

# Informe Laboratorio 5

## Sección 1

Alumno Diego Banda  
e-mail: diego.banda@mail\_udp.cl

Noviembre de 2025

# 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 . . . . .	8
1.1.2. C2 . . . . .	8
1.1.3. C3 . . . . .	8
1.1.4. C4/S1 . . . . .	8
1.2. Creación de las credenciales para S1 . . . . .	9
1.3. Tráfico generado por C1, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado) . . . . .	9
1.4. Tráfico generado por C2, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado) . . . . .	14
1.5. Tráfico generado por C3, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado) . . . . .	16
1.6. Tráfico generado por C4 (iface lo), detallando tamaño paquetes del flujo y el HASSH respectivo (detallado) . . . . .	18
1.7. Compara la versión de HASSH obtenida con la base de datos para validar si el cliente corresponde al mismo . . . . .	20
1.8. Tipo de información contenida en cada uno de los paquetes generados en texto plano . . . . .	20
1.8.1. C1 . . . . .	20
1.8.2. C2 . . . . .	21
1.8.3. C3 . . . . .	21
1.8.4. C4/S1 . . . . .	22
1.9. Diferencia entre C1 y C2 . . . . .	22
1.10. Diferencia entre C2 y C3 . . . . .	23
1.11. Diferencia entre C3 y C4 . . . . .	24
<b>2. Desarrollo (Parte 2)</b>	<b>25</b>
2.1. Identificación del cliente SSH con versión “?” . . . . .	25
2.2. Replicación de tráfico al servidor (paso por paso) . . . . .	25
<b>3. Desarrollo (Parte 3)</b>	<b>25</b>
3.1. Replicación del KEI con tamaño menor a 300 bytes (paso por paso) . . . . .	25
<b>4. Desarrollo (Parte 4)</b>	<b>27</b>
4.1. Explicación OpenSSH en general . . . . .	27
4.2. Capas de Seguridad en OpenSSH . . . . .	28
4.3. Identificación de que protocolos no se cumplen . . . . .	28

## Descripción de actividades

Para este último laboratorio, se solicita trabajar con Docker y el protocolo SSH, a fin de poder entender el concepto de criptografía asimétrica y firmas digitales.

Para lo anterior deberá:

- Crear 4 contenedores en Docker por medio de un DockerFile, 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, 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. 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.

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

4. Tomando en cuenta lo aprendido en este laboratorio, así como en los anteriores, explique el protocolo OpenSSH y las diferentes capas de seguridad que son parte del protocolo para garantizar los principios de seguridad de la información, integridad, confidencialidad, disponibilidad, autenticidad y no repudio. Es importante que sea muy específico en el objetivo del principio en el protocolo. En caso de considerar que alguno de los principios no se cumple, justifique su razonamiento. Es fundamental que su análisis se base en el tráfico SSH interceptado.

## 1. Desarrollo (Parte 1)

### 1.1. Códigos de cada Dockerfile

Se creó un *Dockerfile* para cada versión a utilizar, sin embargo y bajo indicación de profesor, se cambiaron tales versiones '.10' a '.04' debido a que las especificadas inicialmente son versiones *EOL* (*End Of Life*), las cuales no tienen soporte para sus repositorios *apt* haciendo que sea más engorroso utilizarlos, las versiones que terminan en 'x.04' son *LTS* (*Long Term Support*), las cuales a pesar de ser versiones antiguas siguen recibiendo soporte y tienen sus repositorios *apt* funcionales. Gracias a este cambio se podrán instalar los clientes ssh y el servidor para el correspondiente contenedor.

Para cada Dockerfile se creó una carpeta con el nombre correspondiente (C1, C2, etcétera),

y antes de cualquier paso posterior, una vez se tienen los dockerfiles se deben crear las imágenes a utilizar.

```
1 > docker build -t c1 ./C1
2 > docker build -t c2 ./C2
3 > docker build -t c3 ./C3
4 > docker build -t c4 ./C4
```

Listing 1: Creación de imágenes

```
diego@MacBook-Air-de-Diego: ~ / .. ografia / Lab05 - - zsh

> ls -R
C1 C2 C3 C4 Informe

./C1:
Dockerfile

./C2:
Dockerfile

./C3:
Dockerfile

./C4:
Dockerfile

./Informe:
Imagenes

./Informe / Imagenes:
```

Figura 3: Carpetas con archivos creados en proyecto

```
> docker build -t c1 ./C1
[+] Building 2.0s (7/7) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                      0.0s
=> => transferring dockerfile: 180B                                         0.0s
=> [internal] load metadata for docker.io/library/ubuntu:16.04            1.9s
=> [auth] library/ubuntu:pull token for registry-1.docker.io               0.0s
=> [internal] load .dockerignore                                         0.0s
=> => transferring context: 2B                                           0.0s
=> [1/2] FROM docker.io/library/ubuntu:16.04@sha256:1f1a2d56de1d604801a9671f301190704c25d604 0.0s
=> CACHED [2/2] RUN apt-get update && apt-get install -y openssh-client curl && rm - 0.0s
=> exporting to image                                                     0.0s
=> => exporting layers                                                   0.0s
=> => writing image sha256:8c9161c6c06bf7076c5c772924289c836eb52e099bfef54cb9980fec1f743e1 0.0s
=> => naming to docker.io/library/c1                                      0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/3qjnvzn3aco79hiua1dz3jzju5
```

Figura 4: Creación de imagen C1

## 1.1 Códigos de cada Dockerfile

```
> docker build -t c2 ./C2
[+] Building 0.4s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 180B
=> [internal] load metadata for docker.io/library/ubuntu:18.04
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/2] FROM docker.io/library/ubuntu:18.04@sha256:152dc042452c496007f07ca9127571cb9c29697f
=> CACHED [2/2] RUN apt-get update && apt-get install -y openssh-client curl && rm -r /var/lib/apt/lists/*
=> exporting to image
=> => exporting layers
=> => writing image sha256:5f76cd24539d93f74276733643f0a5b322cfbeda24b9ed7ff3c014321389993f
=> => naming to docker.io/library/c2

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/1rlhz4dtzc8aqtmb56q
j9op1
```

Figura 5: Creación de imagen C2

```
> docker build -t c3 ./C3
[+] Building 0.4s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 180B
=> [internal] load metadata for docker.io/library/ubuntu:20.04
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/2] FROM docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c42
=> CACHED [2/2] RUN apt-get update && apt-get install -y openssh-client curl && rm -r /var/lib/apt/lists/*
=> exporting to image
=> => exporting layers
=> => writing image sha256:0ce5f4bb79b9d826f4d7a8a9a771abede5ea55ad169506a9e1b93d04b2619499
=> => naming to docker.io/library/c3

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ksqy6bm3iuiaodv3rfd
myamhs
```

Figura 6: Creación de imagen C3

```
> docker build -t c4 ./C4
[+] Building 0.4s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 487B
=> [internal] load metadata for docker.io/library/ubuntu:22.04
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/8] FROM docker.io/library/ubuntu:22.04@sha256:104ae83764a5119017b8e8d6218fa0832b09df65
=> CACHED [2/8] RUN apt-get update && apt-get install -y openssh-client openssh-server curl
=> CACHED [3/8] RUN mkdir /var/run/sshd
=> CACHED [4/8] RUN useradd -m -s /bin/bash prueba
=> CACHED [5/8] RUN echo 'prueba:prueba' | chpasswd
=> CACHED [6/8] RUN sed -i 's/#PasswordAuthentication yes/PasswordAuthentication yes/' /etc/ssh/sshd_config
=> CACHED [7/8] RUN sed -i 's/UsePAM no/UsePAM yes/' /etc/ssh/sshd_config
=> CACHED [8/8] RUN ssh-keygen -A
=> exporting to image
=> => exporting layers
=> => writing image sha256:057d30e8d1099adc4e90ba84a07be8c768dd2b21f35a9ef67b3c63fe31a3e31c
=> => naming to docker.io/library/c4

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/tv16pfwej7fxxt53q5q
jj2pjo
```

Figura 7: Creación de imagen C4

### 1.1.1. C1

Se especifica la versión de ubuntu correspondiente, para luego a través de *apt* instalar *openssh-client* y finalmente quedar como una consola interactiva con su última línea.

```

1 FROM ubuntu:16.04
2
3 RUN apt-get update && \
4     apt-get install -y openssh-client curl && \
5     rm -rf /var/lib/apt/lists/*
6
7 CMD ["/bin/bash"]
```

Listing 2: Dockerfile para C1

### 1.1.2. C2

Realiza lo mismo que contenedor anterior.

```

1 FROM ubuntu:18.04
2
3 RUN apt-get update && \
4     apt-get install -y openssh-client curl && \
5     rm -rf /var/lib/apt/lists/*
6
7 CMD ["/bin/bash"]
```

Listing 3: Dockerfile para C2

### 1.1.3. C3

Realiza lo mismo que contenedor anterior.

```

1 FROM ubuntu:20.04
2
3 RUN apt-get update && \
4     apt-get install -y openssh-client curl && \
5     rm -rf /var/lib/apt/lists/*
6
7 CMD ["/bin/bash"]
```

Listing 4: Dockerfile para C3

### 1.1.4. C4/S1

Inicialmente realiza lo mismo que contenedor anterior, pero se le agregan unos pequeños cambios, primero se debe instalar *openssh-server* también, luego se crea una carpeta para ssh, el usuario 'prueba' para luego establecer la contraseña de este en 'prueba', con '*RUN ssh-keygen -A*' se crean las llaves necesarias para el funcionamiento del *ssh*, con '*EXPOSE 22*' se expone el puerto a utilizar y finalmente '*CMD [...]*' se ejecuta cuando se inicia el contenedor

(con *docker run*), manteniendo vivo el contenedor, con la flag '-D' se evita que se ejecuta en segundo plano y con '-e' se envían todos los registros al *stderr* en vez de anotarlo en un archivo, pudiendo ver estos a la hora de ejecutar el contenedor.

```

1 FROM ubuntu:22.04
2
3 RUN apt-get update && \
4     apt-get install -y openssh-client openssh-server curl && \
5     rm -rf /var/lib/apt/lists/*
6
7 RUN mkdir /var/run/sshd
8 RUN useradd -m -s /bin/bash prueba
9 RUN echo 'prueba:prueba' | chpasswd
10 RUN ssh-keygen -A
11
12 EXPOSE 22
13
14 CMD ["/usr/sbin/sshd", "-D", "-e"]

```

Listing 5: Dockerfile para C4

## 1.2. Creación de las credenciales para S1

Las credenciales se crean a la hora de crear la imagen de C4, esto a través de las siguientes líneas de su dockerfile:

```

1 RUN useradd -m -s /bin/bash prueba
2 RUN echo 'prueba:prueba' | chpasswd
3 RUN ssh-keygen -A

```

Listing 6: Dockerfile para C4

Donde:

- **Línea 1 y 2:** Crea el usuario prueba y su contraseña, siendo estas sus credenciales para iniciar sesión.
- **Línea 3:** Se crea la 'identidad' única del servidor, es la huella digital que el cliente guarda del servidor para identificarlo y evitar problemas de *MITM*.

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

Antes de comenzar la captura se crea una red local con *docker*, de esta manera se aísla del resto de las comunicaciones del computador, pudiendo mejorar el análisis y la búsqueda de paquetes.

```
> docker network create lab_cripto
```

Listing 7: Creación de network desde docker

Con la red local ya creada se podrán ejecutar los demás contenedores en esta red, y las capturas de red deben ser a esta y no la que normalmente se usaría. Existe un problema con *macOS* que si se está usando *linux* se puede omitir esta sección, pero en *linux* se pueden ver estas redes fácilmente desde wireshark, mientras que en *macOS* no, por lo tanto, para capturar los paquetes se realizará de la siguiente manera: Con la red creada, se inicia un nuevo contenedor de *nicolaka/netshoot*, este contenedor realizará una captura en un archivo de extensión *pcap* y luego este archivo se analiza con wireshark.

```
> docker run --rm -it --network=lab_cripto nicolaka/netshoot
```

Listing 8: Inicio contenedor de captura

```
> tcpdump -i eth0 -w /tmp/captura.pcap
```

Listing 9: Inicio de captura (dentro de netshoot) y almacenarla en *captura.pcap*

Una vez se desea dejar de capturar paquetes se termina la ejecución con Control + C, y en una terminal distinta de la utilizada para el contenedor se ejecuta lo siguiente:

```
> docker cp [ID_DEL_CONTENEDOR] :/tmp/captura.pcap ~/Desktop/mi_captura.pcap
```

Listing 10: Extracción de captura a sistema local

Con esto último la captura pasará a estar en el escritorio del dispositivo local, y abriendolo con wireshark se realiza el análisis de los paquetes y la comunicación.

Ahora se pasa a levantar primero el contenedor con el servidor de la siguiente manera:

```
> sudo docker run -d --name s1 --network lab_cripto c4
```

Listing 11: Contenedor con servidor

Donde:

- **-d:** Para ejecutarlo en segundo plano.
- **-name s1:** Para asignarle un nombre al contenedor.
- **--network lab\_cripto:** Para que se conecte a la red local especificada.
- **c4:** La imagen utilizada.

```
> sudo docker run -d --name s1 --network lab_cripto c4
>Password:
5eb819a19daf7dbe47b4ae8aeb6742268bf22e9803f72d8ca3282ba5f16a2008
> docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5eb819a19daf c4 "/usr/sbin/sshd -D -e" 3 seconds ago Up 3 seconds 22/tcp s1
6b84ea56f5fc nicolaka/netshoot "zsh" 24 minutes ago Up 24 minutes nice_fermi
```

Figura 8: Creación y verificación del contenedor de servidor

Posteriormente se levantan los contenedores de los clientes (en distintas terminales ya que son contenedores con terminal interactiva a través de la cual se pueden ejecutar comandos), a través de las siguientes líneas para cada uno:

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

## 1 DESARROLLO (PARTE 1)

```
1 > sudo docker run -it --rm --name c1 --network lab_cripto c1
2 > sudo docker run -it --rm --name c2 --network lab_cripto c2
3 > sudo docker run -it --rm --name c3 --network lab_cripto c3
4 > sudo docker run -it --rm --name c4 --network lab_cripto c4 bash
```

Listing 12: Inicio contenedores con clientes

Donde:

- **-it:** Para indicar que se quiere una terminal interactiva.
- **--rm:** Para que se elimine todo automáticamente una vez se termine de ejecutar el contenedor.
- **-name c1, c2, c3 o c4:** Para asignarle un nombre al contenedor.
- **--network lab\_cripto:** Para que se conecte a la red local especificada.

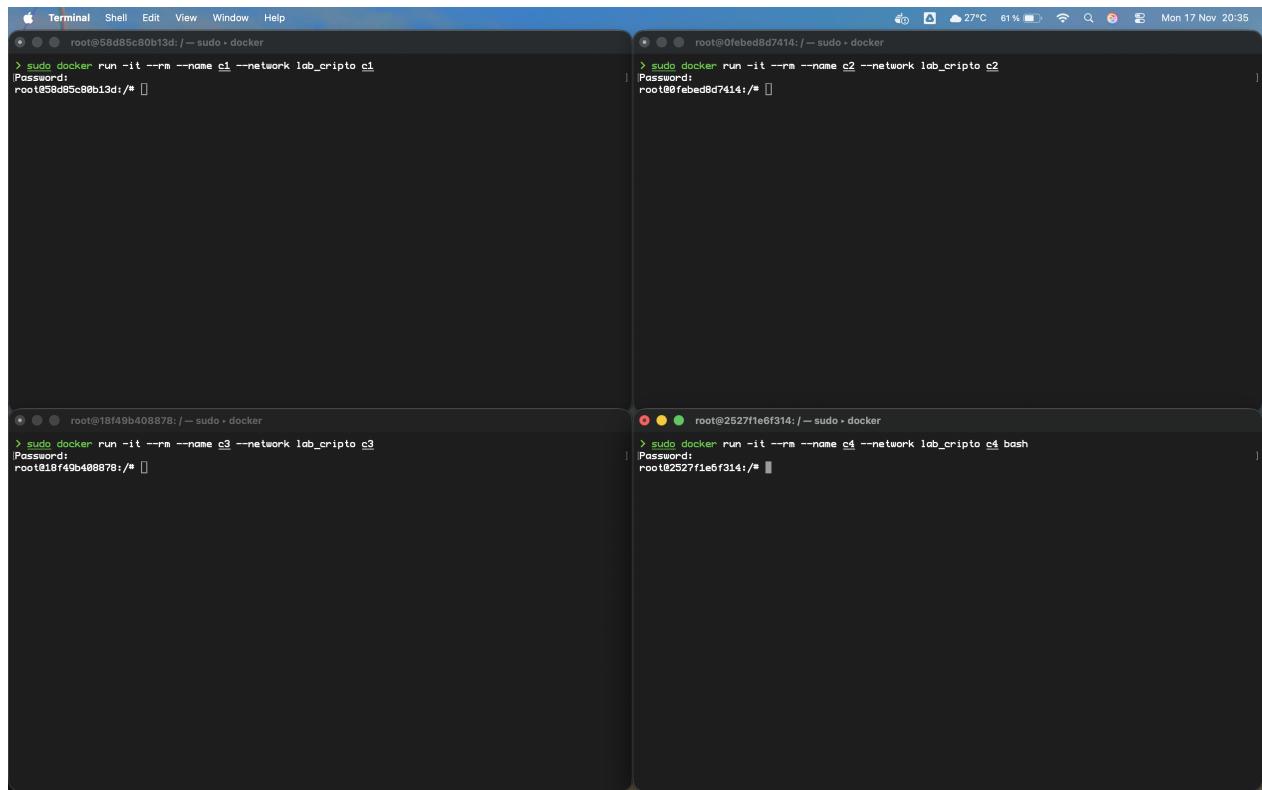


Figura 9: Inicio contenedores con clientes

Con todo esto listo se puede pasar a realizar las pruebas, iniciando la captura y ejecutando lo siguiente en cada terminal para conectarse a través del ssh al servidor, utilizando 'prueba' como se designó anteriormente.

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

1 DESARROLLO (PARTE 1)

```
1 root@xxxxxx...:/# ssh prueba@s1
```

Listing 13: Conexión ssh con servidor

```
> sudo docker run -it --rm --name c1 --network lab_cripto c1
>Password:
root@58d85c80b13d:/# ssh prueba@s1
The authenticity of host 's1 (172.18.0.3)' can't be established.
ECDSA key fingerprint is SHA256:Us/cp3CerC1bnQgqwX68HeNTCv0VnKHzLvhq4tbUkb4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 's1,172.18.0.3' (ECDSA) to the list of known hosts.
prueba@s1's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.10.14-linuxkit aarch64)

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

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.
```

Figura 10: Conexión ssh de cliente 1

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

No.	Time	Source	Destination	Protocol	Length	Info
1	82:fe:28:74:83:02	Broadcast	ARP	42	Who has 172.18.0.3? Tell 172.18.0.2	
2	82:fe:28:74:83:02	Broadcast	ARP	42	Who has 172.18.0.3? Tell 172.18.0.2	
3	0.0000082	76:b0:2a:41:42:92	82:fe:28:74:83:02	ARP	42	172.18.0.3 is at 76:b0:2a:41:42:92
4	0.0000083	172.18.0.2	172.18.0.3	TCP	74	34162 - 22 [SYN] Seq=0 Win=64248 Len=0 MSS=1460 SACK_PERM Tsvl=1961783117 Tsecr=0 WS=128
5	0.0000084	172.18.0.3	172.18.0.2	TCP	74	34162 - 22 [SYN] Seq=0 Win=64248 Len=0 MSS=1460 SACK_PERM Tsvl=1961783117 Tsecr=0 WS=128
6	0.0000017	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1 Win=64256 Len=0 Tsvl=1961783117 Tsecr=1086866024 Tsecr=1961783117 WS=128
7	0.003932	172.18.0.2	172.18.0.3	SSHv2	108	Client: Protocol (SSH-2.0-OpenSSH_7.3p2 Ubuntu-3ubuntu0.10)
8	0.0000020	172.18.0.3	172.18.0.2	TCP	66	2 - 34162 [ACK] Seq=1 Ack=43 Win=65152 Len=0 Tsvl=18969656028 Tsecr=1961783121
9	0.013509	172.18.0.3	172.18.0.2	SSHv2	108	Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.13)
10	0.0000019	172.18.0.3	172.18.0.2	TCP	66	34162 - 22 [ACK] Seq=43 Ack=43 Win=4256 Len=0 Tsvl=1961783135 Tsecr=1086866042
11	0.000251	172.18.0.2	172.18.0.3	SSHv2	1482	Client: Key Exchange Init
12	0.001349	172.18.0.3	172.18.0.2	SSHv2	1178	Server: Key Exchange Init
13	0.002276	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
14	0.004166	172.18.0.3	172.18.0.2	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
15	0.041569	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1427 Ack=1751 Win=69888 Len=0 Tsvl=1961783185 Tsecr=1086866050
16	1.872456	172.18.0.2	172.18.0.3	SSHv2	82	Client: New Keys
17	0.041665	172.18.0.3	172.18.0.2	TCP	66	34162 - 22 [ACK] Seq=1751 Ack=1443 Win=67072 Len=0 Tsvl=1086868086 Tsecr=1961785057
18	0.0000032	172.18.0.2	172.18.0.3	SSHv2	118	Client: Encrypted packet (len=44)
19	0.000055	172.18.0.3	172.18.0.2	TCP	66	22 - 34162 [ACK] Seq=1751 Ack=1487 Win=67072 Len=0 Tsvl=1086868086 Tsecr=1961785099
20	0.000203	172.18.0.3	172.18.0.2	SSHv2	118	Server: Encrypted packet (len=44)
21	0.0000037	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1795 Win=69888 Len=0 Tsvl=1961785099 Tsecr=1086868006
22	0.000140	172.18.0.2	172.18.0.3	SSHv2	134	Client: Encrypted packet (len=68)
23	0.008714	172.18.0.3	172.18.0.2	SSHv2	118	Server: Encrypted packet (len=52)
24	0.042497	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1555 Ack=1847 Win=69888 Len=0 Tsvl=1961785150 Tsecr=1086868015
25	2.421807	172.18.0.2	172.18.0.3	SSHv2	214	Client: Encrypted packet (len=148)
26	0.040367	172.18.0.3	172.18.0.2	TCP	66	2 - 34162 [ACK] Seq=1847 Ack=1703 Win=69888 Len=0 Tsvl=1086870520 Tsecr=1961787572
27	0.016061	172.18.0.3	172.18.0.2	SSHv2	93	Server: Encrypted packet (len=28)
28	0.000023	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1780 Ack=1875 Win=69888 Len=0 Tsvl=1961787629 Tsecr=1086870536
29	0.000142	172.18.0.2	172.18.0.3	SSHv2	178	Client: Encrypted packet (len=112)
30	0.0000034	172.18.0.3	172.18.0.2	TCP	66	2 - 34162 [ACK] Seq=1874 Ack=1856 Win=69888 Len=0 Tsvl=1086870536 Tsecr=1961787629
31	0.010681	172.18.0.3	172.18.0.2	SSHv2	694	Server: Encrypted packet (len=201)
32	0.041677	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1815 Ack=2503 Win=72832 Len=0 Tsvl=1961787681 Tsecr=1086870546
33	0.0000028	172.18.0.3	172.18.0.2	SSHv2	118	Server: Encrypted packet (len=44)
34	0.000011	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1815 Ack=2547 Win=72832 Len=0 Tsvl=1961787681 Tsecr=1086870588
35	0.000071	172.18.0.2	172.18.0.3	SSHv2	442	Client: Encrypted packet (len=376)
36	0.0000009	172.18.0.3	172.18.0.2	SSHv2	174	Server: Encrypted packet (len=108)
37	0.0000008	172.18.0.3	172.18.0.2	SSHv2	566	Server: Encrypted packet (len=500)
38	0.0000044	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=2191 Ack=3155 Win=74112 Len=0 Tsvl=1961787683 Tsecr=1086870589
39	0.005512	172.18.0.3	172.18.0.2	SSHv2	158	Server: Encrypted packet (len=92)
40	0.040886	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=2191 Ack=3247 Win=74112 Len=0 Tsvl=1961787729 Tsecr=1086870595

Figura 11: Paquetes capturados de cliente C1

No.	Time	Source	Destination	Protocol	Length	Info
5	0.0000017	172.18.0.2	172.18.0.3	TCP	66	> Frame 11: Packet on wire (11216 bytes), 1402 bytes captured (11216 bits)
6	0.0000017	172.18.0.2	172.18.0.3	SSHv2	108	Client: Protocol (SSH-2.0-OpenSSH_7.3p2 Ubuntu-3ubuntu0.10)
7	0.003932	172.18.0.2	172.18.0.3	TCP	66	2 - 34162 [ACK] Seq=1 Win=65152 Len=0 Tsvl=18969656028 Tsecr=1961783121
8	0.0000020	172.18.0.2	172.18.0.3	SSHv2	108	Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.13)
9	0.013509	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=43 Ack=43 Win=4256 Len=0 Tsvl=1961783135 Tsecr=1086866042
10	0.0000019	172.18.0.2	172.18.0.3	SSHv2	1482	Client: Key Exchange Init
11	0.000251	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
12	0.001349	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
13	0.002276	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
14	0.004166	172.18.0.2	172.18.0.3	SSHv2	662	Client: New Keys
15	0.041569	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1427 Ack=1751 Win=69888 Len=0 Tsvl=1961783185 Tsecr=1086866050
16	1.872456	172.18.0.2	172.18.0.3	SSHv2	82	Client: Key Exchange Init
17	0.0000017	172.18.0.2	172.18.0.3	TCP	66	> Frame 11: Packet on wire (11216 bytes), 1402 bytes captured (11216 bits)
18	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1178	Client: Key Exchange Init
19	0.0000017	172.18.0.2	172.18.0.3	SSHv2	114	Server: Key Exchange Init
20	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
21	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
22	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Client: New Keys
23	0.0000017	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1427 Ack=1751 Win=69888 Len=0 Tsvl=1961783185 Tsecr=1086866050
24	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1482	Client: Key Exchange Init
25	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
26	0.0000017	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
27	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
28	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Client: New Keys
29	0.0000017	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1427 Ack=1751 Win=69888 Len=0 Tsvl=1961783185 Tsecr=1086866050
30	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1482	Client: Key Exchange Init
31	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
32	0.0000017	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
33	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
34	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Client: New Keys
35	0.0000017	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1427 Ack=1751 Win=69888 Len=0 Tsvl=1961783185 Tsecr=1086866050
36	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1482	Client: Key Exchange Init
37	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
38	0.0000017	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
39	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
40	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Client: New Keys
41	0.0000017	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1427 Ack=1751 Win=69888 Len=0 Tsvl=1961783185 Tsecr=1086866050
42	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1482	Client: Key Exchange Init
43	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
44	0.0000017	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
45	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
46	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Client: New Keys
47	0.0000017	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1427 Ack=1751 Win=69888 Len=0 Tsvl=1961783185 Tsecr=1086866050
48	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1482	Client: Key Exchange Init
49	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
50	0.0000017	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
51	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
52	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Client: New Keys
53	0.0000017	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1427 Ack=1751 Win=69888 Len=0 Tsvl=1961783185 Tsecr=1086866050
54	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1482	Client: Key Exchange Init
55	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
56	0.0000017	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
57	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
58	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Client: New Keys
59	0.0000017	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1427 Ack=1751 Win=69888 Len=0 Tsvl=1961783185 Tsecr=1086866050
60	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1482	Client: Key Exchange Init
61	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
62	0.0000017	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
63	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
64	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Client: New Keys
65	0.0000017	172.18.0.2	172.18.0.3	TCP	66	34162 - 22 [ACK] Seq=1427 Ack=1751 Win=69888 Len=0 Tsvl=1961783185 Tsecr=1086866050
66	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1482	Client: Key Exchange Init
67	0.0000017	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
68	0.0000017	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
69	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
70	0.0000017	172.18.0.2	172.18.0.3	SSHv2	662	Client: New Keys
71	0.0000017	172.18.0.				

- **Packet length:** 1402 bytes.
- **SSH Packet Length:** 1332 bytes.
- **SSH Padding Length:** 5 bytes.
- **hassh:** 68e0ba85e1a818f7c49ea3f4b849bd15

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

```
> sudo docker run -it --rm --name c2 --network lab_cripto c2
>Password:
[root@0febed8d7414:/# ssh prueba@s1
The authenticity of host 's1 (172.18.0.3)' can't be established.
ECDSA key fingerprint is SHA256:Us/cp3CerC1bnQgqwX68HeNTCv0VnKHzLvhq4tbUkb4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 's1,172.18.0.3' (ECDSA) to the list of known hosts.
[prueba@s1's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.10.14-linuxkit aarch64)

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

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 Nov 17 23:44:19 2025 from 172.18.0.4
prueba@5eb819a19daf:~$ █
```

Figura 13: Conexión ssh de cliente 2

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

### 1 DESARROLLO (PARTE 1)

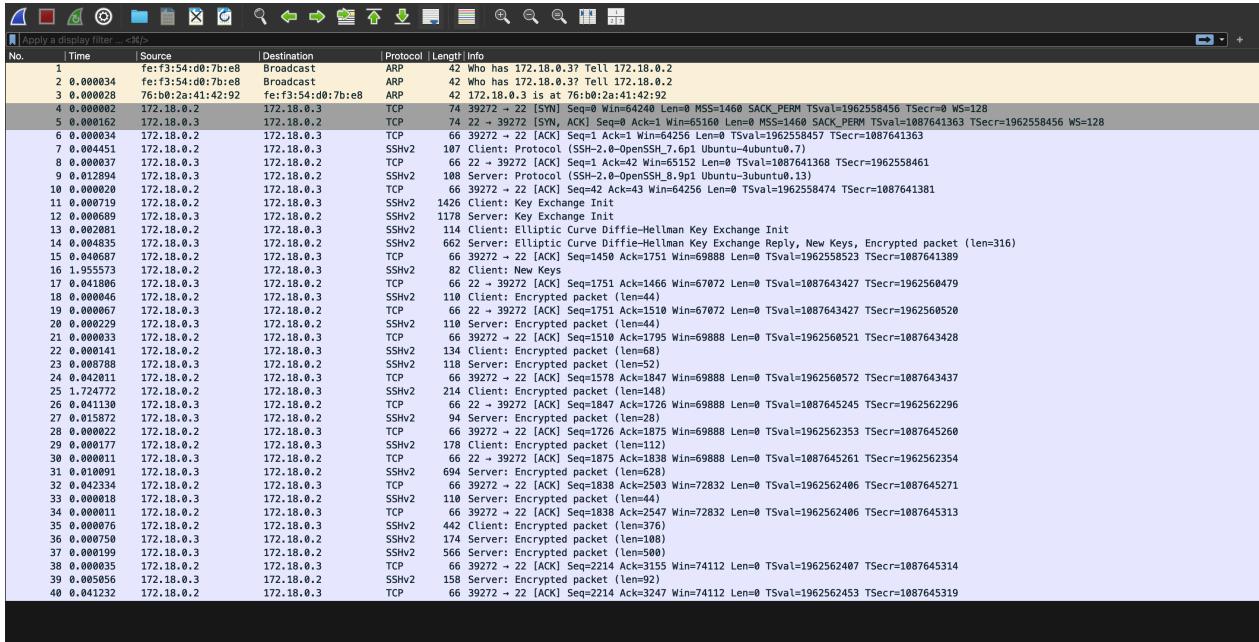


Figura 14: Paquetes capturados de cliente C2

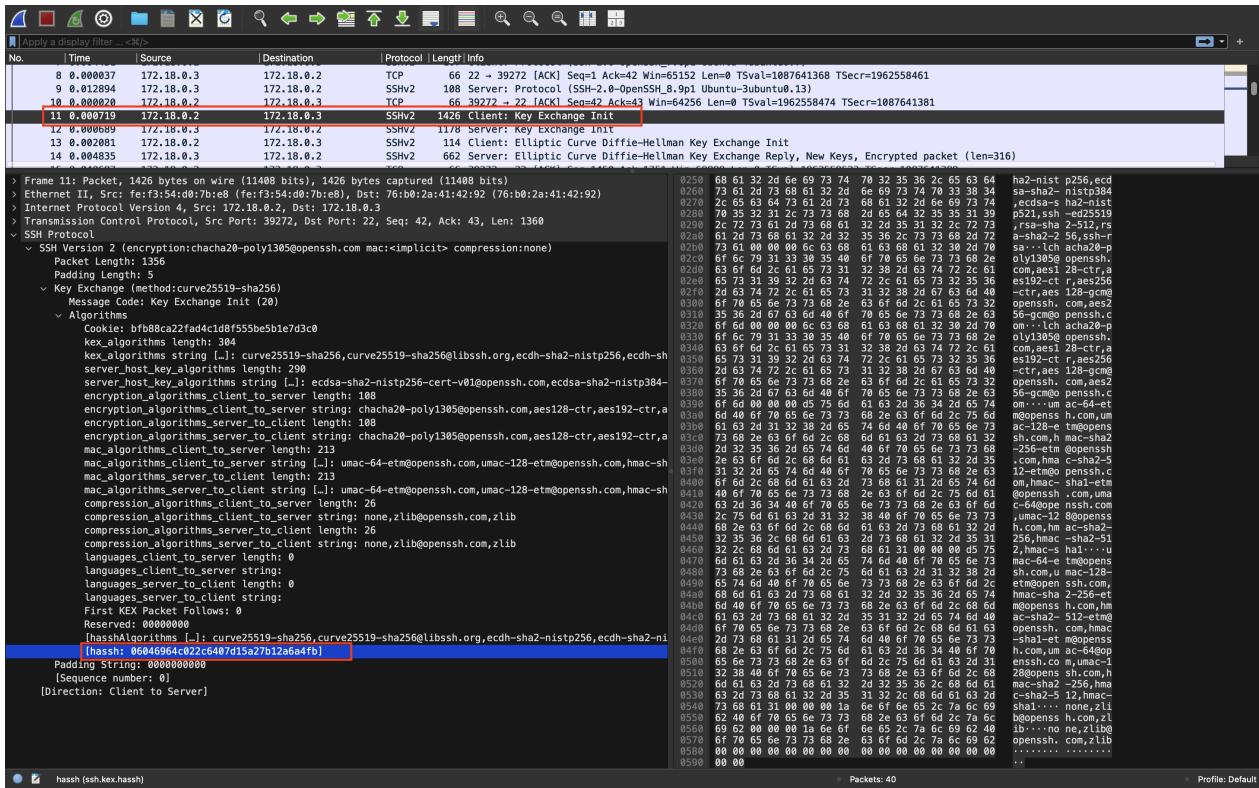


Figura 15: Key Exchange Init de C2

- **Packet length:** 1426 bytes.
- **SSH Packet Length:** 1356 bytes.
- **SSH Padding Length:** 5 bytes.
- **hassh:** 06046964c022c6407d15a27b12a6a4fb

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

```
> sudo docker run -it --rm --name c3 --network lab_cripto c3
[Password:
[root@18f49b408878:/# ssh prueba@s1
The authenticity of host 's1 (172.18.0.3)' can't be established.
ECDSA key fingerprint is SHA256:Us/cp3CerC1bnQgqwX68HeNTCv0VnKHzLvhq4tbUkb4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 's1,172.18.0.3' (ECDSA) to the list of known hosts.
[prueba@s1's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.10.14-linuxkit aarch64)

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

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 Nov 17 23:44:33 2025 from 172.18.0.5
prueba@5eb819a19daf:~$ ]
```

Figura 16: Conexión ssh de cliente 3

## 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		b6:6a:6c:0e:1a:db	Broadcast	ARP	42	Who has 172.18.0.3? Tell 172.18.0.2
2	0.000028	b6:6a:6c:0e:1a:db	Broadcast	ARP	42	Who has 172.18.0.3? Tell 172.18.0.2
3	0.000007	76:b0:2a:41:42:92	b6:6a:6c:0e:1a:db	ARP	42	172.18.0.3 is at 76:b0:2a:41:42:92
4	0.000016	172.18.0.2	172.18.0.3	TCP	74	51978 - 22 [SYN] Seq=0 Win=64248 Len=0 MSS=1460 SACK_PERM TSecr=0 WS=128
5	0.000016	172.18.0.2	172.18.0.3	TCP	74	51978 - 22 [SYN] Seq=0 Win=64248 Len=0 MSS=1460 SACK_PERM TSecr=0 WS=128
6	0.000019	172.18.0.2	172.18.0.3	TCP	66	51978 - 22 [ACK] Seq=1 Win=64256 Len=0 TSecr=1962824588 TSecv=1087907495
7	0.004519	172.18.0.2	172.18.0.3	SSHv2	108	Client: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-19.10) Seq=0 Win=64256 Len=0 TSecr=1962824588 TSecv=1087907495
8	0.000032	172.18.0.2	172.18.0.3	TCP	66	22 - 51978 [ACK] Seq=1 Ack=43 Win=65152 Len=0 TSecr=1962824593
9	0.011404	172.18.0.2	172.18.0.3	SSHv2	108	Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-19.10) Seq=0 Win=64256 Len=0 TSecr=1962824593
10	0.000019	172.18.0.2	172.18.0.3	TCP	66	51978 - 22 [ACK] Seq=43 Ack=43 Win=64256 Len=0 TSecr=1087907511
11	0.000322	172.18.0.2	172.18.0.3	SSHv2	1602	Client: Key Exchange Init
12	0.000024	172.18.0.2	172.18.0.3	TCP	66	22 - 51978 [ACK] Seq=43 Ack=1579 Win=67200 Len=0 TSecr=1962824604
13	0.001168	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
14	0.002459	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
15	0.004686	172.18.0.2	172.18.0.3	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
16	0.040709	172.18.0.2	172.18.0.3	TCP	66	51978 - 22 [ACK] Seq=1627 Ack=1751 Win=69888 Len=0 TSecr=1087907520
17	0.154084	172.18.0.2	172.18.0.3	SSHv2	182	Client: New Keys
18	0.042441	172.18.0.2	172.18.0.3	TCP	66	22 - 51978 [ACK] Seq=1751 Ack=1643 Win=67200 Len=0 TSecr=108790757 TSecr=1962826808
19	0.000046	172.18.0.2	172.18.0.3	SSHv2	110	Client: Encrypted packet (len=44)
20	0.000071	172.18.0.2	172.18.0.3	TCP	66	22 - 51978 [ACK] Seq=1751 Ack=1687 Win=67200 Len=0 TSecr=1962826850
21	0.000025	172.18.0.2	172.18.0.3	SSHv2	110	Server: Encrypted packet (len=44)
22	0.000042	172.18.0.2	172.18.0.3	TCP	66	51978 - 22 [ACK] Seq=1687 Ack=1795 Win=69888 Len=0 TSecr=108790757
23	0.0000205	172.18.0.2	172.18.0.3	SSHv2	13	Client: Encrypted packet (len=68)
24	0.009396	172.18.0.2	172.18.0.3	SSHv2	111	Server: Encrypted packet (len=52)
25	0.041534	172.18.0.2	172.18.0.3	TCP	66	51978 - 22 [ACK] Seq=1750 Ack=1847 Win=69888 Len=0 TSecr=108790767
26	2.223566	172.18.0.2	172.18.0.3	SSHv2	214	Client: Encrypted packet (len=148)
27	0.040415	172.18.0.2	172.18.0.3	TCP	66	22 - 51978 [ACK] Seq=1847 Ack=1983 Win=70144 Len=0 TSecr=1087912073 TSecr=1962829125
28	0.017603	172.18.0.2	172.18.0.3	SSHv2	95	Server: Encrypted packet (len=44)
29	0.000016	172.18.0.2	172.18.0.3	TCP	66	22 - 51978 [ACK] Seq=1847 Ack=1975 Win=69888 Len=0 TSecr=1087912090
30	0.000143	172.18.0.2	172.18.0.3	SSHv2	178	Client: Encrypted packet (len=19)
31	0.000019	172.18.0.2	172.18.0.3	TCP	66	22 - 51978 [ACK] Seq=1875 Ack=2015 Win=70144 Len=0 TSecr=1087912090 TSecr=1962829183
32	0.000099	172.18.0.2	172.18.0.3	SSHv2	694	Server: Encrypted packet (len=68)
33	0.043426	172.18.0.2	172.18.0.3	TCP	66	51978 - 22 [ACK] Seq=2015 Ack=2503 Win=72832 Len=0 TSecr=1087912099
34	0.000019	172.18.0.2	172.18.0.3	SSHv2	116	Server: Encrypted packet (len=44)
35	0.000019	172.18.0.2	172.18.0.3	TCP	66	51978 - 22 [ACK] Seq=2015 Ack=2547 Win=72832 Len=0 TSecr=1087912142
36	0.000086	172.18.0.2	172.18.0.3	SSHv2	442	Client: Encrypted packet (len=376)
37	0.000075	172.18.0.2	172.18.0.3	SSHv2	174	Server: Encrypted packet (len=108)
38	0.000018	172.18.0.2	172.18.0.3	SSHv2	566	Server: Encrypted packet (len=500)
39	0.000044	172.18.0.2	172.18.0.3	TCP	66	51978 - 22 [ACK] Seq=2391 Ack=3155 Win=74112 Len=0 TSecr=1087912143
40	0.004887	172.18.0.2	172.18.0.3	SSHv2	158	Server: Encrypted packet (len=92)
41	0.041591	172.18.0.2	172.18.0.3	TCP	66	51978 - 22 [ACK] Seq=2391 Win=74112 Len=0 TSecr=1087912148

Figura 17: Paquetes capturados de cliente C3

No.	Time	Source	Destination	Protocol	Length	Info
7	0.004519	172.18.0.2	172.18.0.3	SSHv2	108	Client: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-19.10) Seq=0 Win=64256 Len=0 TSecr=1962824593
8	0.000032	172.18.0.2	172.18.0.3	TCP	66	22 - 51978 [ACK] Seq=1 Ack=43 Win=65152 Len=0 TSecr=1087907500 TSecr=1962824593
9	0.011404	172.18.0.2	172.18.0.3	SSHv2	108	Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-19.10) Seq=0 Win=64256 Len=0 TSecr=1087907500 TSecr=1962824593
10	0.000019	172.18.0.2	172.18.0.3	TCP	66	51978 - 22 [ACK] Seq=43 Ack=43 Win=64256 Len=0 TSecr=1087907511
11	0.000322	172.18.0.2	172.18.0.3	SSHv2	1602	Client: Key Exchange Init
12	0.000024	172.18.0.2	172.18.0.3	TCP	66	22 - 51978 [ACK] Seq=43 Ack=1579 Win=67200 Len=0 TSecr=1087907512 TSecr=1962824604
13	0.001168	172.18.0.2	172.18.0.3	SSHv2	1178	Server: Key Exchange Init
14	0.002459	172.18.0.2	172.18.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
> Frame 11: Packet, 1602 bytes on wire (12816 bits), 1602 bytes captured (12816 bits)						
> Ethernet II, Src: b6:6a:6c:0e:1a:db (b6:6a:6c:0e:1a:db), Dst: 76:b0:2a:41:42:92 (76:b0:2a:41:42:92)						
> Internet Protocol Version 4, Src: 172.18.0.2, Dst: 172.18.0.3						
> Transmission Control Protocol, Src Port: 51978, Dst Port: 22, Seq: 22, Ack: 43, Len: 1536						
> SSH Protocol						
> SSH Version 2 (encryption:chacha20-poly1305@openssh.com mac:<implicit> compression:none)						
Packet Length: 1532						
Padding Length: 5						
< Key Exchange (method:curve25519-sha256)						
Message Key: Key Exchange Init (28)						
< Algorithms						
Cookie: 8135f71de20a540746f7cb332171f1						
key_algorithm_name: chacha20						
Padding length: 270						
key_algorithms string []: curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp512,ecdh-sha2-nistp8096,ecdh-sha2-nistp1024,ecdh-sha2-nistp1536,ecdh-sha2-nistp2304,ecdh-sha2-nistp3072,ecdh-sha2-nistp4096,ecdh-sha2-nistp5184,ecdh-sha2-nistp7168,ecdh-sha2-nistp10336,ecdh-sha2-nistp140704,ecdh-sha2-nistp201120,ecdh-sha2-nistp302240,ecdh-sha2-nistp403360,ecdh-sha2-nistp605680,ecdh-sha2-nistp808320,ecdh-sha2-nistp121248,ecdh-sha2-nistp181872,ecdh-sha2-nistp242544,ecdh-sha2-nistp363816,ecdh-sha2-nistp545728,ecdh-sha2-nistp818560,ecdh-sha2-nistp122736,ecdh-sha2-nistp184104,ecdh-sha2-nistp245152,ecdh-sha2-nistp367680,ecdh-sha2-nistp541128,ecdh-sha2-nistp812240,ecdh-sha2-nistp122448,ecdh-sha2-nistp184640,ecdh-sha2-nistp245696,ecdh-sha2-nistp368064,ecdh-sha2-nistp541280,ecdh-sha2-nistp814560,ecdh-sha2-nistp122896,ecdh-sha2-nistp184992,ecdh-sha2-nistp245984,ecdh-sha2-nistp369184,ecdh-sha2-nistp542560,ecdh-sha2-nistp819360,ecdh-sha2-nistp123312,ecdh-sha2-nistp185024,ecdh-sha2-nistp246016,ecdh-sha2-nistp369248,ecdh-sha2-nistp543840,ecdh-sha2-nistp819760,ecdh-sha2-nistp123728,ecdh-sha2-nistp185440,ecdh-sha2-nistp246432,ecdh-sha2-nistp369456,ecdh-sha2-nistp544480,ecdh-sha2-nistp819920,ecdh-sha2-nistp123952,ecdh-sha2-nistp185664,ecdh-sha2-nistp246648,ecdh-sha2-nistp369672,ecdh-sha2-nistp544960,ecdh-sha2-nistp819984,ecdh-sha2-nistp124160,ecdh-sha2-nistp185872,ecdh-sha2-nistp246856,ecdh-sha2-nistp369880,ecdh-sha2-nistp545280,ecdh-sha2-nistp819992,ecdh-sha2-nistp124368,ecdh-sha2-nistp186080,ecdh-sha2-nistp247056,ecdh-sha2-nistp370096,ecdh-sha2-nistp545600,ecdh-sha2-nistp819996,ecdh-sha2-nistp124576,ecdh-sha2-nistp186296,ecdh-sha2-nistp247256,ecdh-sha2-nistp370292,ecdh-sha2-nistp545920,ecdh-sha2-nistp819998,ecdh-sha2-nistp124784,ecdh-sha2-nistp186496,ecdh-sha2-nistp247456,ecdh-sha2-nistp370496,ecdh-sha2-nistp546240,ecdh-sha2-nistp819999,ecdh-sha2-nistp124992,ecdh-sha2-nistp186696,ecdh-sha2-nistp247648,ecdh-sha2-nistp370696,ecdh-sha2-nistp546480,ecdh-sha2-nistp819999,ecdh-sha2-nistp125104,ecdh-sha2-nistp186896,ecdh-sha2-nistp247840,ecdh-sha2-nistp370896,ecdh-sha2-nistp546720,ecdh-sha2-nistp819999,ecdh-sha2-nistp125312,ecdh-sha2-nistp187096,ecdh-sha2-nistp248032,ecdh-sha2-nistp371096,ecdh-sha2-nistp547040,ecdh-sha2-nistp819999,ecdh-sha2-nistp125520,ecdh-sha2-nistp187296,ecdh-sha2-nistp248224,ecdh-sha2-nistp371296,ecdh-sha2-nistp547280,ecdh-sha2-nistp819999,ecdh-sha2-nistp125728,ecdh-sha2-nistp187496,ecdh-sha2-nistp248416,ecdh-sha2-nistp371496,ecdh-sha2-nistp547520,ecdh-sha2-nistp819999,ecdh-sha2-nistp125936,ecdh-sha2-nistp187696,ecdh-sha2-nistp248608,ecdh-sha2-nistp371696,ecdh-sha2-nistp547760,ecdh-sha2-nistp819999,ecdh-sha2-nistp126144,ecdh-sha2-nistp187896,ecdh-sha2-nistp248796,ecdh-sha2-nistp371896,ecdh-sha2-nistp547920,ecdh-sha2-nistp819999,ecdh-sha2-nistp126352,ecdh-sha2-nistp188096,ecdh-sha2-nistp248984,ecdh-sha2-nistp372096,ecdh-sha2-nistp548160,ecdh-sha2-nistp819999,ecdh-sha2-nistp126560,ecdh-sha2-nistp188296,ecdh-sha2-nistp249176,ecdh-sha2-nistp372296,ecdh-sha2-nistp548400,ecdh-sha2-nistp819999,ecdh-sha2-nistp126768,ecdh-sha2-nistp188496,ecdh-sha2-nistp249368,ecdh-sha2-nistp372496,ecdh-sha2-nistp548640,ecdh-sha2-nistp819999,ecdh-sha2-nistp126976,ecdh-sha2-nistp188696,ecdh-sha2-nistp249556,ecdh-sha2-nistp372696,ecdh-sha2-nistp548880,ecdh-sha2-nistp819999,ecdh-sha2-nistp127184,ecdh-sha2-nistp188896,ecdh-sha2-nistp249744,ecdh-sha2-nistp372896,ecdh-sha2-nistp549120,ecdh-sha2-nistp819999,ecdh-sha2-nistp127392,ecdh-sha2-nistp189096,ecdh-sha2-nistp249932,ecdh-sha2-nistp373096,ecdh-sha2-nistp549360,ecdh-sha2-nistp819999,ecdh-sha2-nistp127510,ecdh-sha2-nistp189296,ecdh-sha2-nistp249996,ecdh-sha2-nistp373296,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp127718,ecdh-sha2-nistp189496,ecdh-sha2-nistp249996,ecdh-sha2-nistp373496,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp127926,ecdh-sha2-nistp189696,ecdh-sha2-nistp249996,ecdh-sha2-nistp373696,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp128134,ecdh-sha2-nistp189896,ecdh-sha2-nistp249996,ecdh-sha2-nistp373896,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp128342,ecdh-sha2-nistp190096,ecdh-sha2-nistp249996,ecdh-sha2-nistp374096,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp128550,ecdh-sha2-nistp190296,ecdh-sha2-nistp249996,ecdh-sha2-nistp374296,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp128758,ecdh-sha2-nistp190496,ecdh-sha2-nistp249996,ecdh-sha2-nistp374496,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp128966,ecdh-sha2-nistp190696,ecdh-sha2-nistp249996,ecdh-sha2-nistp374696,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp129174,ecdh-sha2-nistp190896,ecdh-sha2-nistp249996,ecdh-sha2-nistp374896,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp129382,ecdh-sha2-nistp191096,ecdh-sha2-nistp249996,ecdh-sha2-nistp375096,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp129590,ecdh-sha2-nistp191296,ecdh-sha2-nistp249996,ecdh-sha2-nistp375296,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp129798,ecdh-sha2-nistp191496,ecdh-sha2-nistp249996,ecdh-sha2-nistp375496,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp129996,ecdh-sha2-nistp191696,ecdh-sha2-nistp249996,ecdh-sha2-nistp375696,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp130104,ecdh-sha2-nistp191896,ecdh-sha2-nistp249996,ecdh-sha2-nistp375896,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp130312,ecdh-sha2-nistp192096,ecdh-sha2-nistp249996,ecdh-sha2-nistp376096,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp130520,ecdh-sha2-nistp192296,ecdh-sha2-nistp249996,ecdh-sha2-nistp376296,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp130728,ecdh-sha2-nistp192496,ecdh-sha2-nistp249996,ecdh-sha2-nistp376496,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp130936,ecdh-sha2-nistp192696,ecdh-sha2-nistp249996,ecdh-sha2-nistp376696,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp131144,ecdh-sha2-nistp192896,ecdh-sha2-nistp249996,ecdh-sha2-nistp376896,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp131352,ecdh-sha2-nistp193096,ecdh-sha2-nistp249996,ecdh-sha2-nistp377096,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp131560,ecdh-sha2-nistp193296,ecdh-sha2-nistp249996,ecdh-sha2-nistp377296,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp131768,ecdh-sha2-nistp193496,ecdh-sha2-nistp249996,ecdh-sha2-nistp377496,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp131976,ecdh-sha2-nistp193696,ecdh-sha2-nistp249996,ecdh-sha2-nistp377696,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp132184,ecdh-sha2-nistp193896,ecdh-sha2-nistp249996,ecdh-sha2-nistp377896,ecdh-sha2-nistp549600,ecdh-sha2-nistp819999,ecdh-sha2-nistp132392,ecdh-sha2-n						

- SSH Packet Length: 1532 bytes.
- SSH Padding Length: 5 bytes.
- hassh: c11b200866cf918393e62ea25d851d90

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

```
> sudo docker run -it --rm --name c4 --network lab_cripto c4 bash
[Password:
[root@2527f1e6f314:/# ssh prueba@s1
The authenticity of host 's1 (172.18.0.3)' can't be established.
ED25519 key fingerprint is SHA256:Z/fC9fZFzpxgs6ha10oCsrXp5D+WsXhJXX02dvPmv9w.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 's1' (ED25519) to the list of known hosts.
[prueba@s1's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.10.14-linuxkit aarch64)

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

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 Nov 17 23:44:48 2025 from 172.18.0.6
prueba@5eb819a19daf:~$ ]
```

Figura 19: Conexión ssh de cliente 4

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

### 1 DESARROLLO (PARTE 1)

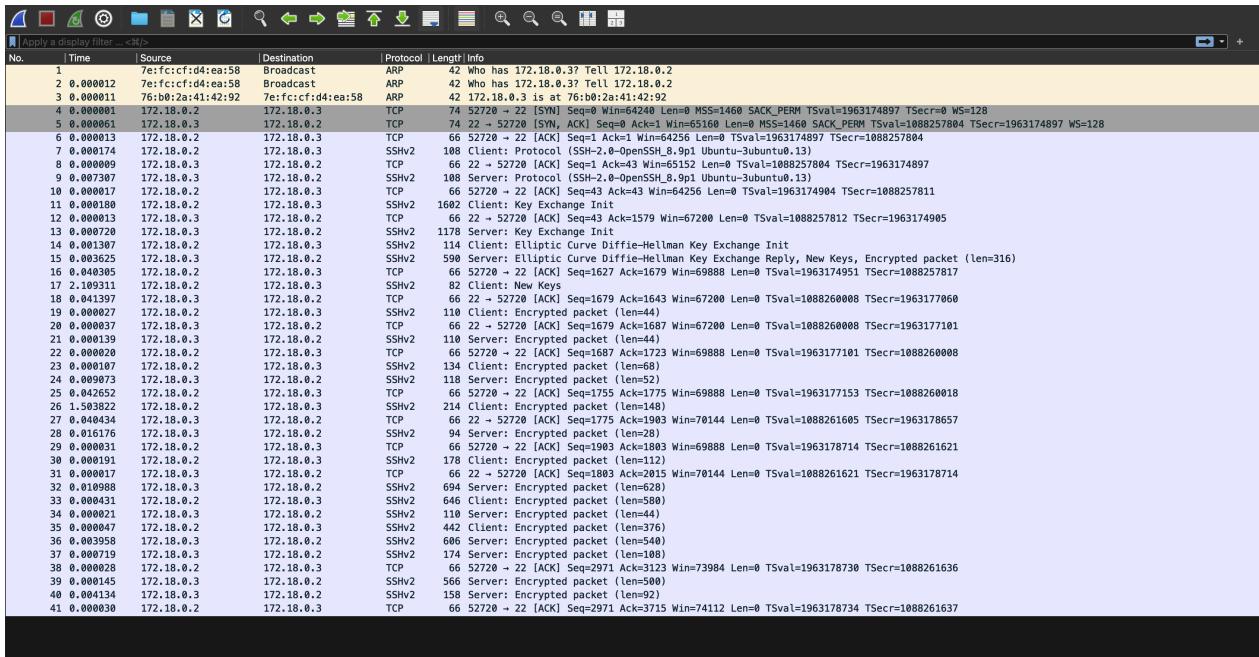


Figura 20: Paquetes capturados de cliente C4

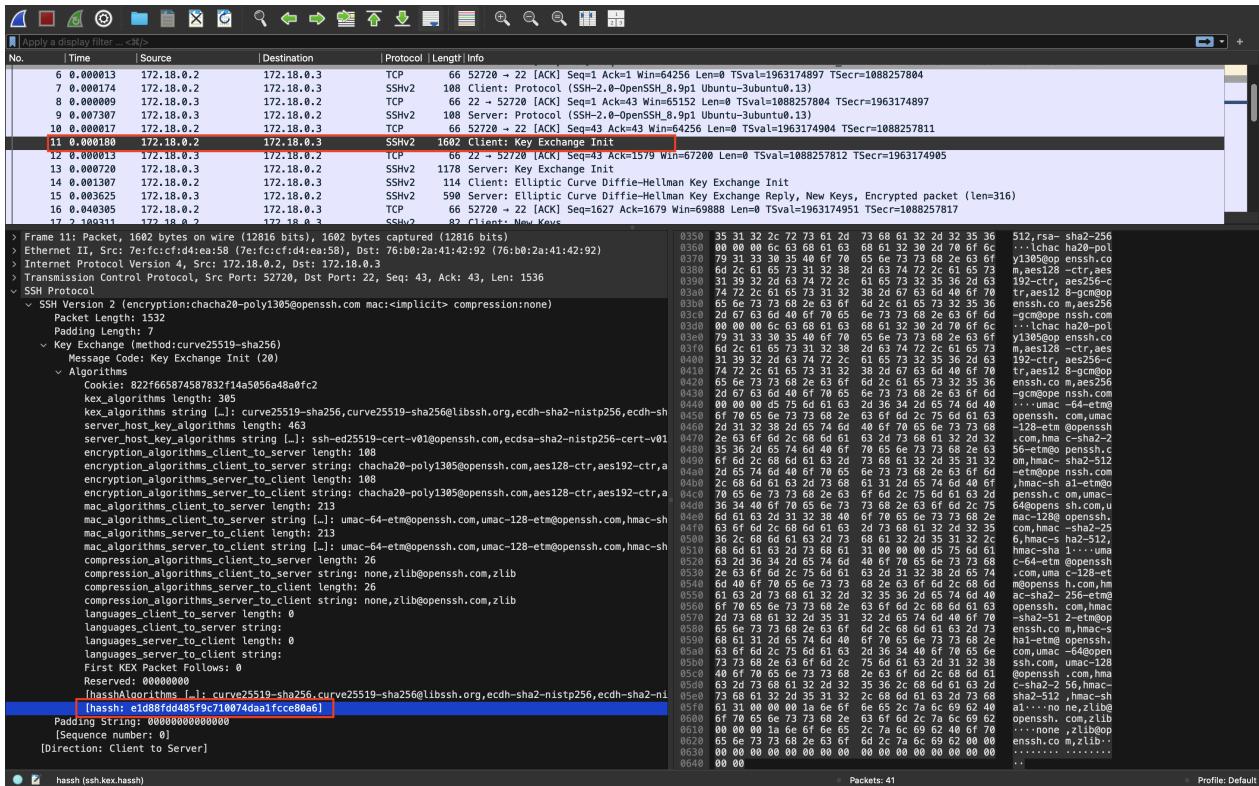


Figura 21: Key Exchange Init de C4

- **Packet length:** 1602 bytes
- **SSH Packet Length:** 1532 bytes.
- **SSH Padding Length:** 7 bytes.
- **hassh:** e1d88fdd485f9c710074daa1fcce80a6

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

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

Los paquetes que se envían en texto plano a la hora de realizar una conexión *ssh* son 2 principalmente, o 4 si consideramos los mensajes de parte del servidor también, estos son la identificación inicial *Client: Protocol* en donde se envía qué software y versión son. El segundo paquete enviado en texto plano es el de *Key Exchange Initiation* visto anteriormente, este es el paquete inicial más importante, ya que contiene la lista de algoritmos criptográficos soportados (intercambio de claves, cifrado, autenticación de mensajes y algoritmos de compresión), con este intercambio se genera el *hassh* respecto de los algoritmos que se usarán.

### 1.8.1. C1

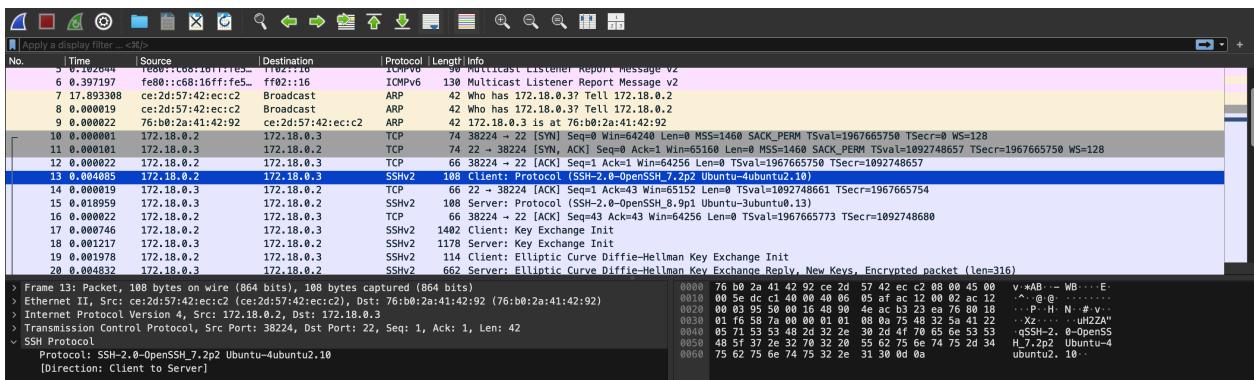


Figura 22: 'Client: Protocol' de C1

## 1.8 Tipo de información contenida en cada uno de los paquetes DESARROLLO (PARTE 1)

### 1.8.2. C2

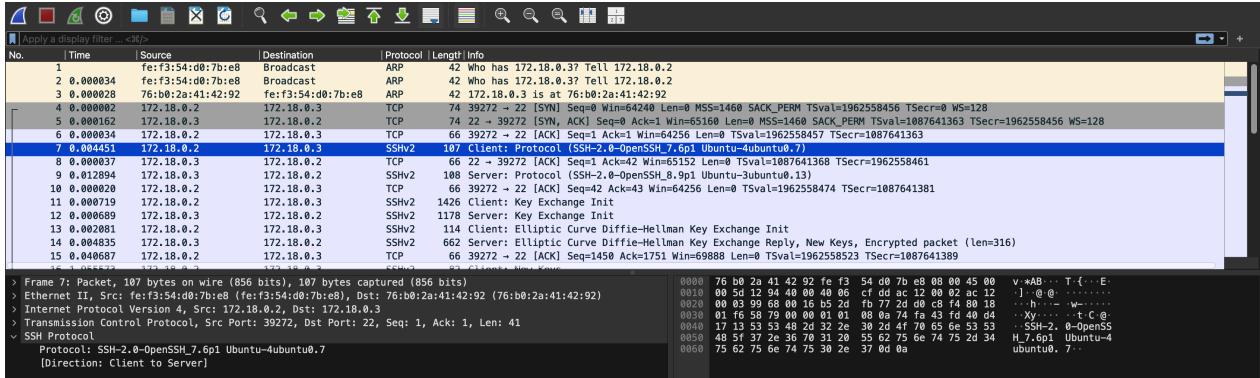


Figura 23: 'Client: Protocol' de C2

### 1.8.3. C3

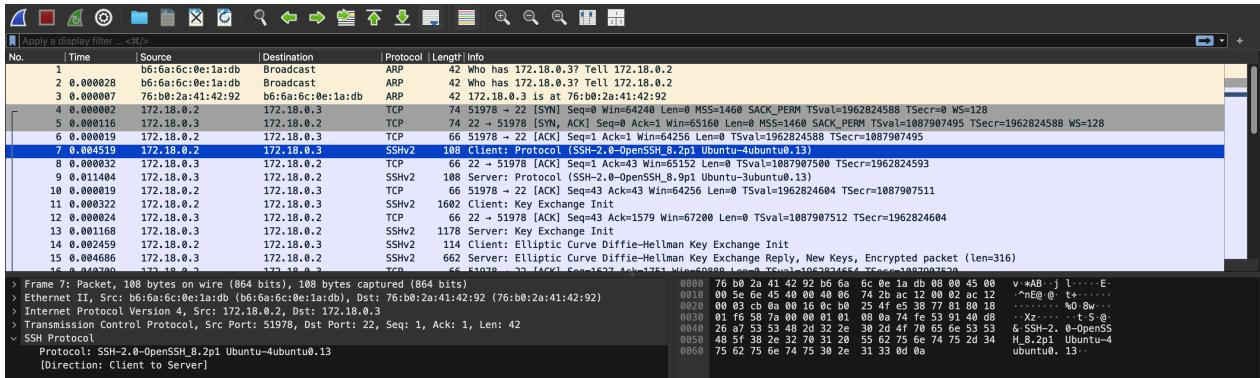


Figura 24: 'Client: Protocol' de C3

## 1.9 Diferencia entre C1 y C2

# 1 DESARROLLO (PARTE 1)

### 1.8.4. C4/S1

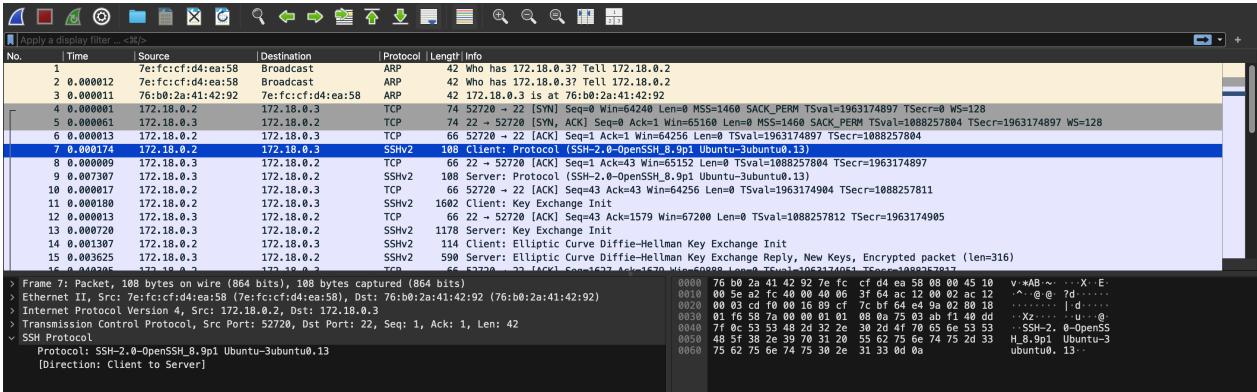


Figura 25: 'Client: Protocol' de C4

## 1.9. Diferencia entre C1 y C2

Tabla 1: Comparación entre C1 y C2

Diferencia	
<b>Versión OpenSSH</b>	C1 con 7.2p2 (Ubuntu 16.04) y C2 con 7.6p1 (Ubuntu 18.04). C2 tiene una versión más reciente.
<b>SSH packet length</b>	C1 con 1332 bytes y C2 con 1356 bytes. C2 tiene la mayor trama.
<b>Kex Algorithms</b>	C2 tiene los siguientes algoritmos que C1 no: curve25519-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, diffie-hellman-group14-sha256.
<b>Server Host Key Alg.</b>	Son iguales para C1 y C2.
<b>Encryption Algorithms</b>	En C2 se eliminaron los siguientes algoritmos respecto de C1: 3des-cbc, aes128-cbc, aes192-cbc, aes256-cbc. Estos algoritmos se eliminaron debido a los problemas de seguridad que podrían representar.
<b>MAC Algorithms</b>	Son iguales para C1 y C2.
<b>Compression Algorithms</b>	Son iguales para C1 y C2.

## 1.10. Diferencia entre C2 y C3

Tabla 2: Comparación entre C2 y C3

<b>Diferencia</b>	
<b>Versión OpenSSH</b>	C2 con 7.6p1 (Ubuntu 18.04), C3 con 8.2p1 (Ubuntu 20.04). C3 tiene una versión más reciente.
<b>SSH packet length</b>	C2 con 1356 bytes y C3 con 1532 bytes. C3 tiene la mayor trama.
<b>Kex Algorithms</b>	En C3 se agregó <code>kex-strict-c-v00@openssh.com</code> , y se eliminaron <code>diffie-hellman-group-exchange-sha1</code> y <code>diffie-hellman-group14-sha1</code> con respecto a C2. Las eliminaciones son para mejorar la seguridad, mientras que el nuevo añadido es para mitigar ataques de MITM.
<b>Server Host Key Alg.</b>	En C3 se agregaron <code>sk-ecdsa-sha2-nistp256@openssh.com</code> y <code>sk-ssh-ed25519@openssh.com</code> con sus certificados; estas permiten certificaciones a nivel físico.
<b>Encryption Algorithms</b>	Son iguales para C2 y C3.
<b>MAC Algorithms</b>	Son iguales para C2 y C3.
<b>Compression Algorithms</b>	Son iguales para C2 y C3.

## 1.11. Diferencia entre C3 y C4

Tabla 3: Comparación entre C3 y C4

<b>Diferencia</b>	
<b>Versión OpenSSH</b>	C3 con 8.2p1 (Ubuntu 20.04), C4 con 8.9p1 (Ubuntu 22.04). C4 tiene una versión más reciente.
<b>SSH packet length</b>	C3 con 1532 bytes, C4 con 1532 bytes, utilizando la misma cantidad.
<b>Kex Algorithms</b>	En C4 se agregó <code>sntrup761x25519-sha512@openssh.com</code> , el cual es muy importante y seguro, ya que está pensado para protegerse contra ataques de futuras computadoras cuánticas.
<b>Server Host Key Alg.</b>	En C4 se eliminaron <code>ssh-rsa-cert-v01@openssh.com</code> y <code>ssh-rsa</code> para volverse más estricto con los algoritmos de cifrado, eliminando posibles fallas y quedándose con los más seguros.
<b>Encryption Algorithms</b>	Son iguales para C3 y C4.
<b>MAC Algorithms</b>	Son iguales para C3 y C4.
<b>Compression Algorithms</b>	Son iguales para C3 y C4.

## 2. Desarrollo (Parte 2)

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

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

## 3. Desarrollo (Parte 3)

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

Para poder reducir el *Key Exchange Init* se debe reducir la cantidad de opciones de algoritmos para cada sección (como *Kex Algorithm*, *HostKeyAlgorithm*, *Encryption* y más), si se envía solo uno de cada uno se reduce muchísimo la cantidad de bytes utilizados.

Para llegar a un *KEI* menor a 300 bytes primero se debe tener la red creada anteriormente en funcionamiento, al igual que el mismo servidor ubuntu, por lo tanto, se debe ejecutar lo siguiente:

```
1 > docker network create lab_cripto
2 > sudo docker run -d --name s1 --network lab_cripto c4
```

Listing 14: Contenedor con servidor

Una vez está ejecutándose el servidor, se inicializa un contenedor de ubuntu, idealmente en una consola nueva con el siguiente comando:

```
1 > sudo docker run -it --rm --name c4 --network lab_cripto c4 bash
```

Listing 15: Contenedor con servidor

Dentro de esta consola interactiva es donde se debe realizar la conexión al servidor *ssh*, directamente a través de este llamado se entregarán los parámetros que se quieren utilizar, siendo solo uno para cada categoría, asegurándonos que el servidor pueda ejecutar estos, a través de esta técnica se reducen las listas que envía por defecto, enviando ahora solo 1, minimizando la cantidad de bytes utilizados.

```
1 > ssh \
2 -o KexAlgorithms=curve25519-sha256 \
3 -o HostKeyAlgorithms=ssh-ed25519 \
4 -o Ciphers=aes128-ctr \
5 -o Macs=hmac-sha1 \
6 -o Compression=no \
7 -o PreferredAuthentications=password \
8 prueba@s1
```

Listing 16: Contenedor con servidor

Donde:

- ***ssh*:** Iniciando conexión.

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

- **KexAlgorithms=curve5519-sha256**: Solo ofrecer *curve5519-sha256* como método de *Key Exchange*.
- **HostKeyAlgorithms=ssh-ed25519**: Solo aceptar claves firmadas con *ssh-ed25519*.
- **Ciphers=aes128-ctr**: Solo utilizar *aes128-ctr* como algoritmo de cifrado simétrico.
- **Macs=hmac-sha1**: Solo usar *hmac-sha1* para verificar los paquetes.
- **Compression=no**: No utilizar compresión.
- **PreferredAuthentications=password**: Directamente utilizar solo la autenticación a través de contraseña.
- **pruebe@ssh**: Usuario y servidor *ssh* al que se está conectando.

La razón de la elección de estos comandos son por su disponibilidad de parte del cliente como en el servidor y sus nombres, ya que entre menor largo de nombre, menos bytes se utilizarán para *KEI*.

```
> sudo docker run -it --rm --name c4 --network lab_cripto c4 bash
root@fe76b4a45511:/# ssh \
-o KexAlgorithms=curve25519-sha256 \
-o HostKeyAlgorithms=ssh-ed25519 \
-o Ciphers=aes128-ctr \
-o Macs=hmac-sha1 \
-o Compression=no \
-o PreferredAuthentications=password \
[prueba@s1
The authenticity of host 's1 (172.18.0.3)' can't be established.
ED25519 key fingerprint is SHA256:Z/fC9fZFzpxgs6ha10oCsrXp5D+WsXhJXX02dvPmv9w.
This key is not known by any other names
[Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 's1' (ED25519) to the list of known hosts.
[prueba@s1's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.10.14-1ubuntukit aarch64)

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

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 19 00:55:27 2025 from 172.18.0.2
```

Figura 26: Conexión minimizando *KEI*

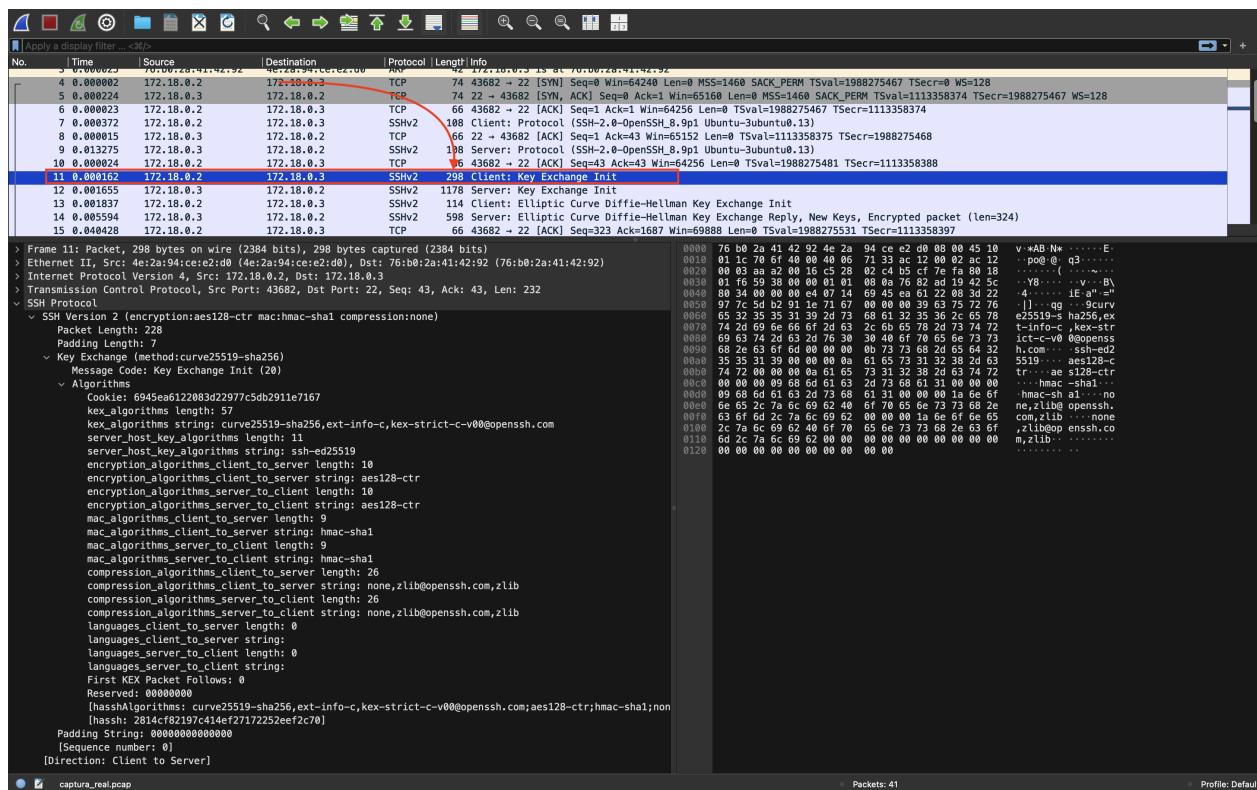


Figura 27: Captura de conexión

Como se observa en la imagen, la conexión efectivamente redujo el *length* a 298

## 4. Desarrollo (Parte 4)

### 4.1. Explicación OpenSSH en general

*OpenSSH* es un protocolo que tiene como objetivo realizar conexiones y operar servicios de red sobre una red insegura pero de una forma segura, a través de distintas capas logra una robustez y seguridad de muy alto nivel. Su objetivo a la hora de buscar toda esta seguridad es reemplazar a opciones anteriores como *telnet* o *ftp*, las cuales envían información (incluyendo claves privadas) en texto plano, siendo muy vulnerable a ataques.

*OpenSSH* consta de 3 principales componentes:

#### 1. *Transport Layer Protocol (TLP)*

- **Intercambio de Versión:** Cliente y Servidor se comunican sus versiones del protocolo.
- **Intercambio de Claves (*Key Exchange Init*):** Negociación de algoritmos, se establece una clave de sesión a través de distintos algoritmos ofrecidos entre sí.

- **Autenticación de Servidor:** Este firma lo intercambiado con su clave de host, permitiendo a los clientes verificar su autenticidad.
  - **Transición a Cifrado:** Todas las siguientes comunicaciones se realizan de manera encriptada, protegiéndolos de ataques de terceros o algún malicioso interceptando.
2. **User Authentication Protocol:** Una vez existe la seguridad en los canales a utilizar, el cliente se autentica con una contraseña.
  3. **Connection Protocol:** Se crean canales lógicos sobre el seguro para mantener una comunicación estable.

## 4.2. Capas de Seguridad en OpenSSH

1. **Confidencialidad:** Durante la fase de *Key Exchange Init* se busca utilizar algoritmos de cifrado simétrico fuertes con el objetivo de no tener falencias de seguridad. Una vez se estableció la conexión correctamente, todo paquete está completamente encriptado, de esta manera se asegura confidencialidad a la hora de la comunicación.
2. **Integridad:** A través de *MAC's* se logra mantener la integridad, a través de *hash* para cada paquete se verifican en el lado del receptor y en caso de no coincidir ambos, hubo algún error.
3. **Autenticidad:** A la hora de realizar una conexión se requiere atravesar un método de inicio de sesión con una contraseña (la del servidor en este caso), con esta debes autenticarte antes de conectarse con el servidor.

## 4.3. Identificación de que protocolos no se cumplen

1. **Disponibilidad:** A través de los algoritmos de seguridad mismos se busca contar con una alta disponibilidad, a través de su velocidad de conexión a pesar de lo robustos que son, sin embargo, la disponibilidad total no depende solo de los algoritmos empleados o la eficiencia de este, también de otras cosas como servidor, red, energía, o más, lo que lo hace difícil de considerar disponible ya que podrían ocurrir problemas con el resto de efectos externos y no poder hacer nada.
2. **No Repudio:** El no repudio es prevenir que una entidad pueda negar válidamente una acción que si ha realizado, esto no ya que a la hora de utilizar una contraseña, esta puede verse comprometida a través de un secreto mal contado, hackeo a persona que contenía tal contraseña, etc. Esto compromete el no repudio debido a que no se puede demostrar que fue tal persona en específico el que realizó una acción y no otra que robó la contraseña.

# Conclusiones y comentarios