Informe Laboratorio 5

Sección 1

Alumno: Kerssen Barros
e-mail: kerssen.barros@mail.udp.cl

Julio de 2024

Índice

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:

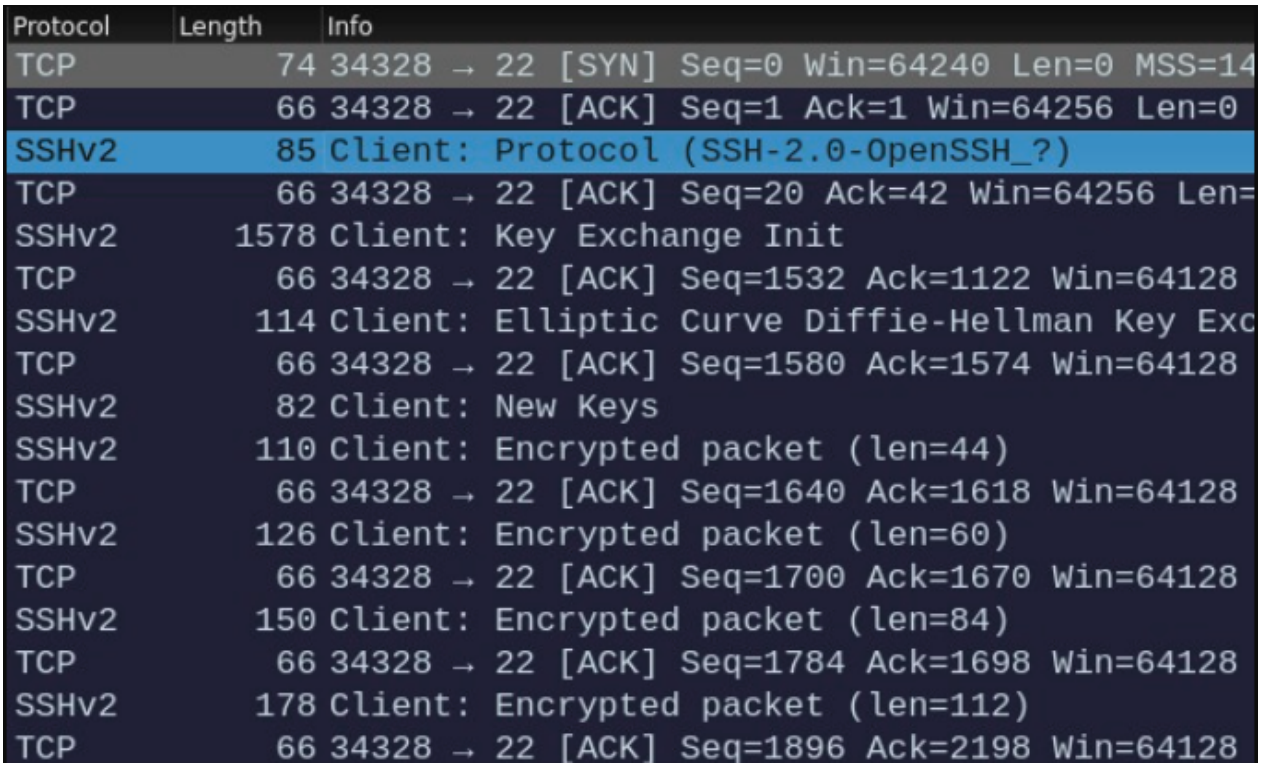
- Crear 4 contenedores en Docker o Podman, donde cada uno tendrá el siguiente SO: Ubuntu 16.10, Ubuntu 18.10, Ubuntu 20.10 y Ubuntu 22.10 a los cuales se llamarán C1, C2, C3 y C4 respectivamente.
  El equipo con Ubuntu 22.10 también será utilizado como S1.
- Para cada uno de ellos, deberá instalar el 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 $\rightarrow$ S1
  - C2 $\rightarrow$ S1
  - C3 $\rightarrow$ S1
  - C4 $\rightarrow$ 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.

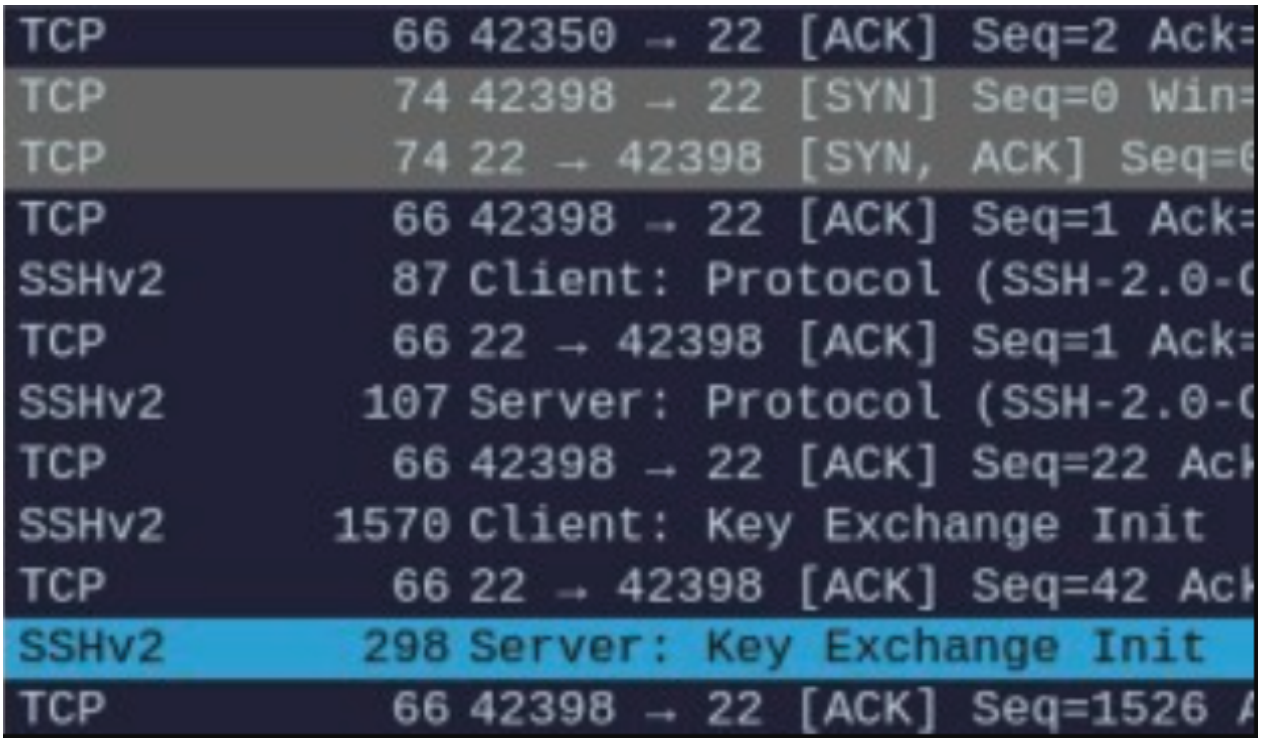


Figura 2: Captura del Key Exchange

1. Desarrollo (Parte 1)

1.1. Códigos de cada Dockerfile

A continuación se mostrarán las imágenes de cada *Dockerfile* utilizado para realizar el laboratorio:

1.1.1. C1

```
FROM ubuntu:16.10

RUN sed -i 's/archive.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    sed -i 's/security.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    apt-get update && apt-get install -y openssh-client

RUN apt-get install tcpdump -y

CMD ["tail", "-f", "/dev/null"]
```

Figura 3: Dockerfile del Cliente 1 (Ubuntu 16.10).

1.1.2. C2

```
FROM ubuntu:18.10

RUN sed -i 's/archive.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    sed -i 's/security.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    apt-get update && apt-get install -y openssh-client

RUN apt-get install tcpdump -y

CMD ["tail", "-f", "/dev/null"]
```

Figura 4: Dockerfile del Cliente 2 (Ubuntu 18.10).

1.1.3. C3

```
FROM ubuntu:20.10

RUN sed -i 's/archive.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    sed -i 's/security.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    apt-get update && apt-get install -y openssh-client

RUN apt-get install tcpdump -y

CMD ["tail", "-f", "/dev/null"]
```

Figura 5: Dockerfile del Cliente 3 (Ubuntu 20.10).

1.1.4. C4/S1

```
FROM ubuntu:22.10

RUN sed -i 's/archive.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    sed -i 's/security.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    apt-get update && apt-get install -y openssh-client openssh-server

RUN apt-get install tcpdump -y && apt-get update

## Para el paso 3
## COPY ./sshd_config /etc/ssh/sshd_config

RUN useradd -m -p $(openssl passwd -1 prueba) prueba
RUN sed -i 's/#PasswordAuthentication yes/PasswordAuthentication yes/' /etc/ssh/sshd_config

CMD ["/usr/sbin/sshd", "-D", "-e"]
```

Figura 6: Dockerfile del Cliente 4 / Servidor 1 (Ubuntu 22.10).

1.2. Creación de las credenciales para S1

La creación de credenciales se realizó en el mismo *Dockerfile* del Servidor en 1.1.4. Se utilizó el comando **useradd** para crear el usuario **prueba**, con las flags **−m** para crear automáticamente un directorio de inicio para este último y **−p** para especificar una contraseña.

La contraseña está codificada usando **openssl passwd -1**, que codifica la contraseña **prueba** usando el algoritmo de encriptación compatible con Linux. Finalmente se ejecuta el comando **sed −i 's/#PasswordAuthentication yes/PasswordAuthentication yes/' /etc/ssh/sshd_config** que edita el archivo de configuración SSH para permitir la autenticación por contraseña.

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

Se puede apreciar que en la captura de paquetes del **Cliente 1** se obtiene un **Key Exchange Init** con tamaño igual a **1498 bytes**, junto con el **HASSH** correspondiente a **0e4584cb9f2dd077dbf8ba0df8112d8e**.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.24.0.2	172.24.0.4	TCP	74	33622 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…
2	0.000056	172.24.0.4	172.24.0.2	TCP	74	22 → 33622 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA…
3	0.000075	172.24.0.2	172.24.0.4	TCP	66	33622 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=795278138 …
4	0.000957	172.24.0.2	172.24.0.4	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.3p1 Ubuntu-1ubuntu0.1)
5	0.001011	172.24.0.4	172.24.0.2	TCP	66	22 → 33622 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=353978645…
6	0.003680	172.24.0.4	172.24.0.2	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
7	0.003696	172.24.0.2	172.24.0.4	TCP	66	33622 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=79527814…
8	0.004007	172.24.0.2	172.24.0.4	SSHv2	1498	Client: Key Exchange Init
9	0.005528	172.24.0.4	172.24.0.2	SSHv2	1146	Server: Key Exchange Init
10	0.008712	172.24.0.2	172.24.0.4	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.015830	172.24.0.4	172.24.0.2	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New…
12	0.059231	172.24.0.2	172.24.0.4	TCP	66	33622 → 22 [ACK] Seq=1522 Ack=1718 Win=64128 Len=0 TSval=7952…
13	1.872700	172.24.0.2	172.24.0.4	SSHv2	82	Client: New Keys
14	1.915487	172.24.0.4	172.24.0.2	TCP	66	22 → 33622 [ACK] Seq=1718 Ack=1538 Win=64128 Len=0 TSval=3539…

Figura 7: Captura del tráfico del Cliente 1 con Key Exchange Init igual a 1498 bytes.

```
~/fatt  master ?:1 ✗
❯ python3 fatt.py -r /home/kerssen/Documentos/Cripto/lab5/trafico/trafico-c1.pcap -p
172.24.0.2:33622 -> 172.24.0.4:22 [SSH] hassh=0e4584cb9f2dd077dbf8ba0df8112d8e client=SSH-2.0-OpenSSH_7.3p1 Ubuntu-1ubuntu0.1
172.24.0.4:22 -> 172.24.0.2:33622 [SSH] hasshS=a984ff804585fabe3cd08f4b3849024a server=SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3
```

Figura 8: HASSH generado por el Cliente 1 utilizando fatt.

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

Se puede apreciar que en la captura de paquetes del **Cliente 2** se obtiene un **Key Exchange Init** con tamaño igual a **1426 bytes**, junto con el **HASSH** correspondiente a **06046964c022c6407d15a27b12a6a4fb**.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.24.0.3	172.24.0.4	TCP	74	60732 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…
2	0.000030	172.24.0.4	172.24.0.3	TCP	74	22 → 60732 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA…
3	0.000040	172.24.0.3	172.24.0.4	TCP	66	60732 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1704541652…
4	0.000494	172.24.0.3	172.24.0.4	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.7p1 Ubuntu-4ubuntu0.3)
5	0.000511	172.24.0.4	172.24.0.3	TCP	66	22 → 60732 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=814107948…
6	0.000821	172.24.0.4	172.24.0.3	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
7	0.000828	172.24.0.3	172.24.0.4	TCP	66	60732 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=17045416…
8	0.000974	172.24.0.3	172.24.0.4	SSHv2	1426	Client: Key Exchange Init
9	0.002093	172.24.0.4	172.24.0.3	SSHv2	1146	Server: Key Exchange Init
10	0.004767	172.24.0.3	172.24.0.4	SSHv2	114	Client: Diffie-Hellman Key Exchange Init
11	0.011719	172.24.0.4	172.24.0.3	SSHv2	662	Server: Diffie-Hellman Key Exchange Reply, New Keys, Encrypte…
12	0.053994	172.24.0.3	172.24.0.4	TCP	66	60732 → 22 [ACK] Seq=1450 Ack=1718 Win=64128 Len=0 TSval=1704…
13	1.637695	172.24.0.3	172.24.0.4	SSHv2	82	Client: New Keys

Figura 9: Captura del tráfico del Cliente 2 con Key Exchange Init igual a 1426 bytes.

```
~/fatt  P master ?:1 ✗
❯ python3 fatt.py -r /home/kerssen/Documentos/Cripto/lab5/trafico/trafico-c2.pcap -p
172.24.0.3:60732 -> 172.24.0.4:22 [SSH] hassh=06046964c022c6407d15a27b12a6a4fb client=SSH-2.0-OpenSSH_7.7p1 Ubuntu-4ubuntu0.3
172.24.0.4:22 -> 172.24.0.3:60732 [SSH] hasshS=a984ff804585fabe3cd08f4b3849024a server=SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3
```

Figura 10: HASSH generado por el Cliente 2 utilizando fatt.

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

Se puede apreciar que en la captura de paquetes del **Cliente 3** se obtiene un **Key Exchange Init** con tamaño igual a **1578 bytes**, junto con el **HASSH** correspondiente a **ae8bd7dd09970555aa4c6ed22adbbf56**.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.24.0.5	172.24.0.4	TCP	74	50682 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…
2	0.000035	172.24.0.4	172.24.0.5	TCP	74	22 → 50682 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA…
3	0.000046	172.24.0.5	172.24.0.4	TCP	66	50682 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2801327820…
4	0.000426	172.24.0.5	172.24.0.4	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
5	0.000443	172.24.0.4	172.24.0.5	TCP	66	22 → 50682 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=266621864…
6	0.000744	172.24.0.4	172.24.0.5	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
7	0.000748	172.24.0.5	172.24.0.4	TCP	66	50682 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=28013278…
8	0.000856	172.24.0.5	172.24.0.4	SSHv2	1578	Client: Key Exchange Init
9	0.001578	172.24.0.4	172.24.0.5	SSHv2	1146	Server: Key Exchange Init
10	0.002986	172.24.0.5	172.24.0.4	SSHv2	114	Client: Diffie-Hellman Key Exchange Init
11	0.007251	172.24.0.4	172.24.0.5	SSHv2	662	Server: Diffie-Hellman Key Exchange Reply, New Keys, Encrypte…
12	0.049145	172.24.0.5	172.24.0.4	TCP	66	50682 → 22 [ACK] Seq=1602 Ack=1718 Win=64128 Len=0 TSval=2801…
13	1.042092	172.24.0.5	172.24.0.4	SSHv2	82	Client: New Keys

Figura 11: Captura del tráfico del Cliente 3 con Key Exchange Init igual a 1578 bytes.

```
~/fatt  master ?:1 ✗
❯ python3 fatt.py -r /home/kerssen/Documentos/Cripto/lab5/trafico/trafico-c3.pcap -p
172.24.0.5:50682 -> 172.24.0.4:22 [SSH] hassh=ae8bd7dd09970555aa4c6ed22adbbf56 client=SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1
172.24.0.4:22 -> 172.24.0.5:50682 [SSH] hasshS=a984ff804585fabe3cd08f4b3849024a server=SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3
```

Figura 12: HASSH generado por el Cliente 3 utilizando fatt.

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

Se puede apreciar que en la captura de paquetes del **Cliente 4 / Servidor 1** se obtiene un **Key Exchange Init** con tamaño igual a **1570 bytes**, junto con el **HASSH** correspondiente a **78c05d999799066a2b4554ce7b1585a6**.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.21.0.5	172.21.0.5	TCP	74	53818 → 22 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 …
2	0.000025	172.21.0.5	172.21.0.5	TCP	74	22 → 53818 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 S…
3	0.000048	172.21.0.5	172.21.0.5	TCP	66	53818 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3669282319…
4	0.000758	172.21.0.5	172.21.0.5	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
5	0.000777	172.21.0.5	172.21.0.5	TCP	66	22 → 53818 [ACK] Seq=1 Ack=42 Win=65536 Len=0 TSval=366928231…
6	0.001791	172.21.0.5	172.21.0.5	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
7	0.001808	172.21.0.5	172.21.0.5	TCP	66	53818 → 22 [ACK] Seq=42 Ack=42 Win=65536 Len=0 TSval=36692823…
8	0.002282	172.21.0.5	172.21.0.5	SSHv2	1570	Client: Key Exchange Init
9	0.004571	172.21.0.5	172.21.0.5	SSHv2	266	Server: Key Exchange Init
10	0.005148	172.21.0.5	172.21.0.5	SSHv2	146	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.007701	172.21.0.5	172.21.0.5	SSHv2	714	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New…
12	0.049084	172.21.0.5	172.21.0.5	TCP	66	53818 → 22 [ACK] Seq=1626 Ack=890 Win=65536 Len=0 TSval=36692…
13	1.227599	172.21.0.5	172.21.0.5	SSHv2	82	Client: New Keys

Figura 13: Captura del tráfico del Cliente 4 / Servidor 1 con Key Exchange Init igual a 1570 bytes.

```
~/fatt  master ?:1 ✗
❯ python3 fatt.py -r /home/kerssen/Documentos/Cripto/lab5/trafico/trafico-c4.pcap -p
172.21.0.5:53818 -> 172.21.0.5:22 [SSH] hassh=78c05d999799066a2b4554ce7b1585a6 client=SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3
172.21.0.5:22 -> 172.21.0.5:53818 [SSH] hasshS=b8f5c09c734aeba3c765e5933e017adf server=SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3
```

Figura 14: HASSH generado por el Cliente 4 / Servidor 1 utilizando fatt.

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

Al momento de comparar los **HASSH obtenidos** con la base de datos proporcionada en la tarea, se puede observar que ningún HASSH coincide con los registrados. A continuación se mostrarán los resultados de las búsquedas:

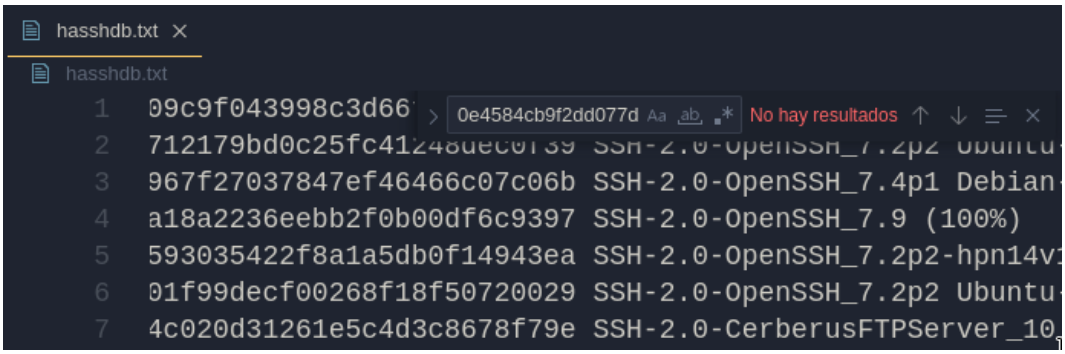


Figura 15: Búsqueda del HASSH de Cliente 1 en la base de datos.

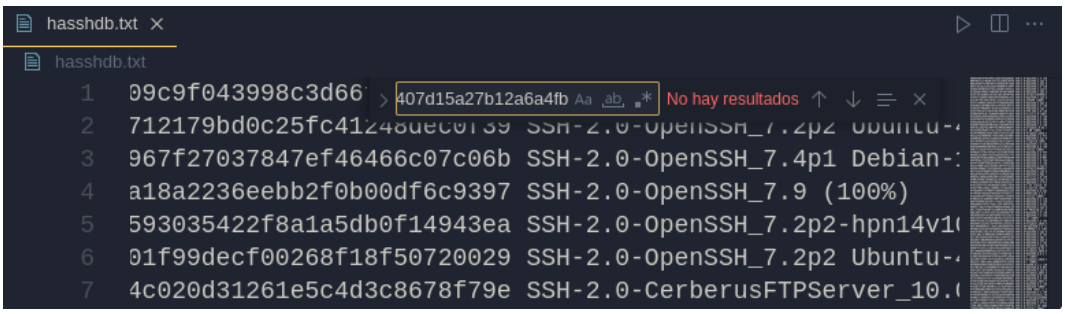


Figura 16: Búsqueda del HASSH de Cliente 2 en la base de datos.

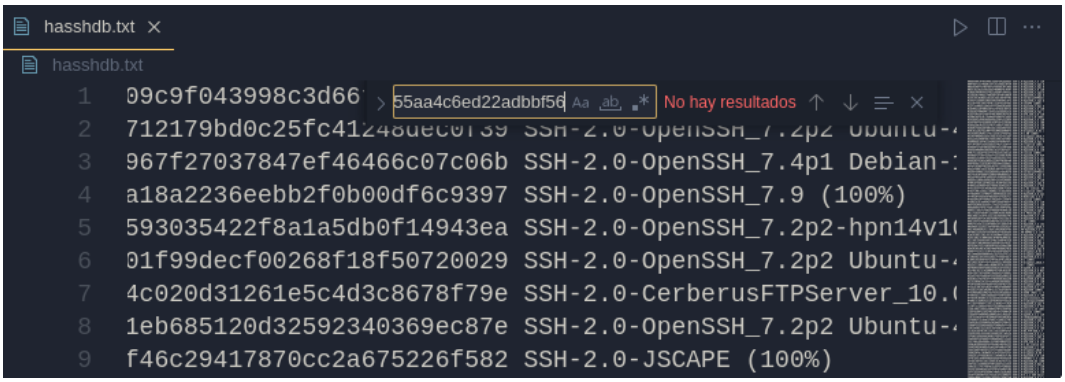


Figura 17: Búsqueda del HASSH de Cliente 3 en la base de datos.

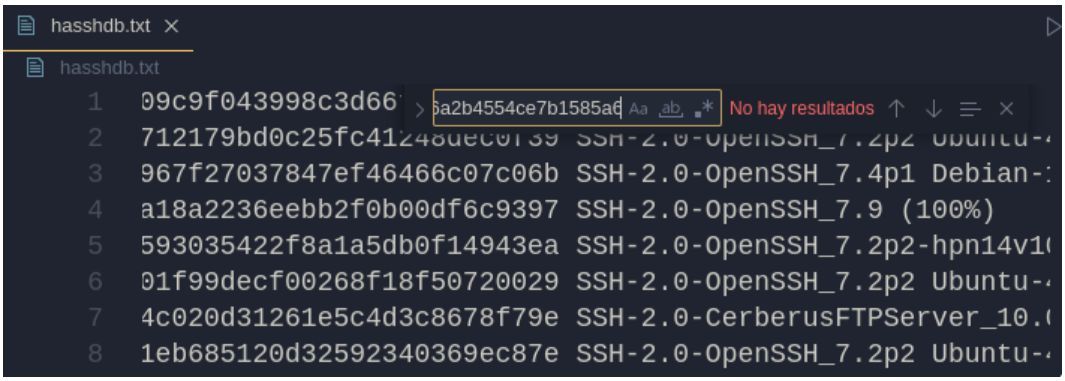


Figura 18: Búsqueda del HASSH de Cliente 4 en la base de datos.

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

1.8.1. C1

La información en texto plano que se puede ver es la siguiente:

Figura 19: Información en texto plano del Cliente 1.

- **Versiones del protocolo SSH:** SSH-2.0-OpenSSH_7.3p1 Ubuntu-1ubuntu0.1 para el cliente y SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3 para el servidor.
- **Algoritmos de intercambio de claves:** curve25519-sha256@libssh.org, ecdh-sha2-nistp256, diffie-hellman-group-exchange-sha256.
- **Algoritmos de autenticación de clave pública:** ecdsa-sha2-nistp256-cert-v01@openssh.com, ssh-ed25519, rsa-sha2-512.
- **Algoritmos de cifrado:** chacha20-poly1305@openssh.com, aes128-ctr, aes256-gcm@openssh.com, aes192-cbc, aes256-cbc, 3des-cbc.
- **Algoritmos de integridad de datos (MAC):** umac-64-etm@openssh.com, hmac-sha2-256-etm@openssh.com.
- **Algoritmos de compresión:** zlib@openssh.com.

1.8.2. C2

La información en texto plano que se puede ver es la siguiente:

Figura 20: Información en texto plano del Cliente 2.

- **Versiones del protocolo SSH:** SSH-2.0-OpenSSH_7.7p1 Ubuntu-4ubuntu0.3 para el cliente y SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3 para el servidor.
- **Algoritmos de intercambio de claves:** curve25519-sha256, curve25519-sha256@libssh.org, ecdh-sha2-nistp256, diffie-hellman-group-exchange-sha256.
- **Algoritmos de autenticación de clave pública:** ecdsa-sha2-nistp256-cert-v01@openssh.com, ssh-ed25519, rsa-sha2-512.
- **Algoritmos de cifrado:** chacha20-poly1305@openssh.com, aes128-ctr, aes256-gcm@openssh.com.
- **Algoritmos de integridad de datos (MAC):** umac-64-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com.
- **Algoritmos de compresión:** zlib@openssh.com.

1.8.3. C3

La información en texto plano que se puede ver es la siguiente:

Figura 21: Información en texto plano del Cliente 3.

- **Versiones del protocolo SSH:** SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1 para el cliente y SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3 para el servidor.
- **Algoritmos de intercambio de claves:** 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, diffie-hellman-group14-sha1, ext-info-c.
- **Algoritmos de autenticación de clave pública:** ecdsa-sha2-nistp256-cert-v01@openssh.com, ecdsa-sha2-nistp384-cert-v01@openssh.com, ecdsa-sha2-nistp521-cert-v01@openssh.com, sk-ecdsa-sha2-nistp256-cert-v01@openssh.com, ssh-ed25519-cert-v01@openssh.com, sk-ssh-ed25519-cert-v01@openssh.com, rsa-sha2-512-cert-v01@openssh.com, rsa-sha2-256-cert-v01@openssh.com, ssh-rsa-cert-v01@openssh.com, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521, sk-ecdsa-sha2-nistp256@openssh.com, ssh-ed25519, sk-ssh-ed25519@openssh.com, rsa-sha2-512, rsa-sha2-256, ssh-rsa.
- **Algoritmos de cifrado:** chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gcm@openssh.com.
- **Algoritmos de integridad de datos (MAC):** 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.
- **Algoritmos de compresión:** zlib@openssh.com, zlib.

1.8.4. C4/S1

La información en texto plano que se puede ver es la siguiente:

Figura 22: Información en texto plano del Cliente 4.

- **Versiones del protocolo SSH:** SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3 para el cliente y SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3 para el servidor.
- **Algoritmos de intercambio de claves:** 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.
- **Algoritmos de autenticación de clave pública:** ssh-ed25519-cert-v01@openssh.com, ecdsa-sha2-nistp256-cert-v01@openssh.com, ecdsa-sha2-nistp384-cert-v01@openssh.com, ecdsa-sha2-nistp521-cert-v01@openssh.com, sk-ssh-ed25519-cert-v01@openssh.com, sk-ecdsa-sha2-nistp256-cert-v01@openssh.com, rsa-sha2-512-cert-v01@openssh.com, rsa-sha2-256-cert-v01@openssh.com, ssh-ed25519, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521, sk-ssh-ed25519@openssh.com, sk-ecdsa-sha2-nistp256@openssh.com, rsa-sha2-512, rsa-sha2-256.
- **Algoritmos de cifrado:** chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gcm@openssh.com.
- **Algoritmos de integridad de datos (MAC):** 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,
- **Algoritmos de compresión:** zlib@openssh.com, zlib.

.

1.9. Diferencia entre C1 y C2

La primera diferencia es la **Versiones del software SSH**, siendo la de Cliente 2 (SSH-2.0-OpenSSH_7.7p1) una versión más actualizada que la de Cliente 1 (SSH-2.0-OpenSSH_7.3p1).

La segunda diferencia está en los **Algoritmos de intercambio de claves**, en donde Cliente 2 tiene tiene el algoritmo **curve25519-sha256** aparte de los que tiene Cliente 1.

La tercera diferencia está en los *Algoritmos de cifrado*, donde el Cliente 2 ya no utiliza **aes192-cbc, aes256-cbc, 3des-cbc**.

1.10. Diferencia entre C2 y C3

La primera diferencia es la **Versiones del software SSH**, siendo la de Cliente 3 (SSH-2.0-OpenSSH_8.3p1) una versión más actualizada que la de Cliente 1 (SSH-2.0-OpenSSH_7.7p1).

En los **Algoritmos de Intercambio de Claves** el tráfico del Cliente 3 incluye más algoritmos como **ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, diffie-hellman-group14-sha256, diffie-hellman-group14-sha1, y ext-info-c**.

En cuanto a los **Algoritmos de autenticación de clave pública** el Cliente 3 incluye una variedad mucho mayor de algoritmos de autenticación, incluyendo varios tipos de autenticación con claves de seguridad como **sk-ecdsa-sha2-nistp256-cert-v01@openssh.com, sk-ssh-ed25519-cert-v01@openssh.com, sk-ecdsa-sha2-nistp256@openssh.com, sk-ssh-ed25519@openssh.com**.

1.11. Diferencia entre C3 y C4

La primera diferencia es la **Versiones del software SSH**, siendo la de Cliente 4 (SSH-2.0-OpenSSH_9.0p1) una versión más actualizada que la de Cliente 1 (SSH-2.0-OpenSSH_8.3p1).

En los **Algoritmos de Intercambio de Claves**, el Cliente 4 usa **sntrup761x25519-sha512@openssh.com** y el Cliente 3 no.

En los **Algoritmos de autenticación de clave pública**, el Cliente 4 utiliza **sk-ssh-ed25519-cert-v01@openssh.com y sk-ecdsa-sha2-nistp256-cert-v01@openssh.com** como variante específicas, mientras que el Cliente 3 no las tiene.

En cuanto a **Algoritmos de cifrado**, el Cliente 4 agrega **aes128-gcm@openssh.com**.

2. Desarrollo (Parte 2)

2.1. Identificación del cliente SSH con versión "?"

Para identificar la versión utilizada por el **Cliente ?** se tiene que analizar detenidamente la imagen proporcionada en la tarea. Se puede ver que este cliente contiene un **Key Exchange Init** con un tamaño de **1578 bytes**, que al revisar los valores obtenidos en la primera parte del laboratorio, es el mismo que tiene el **Cliente 3** (Véase en 1.5).

Con esto, se puede concluir que la versión del Cliente ? es la de **Ubuntu 20.10** con **OpenSSH_8.3p1**.

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

Para la replicación del trafico, se utilizó el siguiente *Dockerfile*:

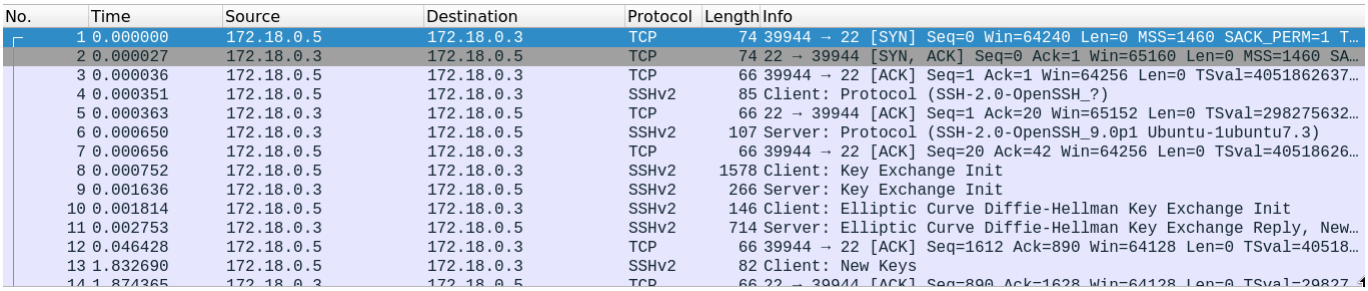
```
FROM ubuntu:20.10

RUN sed -i 's/archive.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    sed -i 's/security.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    apt-get update && apt-get upgrade

RUN apt-get install tcpdump -y

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

COPY ./openssh-8.3p1.tar.gz .

RUN tar zxvf openssh-8.3p1.tar.gz

RUN sed -i '3s/8.3/?/' /openssh-8.3p1/version.h
WORKDIR /openssh-8.3p1

RUN ./configure && \
    make && \
    make install

CMD ["tail", "-f", "/dev/null"]
```

Figura 23: Dockerfile para replicar al Cliente ?.

En donde se descargó la misma versión de *OpenSSH* que utiliza el cliente para modificarla. Se copia la versión al contenedor de *Docker* para luego descomprimirla y modificar el archivo **version.h** para cambiar la versión a traves del comando **sed –i**.

Una vez hecho lo anterior, se procede a instalar esta versión modificada de *OpenSSH* en el sistema operativo del contenedor.

A continuación, se muestra el tráfico capturado en esta versión modificada:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.18.0.5	172.18.0.3	TCP	74	39944 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…
2	0.000027	172.18.0.3	172.18.0.5	TCP	74	22 → 39944 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA…
3	0.000036	172.18.0.5	172.18.0.3	TCP	66	39944 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4051862637…
4	0.000351	172.18.0.5	172.18.0.3	SSHv2	85	Client: Protocol (SSH-2.0-OpenSSH_?)
5	0.000363	172.18.0.3	172.18.0.5	TCP	66	22 → 39944 [ACK] Seq=1 Ack=20 Win=65152 Len=0 TSval=298275632…
6	0.000650	172.18.0.3	172.18.0.5	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
7	0.000656	172.18.0.5	172.18.0.3	TCP	66	39944 → 22 [ACK] Seq=20 Ack=42 Win=64256 Len=0 TSval=40518626…
8	0.000752	172.18.0.5	172.18.0.3	SSHv2	1578	Client: Key Exchange Init
9	0.001636	172.18.0.3	172.18.0.5	SSHv2	266	Server: Key Exchange Init
10	0.001814	172.18.0.5	172.18.0.3	SSHv2	146	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.002753	172.18.0.3	172.18.0.5	SSHv2	714	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New…
12	0.046428	172.18.0.5	172.18.0.3	TCP	66	39944 → 22 [ACK] Seq=1612 Ack=890 Win=64128 Len=0 TSval=40518…
13	1.832690	172.18.0.5	172.18.0.3	SSHv2	82	Client: New Keys
14	1.874365	172.18.0.3	172.18.0.5	TCP	66	22 → 39944 [ACK] Seq=890 Ack=1628 Win=64128 Len=0 TSval=29827…

Figura 24: Tráfico generado por la versión modificada de OpenSSH.

Se puede ver en la imagen que se ha logrado cambiar la versión correctamente.

3. Desarrollo (Parte 3)

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

En la replicación del **Key Exchange Init** con un tamaño menor a 300 bytes, se modificó el contenedor que tenía corriendo el servidor de **Ubuntu 22.10**, descomentando la línea de **COPY ./sshd_config /etc/ssh/sshd_config**.

```
FROM ubuntu:22.10

RUN sed -i 's/archive.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    sed -i 's/security.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list && \
    apt-get update && apt-get install -y openssh-client openssh-server

RUN apt-get install tcpdump -y && apt-get update

## Para el paso 3
COPY ./sshd_config /etc/ssh/sshd_config

RUN useradd -m -p $(openssl passwd -1 prueba) prueba
RUN sed -i 's/#PasswordAuthentication yes/PasswordAuthentication yes/' /etc/ssh/sshd_config

CMD ["/usr/sbin/sshd", "-D", "-e"]
```

Figura 25: Dockerfile modificado para el servidor.

Se obtuvo el archivo de **sshd_config** de la carpeta en donde está instalado ssh en el sistema operativo que se utilizó para hacer esta tarea (ZorinOS) y se copió en el proyecto de la tarea.

Luego, se modifico el archivo añadiéndole las siguiente líneas de código:

```
# Ciphers and keying
#RekeyLimit default none

Ciphers aes128-ctr
HostKeyAlgorithms ecdsa-sha2-nistp256
KexAlgorithms ecdh-sha2-nistp256
MACs hmac-sha2-256
```

Figura 26: Modificación del archivo sshd_config.

Ciphers aes128-ctr especifica el algoritmo de cifrado a utilizar para cifrar datos durante la sesión SSH, siendo el cifrado AES en modo Counter (CTR) con una clave de 128 bits.

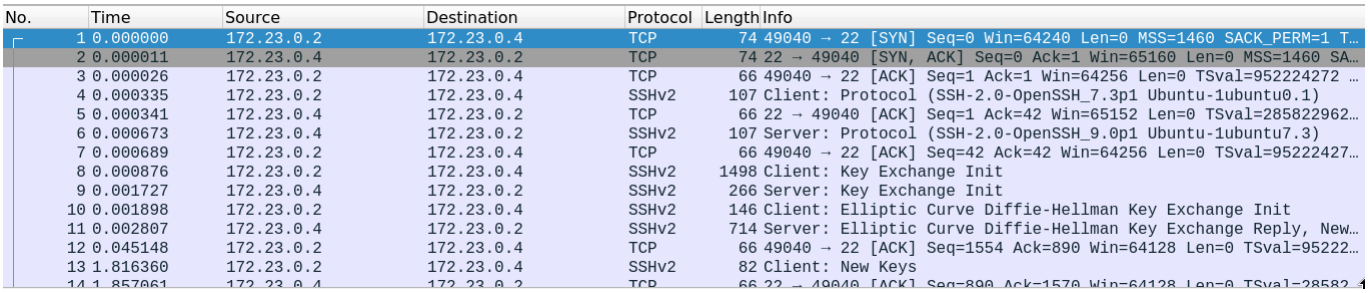
HostKeyAlgorithms ecdsa-sha2-nistp256 define el algoritmo de firma de clave del servidor SSH que se utilizará para autenticar la clave del servidor, siendo este ECDSA con la curva NIST P-256. Este algoritmo proporciona una longitud de clave más corta en comparación con RSA, logrando acelerar el proceso de intercambio de claves.

KexAlgorithms ecdh-sha2-nistp256 especifica el algoritmo de intercambio de claves, utilizando ECDH con la curva NIST P-256 para este caso. Con esto el cliente y el servidor pueden acordar una clave de sesión segura sin transmitir directamente la clave a través de la red, reduciendo el tamaño del KEI.

MACs hmac-sha2-256 especifica el algoritmo de código de autenticación de mensajes, siendo para este caso HMAC utilizando el hash SHA-256.

Una vez modificado el archivo, se copia al contenedor de *Docker* para luego reemplazar el archivo original de la versión de OpenSSH utilizada, logrando cambiar la configuración a la explicada anteriormente.

El resultado de esta modificación es el siguiente:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.23.0.2	172.23.0.4	TCP	74	49040 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…
2	0.000011	172.23.0.4	172.23.0.2	TCP	74	22 → 49040 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA…
3	0.000026	172.23.0.2	172.23.0.4	TCP	66	49040 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=952224272 …
4	0.000335	172.23.0.2	172.23.0.4	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.3p1 Ubuntu-1ubuntu0.1)
5	0.000341	172.23.0.4	172.23.0.2	TCP	66	22 → 49040 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=285822962…
6	0.000673	172.23.0.4	172.23.0.2	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-1ubuntu7.3)
7	0.000689	172.23.0.2	172.23.0.4	TCP	66	49040 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=95222427…
8	0.000876	172.23.0.2	172.23.0.4	SSHv2	1498	Client: Key Exchange Init
9	0.001727	172.23.0.4	172.23.0.2	SSHv2	266	Server: Key Exchange Init
10	0.001898	172.23.0.2	172.23.0.4	SSHv2	146	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.002807	172.23.0.4	172.23.0.2	SSHv2	714	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New…
12	0.045148	172.23.0.2	172.23.0.4	TCP	66	49040 → 22 [ACK] Seq=1554 Ack=890 Win=64128 Len=0 TSval=95222…
13	1.816360	172.23.0.2	172.23.0.4	SSHv2	82	Client: New Keys
14	1.857061	172.23.0.4	172.23.0.2	TCP	66	22 → 49040 [ACK] Seq=890 Ack=1570 Win=64128 Len=0 TSval=28582…

Figura 27: Tráfico generado por el servidor con el sshd_config modificado.

Conclusiones y comentarios

Se puede concluir que se finalizó la experiencia de laboratorio de manera exitosa debido a que se lograron cumplir los objetivos planteados del mismo, levantando los contenedores correspondientes, obteniendo los HASSH de cada versión de Ubuntu, identificar la versión del contenedor modificado replicando su trafico y obtener un KEI menor a 300 bytes en el servidor.

Se aprendió bastante sobre las diferencias de versiones de cada sistema operativo implementado en la tarea, logrando entender las mejoras de los algoritmos entre cada versión. También se aprendió mucho sobre el funcionamiento de OpenSSH y de como pueden afectar la elección de los algoritmos a la conexión.

A continuación se explicarán las problemáticas que se dieron en la elaboración del laboratorio:

1. En un inicio, no se lograban levantar los contenedores correctamente debido a que cada versión de Ubuntu tenían un repositorio en particular que ya no estaba disponible para su descarga. La solución fue utilizar el repositorio de **old-releases** que contienen los paquetes para las versiones que ya no están soportadas oficialmente.

2. Al momento de buscar los HASSH de que cada cliente, *Wireshark* no mostraba ese dato de manera implícita en la versión que se estaba utilizando. La solución fue utilizar **fatt**, que es una herramienta especializada en la identificación y análisis de atributos de protocolos de red, enfocada en la identificación de huellas digitales de aplicaciones de red.

3. Cuando se quiso capturar el tráfico del Cliente 4 (Servidor), *tcpdump* no capturaba ningún paquete tras establecer la sesión en SSH. Esto fue debido a qué no se estaba capturando los paquetes en la red de loopback, así que la solución fue cambiar el comando de *tcpdump* para poder capturar el tráfico de manera correcta.

4. Por último, se dificulto mucho la última parte del laboratorio en la que se tenía que reducir el tamaño del KEI a uno menor a 300 bytes. Se utilizaron varios métodos para intentar bajar el tamaño, pero ninguno tenía efecto en el contenedor. La solución vino al analizar el propio funcionamiento de OpenSSH que estaba en el sistema operativo usado para realizar la tarea, es aquí en donde surgió la idea de copiar el archivo de configuración y modificarlo para reducir el tamaño del KEI.

Repositorio de Github: `https://github.com/KerssenB/Lab5_cripto_KB`