlib@openssh.com,zlib
languages\_client\_to\_server length: 0
languages\_client\_to\_server string:
languages\_server\_to\_client length: 0
languages\_server\_to\_client string:
First KEX Packet Follows: 0
Reserved: 00000000
[hashAlgorithms [truncated]: curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-e
[hashsh: 0e4584cb9f2dd077dbf8ba0df8112d8e]

Figura 4: Tráfico C1 - S1

Luego vemos que el cliente hace el intercambio de Key Init con un paquete de tamaño 1498 bytes, y si observamos el datagrama del paquete se puede apreciar los algoritmos para generar la llave, para encriptarla, los mensajes de autenticación y los algoritmos de compresión. Finalmente observamos que su HASSH generado es **0e4584cb9f2dd077dbf8ba0df8112d8e**.

Finalmente el Cliente inicia el proceso de las curvas elipiticas de Diffie Hellman y genera una nueva llave, estos paquetes pesan 114 y 82 bytes respectivamente.

## 1.4. Tráfico generado por C2, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)

Nos conectamos al servidor S1 con dirección IP 172.17.0.2 como se muestra en la Figura 5, esta vez desde el contenedor C2, utilizando las credenciales creadas previamente. En la

```
alejandro@omen:~/lab5/c4$ docker exec -it c2 bash
root@e99c0387eecf:/# ssssh test@172.17.0.2
bash: ssssh: command not found
root@e99c0387eecf:/# 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:ha7nN6lG6JNAqvJqWHzDorSY1r4lMs8C80ZlzsASqs.
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 22.10 (GNU/Linux 6.5.0-41-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: Mon Jul  1 21:54:08 2024 from 172.17.0.5
```

Figura 5: Conexión C2-C4

Figura 6 podemos observar el tráfico que se genera al conectar el cliente 2 con el servidor. Se puede observar el handshake de ssh en donde comparten las versiones de ssh que soportan. En el caso del cliente vemos en el paquete correspondiente que posee un tamaño de 107 bytes con una versión de SSH-2.0 OpenSSH 7.7p1.

## 1.5 Tráfico generado por C3, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)

### 1 DESARROLLO (PARTE 1)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.4	172.17.0.2	TCP	74	53674 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3411032735 TSecr=0 WS
2	0.000045473	172.17.0.2	172.17.0.4	TCP	74	22 → 53674 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3794835072
3	0.000074896	172.17.0.4	172.17.0.2	TCP	66	53674 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3411032735 TSecr=3794835072
4	0.000451501	172.17.0.4	172.17.0.2	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.7p1 Ubuntu-4ubuntu0.3)
5	0.000467122	172.17.0.2	172.17.0.4	TCP	66	22 → 53674 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=3794835073 TSecr=3411032736
6	0.001857011	172.17.0.2	172.17.0.4	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
7	0.001884624	172.17.0.4	172.17.0.2	TCP	66	53674 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=3411032737 TSecr=3794835074
8	0.002619511	172.17.0.4	172.17.0.2	SSHv2	1426	Client: Key Exchange Init
9	0.004358684	172.17.0.2	172.17.0.4	SSHv2	1146	Server: Key Exchange Init
10	0.008976315	172.17.0.2	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.017884522	172.17.0.2	172.17.0.4	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (1
12	0.022775838	172.17.0.4	172.17.0.2	SSHv2	82	Client: New Keys

Algorithms  
Cookie: 2903da0131e6c8f42b777a058ddc18cc  
kex\_algorithms length: 304  
kex\_algorithms string [truncated]: curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,dif  
server\_host\_key\_algorithms length: 290  
server\_host\_key\_algorithms string [truncated]: ecdsa-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@openssh.com,ecdsa-sha2-ni  
encryption\_algorithms\_client\_to\_server length: 108  
encryption\_algorithms\_client\_to\_server string: chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gc  
encryption\_algorithms\_server\_to\_client length: 108  
encryption\_algorithms\_server\_to\_client string: chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gc  
mac\_algorithms\_client\_to\_server length: 213  
mac\_algorithms\_client\_to\_server string [truncated]: umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-5  
mac\_algorithms\_server\_to\_client length: 213  
mac\_algorithms\_server\_to\_client string [truncated]: umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-5  
compression\_algorithms\_client\_to\_server string: none,zlib@openssh.com,zlib  
compression\_algorithms\_server\_to\_client length: 26  
compression\_algorithms\_server\_to\_client string: none,zlib@openssh.com,zlib  
languages\_client\_to\_server length: 0  
languages\_client\_to\_server string:  
languages\_server\_to\_client length: 0  
languages\_server\_to\_client string:  
First KEX Packet Follows: 0  
Reserved: 00000000  
[hasshAlgorithms [truncated]: curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-h

Figura 6: Tráfico C2 - S1

Luego vemos que el cliente hace el intercambio de Key Init con un paquete de tamaño 1426 bytes, y si observamos el datagrama del paquete se puede apreciar los algoritmos para generar la llave, para encriptarla, los mensajes de autenticación y los algoritmos de compresión. Finalmente observamos que su HASSH generado es **06046964c022c6407d15a27b12a6a4fb**.

Finalmente el Cliente inicia el proceso de las curvas elipiticas de Diffie Hellman y genera una nueva llave, estos paquetes pesan 114 y 82 bytes respectivamente.

## 1.5. Tráfico generado por C3, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)

Nos conectamos al servidor S1 con dirección IP 172.17.0.2 como se muestra en la Figura 7, esta vez desde el contenedor C3, utilizando las credenciales creadas previamente.

## 1.5 Tráfico generado por C3, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)

### 1 DESARROLLO (PARTE 1)

```

alejandro@omen:~/lab5/c4$ docker exec -it c3 bash
root@e08e0d8850fb:/# 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:ha7nN6lG6JNAqvJqWHzDorSY1r4lMs8C80ZlZsuASqs.
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 22.10 (GNU/Linux 6.5.0-41-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: Mon Jul  1 23:20:08 2024 from 172.17.0.4

```

Figura 7: Conexión C3-S1

En la Figura 8 podemos observar el tráfico que se genera al conectar el cliente 3 con el servidor. Se puede observar el handshake de ssh en donde comparten las versiones de ssh que soportan. En el caso del cliente vemos en el paquete correspondiente que posee un tamaño de 107 bytes con una versión de SSH-2.0 OpenSSH 8.3p1.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.000051847	172.17.0.3	172.17.0.2	TCP	74	38662 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=985851249 TSecr=0 WS=1
4	0.000062066	172.17.0.2	172.17.0.3	TCP	74	22 → 38662 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=175673435 TSecr=175673435
5	0.000070385	172.17.0.3	172.17.0.2	TCP	66	38662 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=985851249 TSecr=175673435
6	0.000223319	172.17.0.3	172.17.0.2	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
7	0.000227607	172.17.0.2	172.17.0.3	TCP	66	22 → 38662 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=175673436 TSecr=985851250
8	0.000608941	172.17.0.2	172.17.0.3	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
9	0.000622675	172.17.0.3	172.17.0.2	TCP	66	38662 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=985851250 TSecr=175673436
10	0.000712072	172.17.0.3	172.17.0.2	SSHv2	153	Client: Key Exchange Init
11	0.001370848	172.17.0.2	172.17.0.3	SSHv2	114	Server: Key Exchange Init
12	0.002382660	172.17.0.3	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
13	0.008419474	172.17.0.2	172.17.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (1)
14	0.052315988	172.17.0.3	172.17.0.2	TCP	66	38662 → 22 [ACK] Seq=1602 Ack=1718 Win=64128 Len=0 TSval=985851302 TSecr=175673444
15	2.002127756	172.17.0.3	172.17.0.2	SSHv2	82	Client: New Keys

Algorithms

Cookie: 0e8c0d9e994e162a1d7a745b5672f12a

key\_algorithms length: 241

key\_algorithms string [truncated]: curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-sha256

server\_host\_key\_algorithms length: 500

server\_host\_key\_algorithms string [truncated]: ecdsa-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@openssh.com,ecdsa-sha2-nistp521-cert-v01@openssh.com,ssh-rsa-cert-v01@openssh.com,ssh-rsa-cert-v01@openssh.com

encryption\_algorithms\_client\_to\_server length: 108

encryption\_algorithms\_client\_to\_server string: chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com

encryption\_algorithms\_server\_to\_client length: 108

encryption\_algorithms\_server\_to\_client string: chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com

mac\_algorithms\_client\_to\_server length: 213

mac\_algorithms\_client\_to\_server string [truncated]: umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha1

mac\_algorithms\_server\_to\_client length: 213

mac\_algorithms\_server\_to\_client string [truncated]: umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha1

compression\_algorithms\_client\_to\_server length: 26

compression\_algorithms\_client\_to\_server string: none,zlib@openssh.com,zlib

compression\_algorithms\_server\_to\_client length: 26

compression\_algorithms\_server\_to\_client string: none,zlib@openssh.com,zlib

languages\_client\_to\_server length: 0

languages\_client\_to\_server string:

languages\_server\_to\_client length: 0

languages\_server\_to\_client string:

First KEX Packet Follows: 0

Reserved: 00000000

[hashAlgorithms [truncated]: curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-sha256

[hashs: ae8bd7dd89978935a4c0e022adbbf30]

Figura 8: Tráfico C3 - S1

## 1.6 Tráfico generado por C4 (iface lo), detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)

### 1 DESARROLLO (PARTE 1)

Luego vemos que el cliente hace el intercambio de Key Init con un paquete de tamaño 1578 bytes, y si observamos el datagrama del paquete se puede apreciar los algoritmos para generar la llave, para encriptarla, los mensajes de autenticación y los algoritmos de compresión. Finalmente observamos que su HASSH generado es **ae8bd7dd09970555aa4c6ed22adbbf56**.

Finalmente el Cliente inicia el proceso de las curvas elípticas de Diffie Hellman y genera una nueva llave, estos paquetes pesan 114 y 82 bytes respectivamente.

## 1.6. Tráfico generado por C4 (iface lo), detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.2	172.17.0.6	TCP	74	49430 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=803027211 TSecr=0 WS=
2	0.000014141	172.17.0.6	172.17.0.2	TCP	74	22 → 49430 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2370992433
3	0.000022999	172.17.0.2	172.17.0.6	TCP	66	49430 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=803027211 TSecr=2370992433
4	0.000133039	172.17.0.2	172.17.0.6	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
5	0.000136915	172.17.0.6	172.17.0.2	TCP	66	22 → 49430 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=2370992433 TSecr=803027211
6	0.000560217	172.17.0.6	172.17.0.2	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
7	0.000568162	172.17.0.2	172.17.0.6	TCP	66	49430 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=803027212 TSecr=2370992434
8	0.000658762	172.17.0.2	172.17.0.6	SSHv2	1570	Client: Key Exchange Init
9	0.001339968	172.17.0.6	172.17.0.2	SSHv2	1146	Server: Key Exchange Init
10	0.040649904	172.17.0.2	172.17.0.6	SSHv2	1274	Client: Diffie-Hellman Key Exchange Init
11	0.040625130	172.17.0.6	172.17.0.2	SSHv2	1630	Server: Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
12	0.040633233	172.17.0.2	172.17.0.6	TCP	66	49430 → 22 [ACK] Seq=2754 Ack=2686 Win=64128 Len=0 TSval=803027260 TSecr=2370992482
13	0.065709656	172.17.0.2	172.17.0.6	SSHv2	82	Client: New Keys

Algorithms Cookie: ff63d633da1288c17f16b72e4ca51346 kex_algorithms length: 276 kex_algorithms string [truncated]: sntrup761x25519-sha512@openssh.com, curve25519-sha256, curve25519-sha256@libssh.org, ecdh-sha2-nistp256, ecdh-s server_host_key_algorithms length: 463 server_host_key_algorithms string [truncated]: ssh-ed25519-cert-v01@openssh.com, ecdsa-sha2-nistp256-cert-v01@openssh.com, ecdsa-sha2-nistp384-c encryption_algorithms_client_to_server length: 108 encryption_algorithms_client_to_server string: chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gc encryption_algorithms_server_to_client length: 108 encryption_algorithms_server_to_client string: chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gc mac_algorithms_client_to_server length: 213 mac_algorithms_client_to_server string [truncated]: umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-5 mac_algorithms_server_to_client length: 213 mac_algorithms_server_to_client string [truncated]: umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-5 compression_algorithms_client_to_server length: 26 compression_algorithms_client_to_server string: none, zlib@openssh.com, zlib compression_algorithms_server_to_client length: 26 compression_algorithms_server_to_client string: none, zlib@openssh.com, zlib languages_client_to_server length: 0 languages_client_to_server string: languages_server_to_client length: 0 languages_server_to_client string: First KEX Packet Follows: 0 Reserved: 00000000 [hashAlgorithms [truncated]: sntrup761x25519-sha512@openssh.com, curve25519-sha256, curve25519-sha256@libssh.org, ecdh-sha2-nistp256, ecdh-sha2-n [hash: 78c05d999799066a2b4554ce7b1585a6]
--

Figura 9: Tráfico C4 - S1

En la Figura 9 podemos observar el tráfico que se genera al conectar el cliente 4 con el servidor. Se puede observar el handshake de ssh en donde comparten las versiones de ssh que soportan. En el caso del cliente vemos en el paquete correspondiente que posee un tamaño de 107 bytes con una versión de SSH-2.0 OpenSSH 9.0p1.

Luego vemos que el cliente hace el intercambio de Key Init con un paquete de tamaño 1570 bytes, y si observamos el datagrama del paquete se puede apreciar los algoritmos para generar la llave, para encriptarla, los mensajes de autenticación y los algoritmos de compresión. Finalmente observamos que su HASSH generado es **78c05d999799066a2b4554ce7b1585a6**.

Finalmente el Cliente inicia el proceso de Diffie Hellman y genera una nueva llave, estos paquetes pesan 1274 y 82 bytes respectivamente.

## 1.7. Compara la versión de HASSH obtenida con la base de datos para validar si el cliente corresponde al mismo

Se buscaron las siguientes combinaciones de hash / versión de SSH en la base de datos hasshdb obtenida en módulos.

- 0e4584cb9f2dd077dbf8ba0df8112d8e / SSH-2.0-OpenSSH 7.3p1
- 06046964c022c6407d15a27b12a6a4fb / SSH-2.0-OpenSSH 7.7p1
- ae8bd7dd09970555aa4c6ed22adbbf56 / SSH-2.0-OpenSSH 8.3p1
- 78c05d999799066a2b4554ce7b1585a6 / SSH-2.0-OpenSSH 9.0p1

Sin embargo no se encontraron ni los Hashes ni las versiones exactas en la base de datos, esto se puede deber a las versiones particulares de ubuntu instaladas en los contenedores.

## 1.8. Tipo de información contenida en cada uno de los paquetes generados en texto plano

### 1.8.1. C1

En el contenedor 1 se pueden observar los siguientes campos más relevantes en texto plano:

#### Cliente

- **Protocolo Cliente:** SSH-2.0-OpenSSH\_7.3p1 Ubuntu-1ubuntu0.1
- **Key Algorithms (286):** curve25519-sha256@libssh.org, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group-exchange-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, diffie-hellman-group-exchange-sha1, diffie-hellman-group14-sha256, diffie-hellman-group14-sha1, ext-info-c
- **Encryption Algorithms (150):** chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gcm@openssh.com, aes128-cbc, aes192-cbc, aes256-cbc, 3des-cbc
- **MAC Algorithms (213):** umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com, hmac-sha1-etm@openssh.com, umac-64@openssh.com, umac-128@openssh.com, hmac-sha2-256, hmac-sha2-512, hmac-sha1
- **Compression Algorithms (26):** none, zlib@openssh.com, zlib
- **Hash:** 0e4584cb9f2dd077dbf8ba0df8112d8e
- **Llave pública:** 594b8140b067d4cabf54185bb047e456b990edb99e53304382411a18e5961d29

#### Servidor

- **Protocolo Servidor:** SSH-2.0-OpenSSH\_9.0p1 Ubuntu-1ubuntu7.3



- **Key Algorithms:** sntrup761x25519-sha512@openssh.com, curve25519-sha256, curve25519-sha256@libssh.org, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group-exchange-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, diffie-hellman-group14-sha256
- **Encryption Algorithms:** chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gcm@openssh.com
- **MAC Algorithms:** umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com, hmac-sha1-etm@openssh.com, umac-64@openssh.com, umac-128@openssh.com, hmac-sha2-256, hmac-sha2-512, hmac-sha1
- **Compression Algorithms:** none, zlib@openssh.com, zlib
- **Hash:** a984ff804585fabe3cd08f4b3849024a
- **ECDSA public key (Q C):** 0437e34cfa530c4862d460a81fbdb2e5d60345389cd53018d5e00d5c8a39

### 1.8.2. C2

En el contenedor 2 se pueden observar los siguientes campos más relevantes en texto plano:

#### Cliente

- **Protocolo Cliente:** SSH-2.0-OpenSSH\_7.7p1 Ubuntu-4ubuntu0.3
- **Key Algorithms (304):** curve25519-sha256@libssh.org, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group-exchange-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, diffie-hellman-group-exchange-sha1, diffie-hellman-group14-sha256, diffie-hellman-group14-sha1, ext-info-c
- **Encryption Algorithms (108):** chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gcm@openssh.com, aes128-cbc, aes192-cbc, aes256-cbc, 3des-cbc
- **MAC Algorithms (213):** umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com, hmac-sha1-etm@openssh.com, umac-64@openssh.com, umac-128@openssh.com, hmac-sha2-256, hmac-sha2-512, hmac-sha1
- **Compression Algorithms (26):** none, zlib@openssh.com, zlib
- **Hash:** 0e4584cb9f2dd077dbf8ba0df8112d8e
- **Llave pública Q C:** 39c40e9915f5086d2405c4f2c3bfa0b5d4ca695b03f0cf60eff0d543a8747008

Servidor (valores omitidos son iguales al caso C1)

- **ECDSA public key (Q):** 0437e34cfa530c4862d460a81fbdb2e5d60345389cd53018d5e00d5c8a3901

### 1.8.3. C3

En el contenedor 3 se pueden observar los siguientes campos más relevantes en texto plano:

#### Cliente

- **Protocolo Cliente:** SSH-2.0-OpenSSH\_8.3p1 Ubuntu-1ubuntu0.1
- **Key Algorithms (241):** curve25519-sha256, curve25519-sha256@libssh.org, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group-exchange-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, diffie-hellman-group14-sha256, ext-info-c
- **Encryption Algorithms (108):** chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gcm@openssh.com
- **MAC Algorithms (213):** umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com, hmac-sha1-etm@openssh.com, umac-64@openssh.com, umac-128@openssh.com, hmac-sha2-256, hmac-sha2-512, hmac-sha1
- **Compression Algorithms (26):** none, zlib@openssh.com, zlib
- **Hash:** ae8bd7dd09970555aa4c6ed22adbbf56
- **Llave pública Q C:** 37e0f2a2b43317ecca10a19e67cdd6523f400f425e9943c798cd73398cb02e5f

Servidor (valores omitidos son iguales al caso C1)

- **ECDSA public key (Q):** 0437e34cfa530c4862d460a81fbd2e5d60345389cd53018d5e00d5c8a3901

### 1.8.4. C4/S1

En el contenedor 4 se pueden observar los siguientes campos más relevantes en texto plano:

#### Cliente

- **Protocolo Cliente:** SSH-2.0-OpenSSH\_9.0p1 Ubuntu-1ubuntu7.3
- **Key Algorithms (276):** sntrup761x25519-sha512@openssh.com, curve25519-sha256, curve25519-sha256@libssh.org, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group-exchange-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, diffie-hellman-group14-sha256, ext-info-c
- **Encryption Algorithms (108):** chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gcm@openssh.com
- **MAC Algorithms (213):** umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com, hmac-sha1-etm@openssh.com, umac-64@openssh.com, umac-128@openssh.com, hmac-sha2-256, hmac-sha2-512, hmac-sha1
- **Compression Algorithms (26):** none, zlib@openssh.com, zlib
- **Hash:** 78c05d999799066a2b4554ce7b1585a6
- **Llave pública (e)**



**Servidor (valores omitidos son iguales al caso C1)**

- **Hash:** a984ff804585fabe3cd08f4b3849024a
- **Llave pública (f)**
- **EDSA public key:** b2ddf810b2d7d38efdf1bd12eb10c6d03b1df93812dab2ba9c5a6e7735f8ef1

## 1.9. Diferencia entre C1 y C2

Las principales diferencias entre C1 y C2 se centran en la cantidad y selección específica de algoritmos de clave, cifrado y autenticación utilizados. C1 y C2 comparten casi la misma versión general de OpenSSH en sistemas Ubuntu (7.3p1 y 7.7p1 respectivamente), C2 presenta una lista ligeramente más extensa de algoritmos de clave en comparación con C1 pero un menor tamaño en la cantidad de bytes al momento del traspaso de llaves. Esto sugiere que C2 puede ofrecer una configuración más robusta o actualizada en términos de opciones de seguridad para la autenticación y el intercambio de claves. El cifrado principal de ambos contenedores es chacha20-poly1305 con un intercambio de llaves de curve25519-sha256. El contenedor C1 posee Ubuntu 16.10 y el C2 18.10.

## 1.10. Diferencia entre C2 y C3

Entre C2 y C3 que poseen versiones de SSH de 7.7p1 y 8.3p1, sus diferencias se encuentran principalmente en la cantidad y selección de algoritmos de clave disponibles. C2 presenta una lista más amplia y detallada de algoritmos de clave en comparación con C3, pero C3 posee un tamaño en bytes considerablemente mayor en el intercambio de llaves, lo que puede indicar que varios de estos algoritmos fueron deprecados entre las actualizaciones de los sistemas. El contenedor C2 posee Ubuntu 18.10 y el C3 20.10.

## 1.11. Diferencia entre C3 y C4

La comparación entre C3 y C4/S1 al igual que las anteriores, poseen distintas versiones de Ubuntu y SSH (8.3p1 y 9.0p1), el tamaño del cambio de llaves es similar, y también lo es la cantidad de algoritmos utilizados, sin embargo C4 utiliza Diffie-Hellman en vez de las curvas elípticas que usan los otros tres contenedores. El encriptado de ambos contenedores es chacha20-poly1305, pero el contenedor C4 a diferencia del resto, para el intercambio de llaves utiliza sntrup761x25519-sha512. El contenedor C3 posee Ubuntu 20.10 y el C4 22.10.

## 2. Desarrollo (Parte 2)

### 2.1. Identificación del cliente SSH con versión “?”

Podemos ver en la Figura 1, entregada en la descripción de actividades que el intercambio de llaves tiene un tamaño de 1578 bytes y además se realiza un proceso de curvas elípticas de Diffie Helman, por lo que nos indica si comparamos con nuestros datos, que corresponde al contenedor C3.

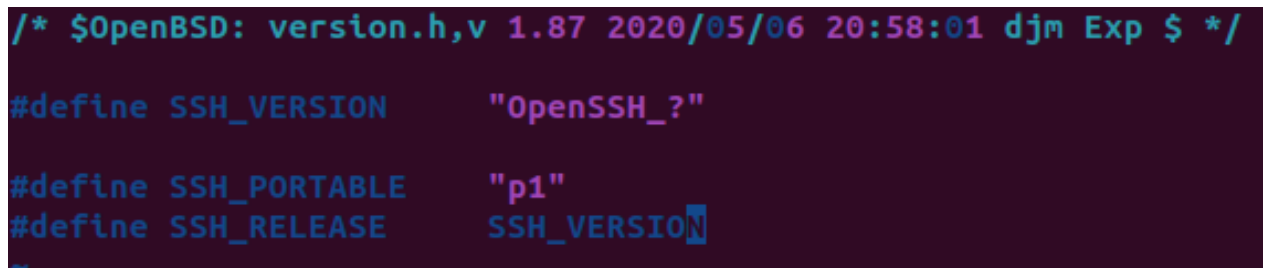
### 2.2. Replicación de tráfico al servidor (paso por paso)

Teniendo identificado el contenedor, sólo debemos modificar la versión de SSH manualmente, para esto ingresamos al contenedor y luego escribimos lo siguiente:

```
1 apt update
2 apt install autoconf libssl-dev zlib1g-dev gcc make git vim
3 wget https://cdn.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-8.3p1.
   tar.gz
4 tar xzf openssh-8.3p1.tar.gz
5 cd openssh-8.3p1
6 vim version.h
```

Código 5: Descarga requerimientos de edición

Seguido de esto, podemos editar la versión como un documento de texto, para que quede como se muestra en la Figura 10. Luego cargamos nuestros cambios según el siguiente código



```
/* $OpenBSD: version.h,v 1.87 2020/05/06 20:58:01 djm Exp $ */

#define SSH_VERSION      "OpenSSH_?"

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

Figura 10: Edición manual versión SSH

en consola dentro del contenedor.

```
1 autoreconf
2 ./configure
3 make
4 make install
5 /usr/local/sbin/sshd
```

Código 6: Actualización de versión SSH

Finalmente volvemos a iniciar la comunicación entre el contenedor 3 y el servidor S1, obteniendo como resultado lo requerido como se destacado en azul en la Figura 11.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.3	172.17.0.2	TCP	74	55662 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1001019486 TSecr=0 W
2	0.000041281	172.17.0.2	172.17.0.3	TCP	74	22 → 55662 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=190841672
3	0.000070894	172.17.0.3	172.17.0.2	TCP	66	55662 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1001019486 TSecr=190841672
4	0.000512346	172.17.0.3	172.17.0.2	SSHv2	85	Client: Protocol (SSH-2.0-OpenSSH_?)
5	0.000527088	172.17.0.2	172.17.0.3	TCP	66	22 → 55662 [ACK] Seq=1 Ack=20 Win=65152 Len=0 TSval=190841673 TSecr=1001019487
6	0.001350136	172.17.0.2	172.17.0.3	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
7	0.001356853	172.17.0.3	172.17.0.2	TCP	66	55662 → 22 [ACK] Seq=20 Ack=42 Win=64256 Len=0 TSval=1001019488 TSecr=190841674
8	0.001572859	172.17.0.3	172.17.0.2	SSHv2	1578	Client: Key Exchange Init
9	0.002108159	172.17.0.2	172.17.0.3	SSHv2	1146	Server: Key Exchange Init
10	0.005632304	172.17.0.3	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.012223580	172.17.0.2	172.17.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (
12	0.013519488	172.17.0.3	172.17.0.2	SSHv2	82	Client: New Keys
13	0.055451821	172.17.0.2	172.17.0.3	TCP	66	22 → 55662 [ACK] Seq=1718 Ack=1596 Win=64128 Len=0 TSval=190841728 TSecr=1001019500

Figura 11: Open SSH?

### 3. Desarrollo (Parte 3)

#### 3.1. Replicación del KEI con tamaño menor a 300 bytes (paso por paso)

Para la replicación del Key Init nos tenemos que percatar que corresponde a la generada por el servidor, por lo que tenemos que realizar un paso similar a la parte 2, pero en este caso en el contenedor C4 que contiene a nuestro servidor S1. Comenzamos de manera similar, pero en este caso descargamos la versión de openssh correspondiente a 9.0p1 y en vez de editar la versión como antes, editamos el archivo sshd config.

```

1 apt update
2 apt install autoconf libssl-dev zlib1g-dev gcc make git vim
3 wget https://cdn.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-9.0p1.
   tar.gz
4 tar xzf openssh-9.0p1.tar.gz
5 cd openssh-9.0p1
6 vim sshd_config

```

Código 7: Descarga requerimientos de edición

Editando este documento deberíamos modificar o agregar algoritmos para cambiar la cantidad de bytes que son generados en el proceso, como por ejemplo agregar la línea `KeyAlgorithms curve25519-sha256@libssh.org` que tiene un peso de 298 bytes en su algoritmo.

Luego deberíamos cargar nuestros cambios según el siguiente código en consola dentro del contenedor.

```

1 autoreconf
2 ./configure
3 make
4 make install
5 /usr/local/sbin/sshd

```

Código 8: Actualización de algoritmos

Lamentablemente no funcionaron estos cambios, dado que a pesar de varios intentos, no pude encontrar una combinación de algoritmos que baje la cantidad de bytes a lo esperado.

## Conclusiones y comentarios

En este laboratorio se aplicaron los conocimientos aprendidos durante todo el semestre, se tuvo que analizar tráfico, modificar documentos mediante sed o vim, utilizar contenedores de docker con versiones diferentes de sistemas operativos y "falsificar" suplantando información transmitida entre redes.

En conjunto se utilizaron los conocimientos de cifrado simétrico y asimétrico característicos de SSH, ya que generan simétricamente una llave de sesión para luego realizar una comunicación asimétrica. En la cual se logró identificar valores clave antes de su cifrado como los tipos de algoritmos utilizados para crear el HASSH, valor importante, ya que sabiendo estos algoritmos se puede recrear el hash.

Con los contenedores montados, se pudo editar la comunicación entre el servidor y un cliente, editando la versión del openssh de manera ficticia y luego se intentó modificar el tamaño de la generación de la llave init alterando los algoritmos utilizados en su creación, sin embargo, no se pudo debido a falta de conocimiento en peso o bits de ocupación que utiliza cada algoritmo diferente.

## Issues

El primer problema fue la creación correcta de cada contenedor, ya debían poseer versiones de Ubuntu deprecadas, y de estas no fue tan sencillo encontrar una librería funcional con repositorio antiguo funcional. Se solucionó probando con diferentes repositorios hasta encontrar <http://old-releases.ubuntu.com/ubuntu/>, que terminó funcionando y se agregaron en todas las sources.lists utilizadas.

El segundo problema fue la comunicación entre el servidor y los clientes, ya que a pesar de poder hacer correr ssh en cada contenedor, fue difícil poder encontrar cómo leer el tráfico, ya sea por descubrir que interfaz utilizar o a que IP. Se solucionó identificando tanto la interfaz de Docker0 junto con el Ip del servidor 172.17.0.0 dentro del contenedor C4/S1.

El tercer problema ocurrió al momento ver las diferencias entre los diferentes contenedores, si bien, a estas alturas ya sabemos leer bien los paquetes de wireshark, el contenido de los contenedores era similar, y era costoso identificar que valores realmente son importantes para ser comparados o que tengan una relevancia clave. Se solucionó sólo con repasar la materia aprendida, cuando se creaba una llave de sesión y cómo se encriptaban.

El último problema encontrado, que no se pudo completar en la experiencia, fue la edición de los algoritmos que debe utilizar el servidor para la generación de las llaves Init, fue simple identificar cual era el objetivo, pero encontrar el archivo que se debía modificar dentro de openssh y que se le debía agregar/cambiar no era nada intuitivo, ni simple de forzar a prueba y error. Se solucionó el problema de la parte 2, correctamente editando los valores de la versión

### 3.1 Replicación del KEI con tamaño menor a 300 bytes (paso DESARROLLO (PARTE 3))

de openssh, y en cuanto al problema de la parte 3 por lo menos se solucionó teóricamente que es lo que se debía realizar para disminuir el largo del paquete en la generación de las llaves.