
```
basicstyle=, keywordstyle=, commentstyle=, stringstyle=, numbers=left, numberstyle=,
stepnumber=1, breaklines=true, frame=single, backgroundColor=, tabsize=4, showstringspaces=false
```

Informe Laboratorio 5

Sección 1

Diego Pastroán
e-mail: diego.pastrian@mail.udp.cl

Noviembre de 2024

Índice

Descripción de actividades	4
1. Desarrollo (Parte 1)	7
1.1. Códigos de cada Dockerfile	7
1.1.1. C1	7
1.1.2. C2	8
1.1.3. C3	9
1.1.4. C4/S1	10
1.2. Creación de las credenciales para S1	11
1.3. Tráfico generado por C1, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)	11
1.4. Tráfico generado por C2, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)	13
1.5. Tráfico generado por C3, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)	14
1.6. Tráfico generado por C4 (iface lo), detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)	15
1.7. Compara la versión de HASSH obtenida con la base de datos para validar si el cliente corresponde al mismo	17
1.8. Tipo de información contenida en cada uno de los paquetes generados en texto plano	17
1.8.1. C1	17
1.8.2. C2	17
1.8.3. C3	18
1.8.4. C4/S1	18
1.9. Diferencia entre C1 y C2	19
1.10. Diferencia entre C2 y C3	19
1.11. Diferencia entre C3 y C4	19
2. Desarrollo (Parte 2)	20
2.1. Identificación del cliente SSH con versión “?”	20
2.2. Replicación de tráfico al servidor (paso por paso)	20
3. Desarrollo (Parte 3)	23
3.1. Replicación del KEI con tamaño menor a 300 bytes (paso por paso)	23
4. Desarrollo (Parte 4)	25
4.1. Explicación OpenSSH en general	25
4.2. Capas de Seguridad en OpenSSH	25
4.3. Identificación de que protocolos no se cumplen	26

Descripción de actividades

Para este último laboratorio, nuestro informante ya sabe que puede establecer un medio seguro sin un intercambio previo de una contraseña, gracias al protocolo diffie-hellman. El problema es que ahora no sabe si confiar en el equipo con el cual establezca comunicación, ya que las credenciales de usuario pueden haber sido divulgadas por algún soplón.

Para el presente laboratorio deberá:

- Crear 4 contenedores en Docker o Podman, donde cada uno tendrá el siguiente SO: Ubuntu 16.10, Ubuntu 18.10, Ubuntu 20.10 y Ubuntu 22.10 a los cuales se llamarán C1, C2, C3 y C4 respectivamente.
El equipo con Ubuntu 22.10 también será utilizado como S1.
- Para cada uno de ellos, deberá instalar el cliente openSSH disponible en los repositorios de apt, y para el equipo S1 deberá también instalar el servidor openSSH.
- En S1 deberá crear el usuario “**prueba**” con contraseña “**prueba**”, para acceder a él desde los clientes por el protocolo SSH.
- En total serán 4 escenarios, donde cada uno corresponderá a los siguientes equipos:
 - C1 → S1
 - C2 → S1
 - C3 → S1
 - C4 → S1

Pasos:

1. Para cada uno de los 4 escenarios, deberá capturar el tráfico generado por cada conexión con el server. A partir de cada handshake, deberá analizar el patrón de tráfico generado por cada cliente y adicionalmente obtener el HASSH que lo identifique. De esta forma podrá obtener una huella digital para cada cliente a partir de su tráfico. Cada HASSH deberá compararlo con la base de datos HASSH disponible en el módulo de TLS, e identificar si el hash obtenido corresponde a la misma versión de su cliente.

Indique el tamaño de los paquetes del flujo generados por el cliente y el contenido asociado a cada uno de ellos. Indique qué información distinta contiene el escenario siguiente (diff incremental). El objetivo de este paso es identificar claramente los cambios entre las distintas versiones de ssh.

2. Para poder identificar que el usuario efectivamente es el informante, éste utilizará una versión única de cliente. ¿Con qué cliente SSH se habrá generado el siguiente tráfico?

Protocol	Length	Info
TCP	74	34328 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=14
TCP	66	34328 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0
SSHv2	85	Client: Protocol (SSH-2.0-OpenSSH_?)
TCP	66	34328 → 22 [ACK] Seq=20 Ack=42 Win=64256 Len=
SSHv2	1578	Client: Key Exchange Init
TCP	66	34328 → 22 [ACK] Seq=1532 Ack=1122 Win=64128
SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exc
TCP	66	34328 → 22 [ACK] Seq=1580 Ack=1574 Win=64128
SSHv2	82	Client: New Keys
SSHv2	110	Client: Encrypted packet (len=44)
TCP	66	34328 → 22 [ACK] Seq=1640 Ack=1618 Win=64128
SSHv2	126	Client: Encrypted packet (len=60)
TCP	66	34328 → 22 [ACK] Seq=1700 Ack=1670 Win=64128
SSHv2	150	Client: Encrypted packet (len=84)
TCP	66	34328 → 22 [ACK] Seq=1784 Ack=1698 Win=64128
SSHv2	178	Client: Encrypted packet (len=112)
TCP	66	34328 → 22 [ACK] Seq=1896 Ack=2198 Win=64128

Figura 1: Tráfico generado del informante

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

3. Para que el informante esté seguro de nuestra identidad, nos pide que el patrón del tráfico de nuestro server también sea modificado, hasta que el Key Exchange Init del server sea menor a 300 bytes. Indique qué pasos realizó para lograr esto.

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-C			
TCP	66	22 → 42398	[ACK]	Seq=1	Ack=
SSHv2	107	Server: Protocol (SSH-2.0-C			
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	A

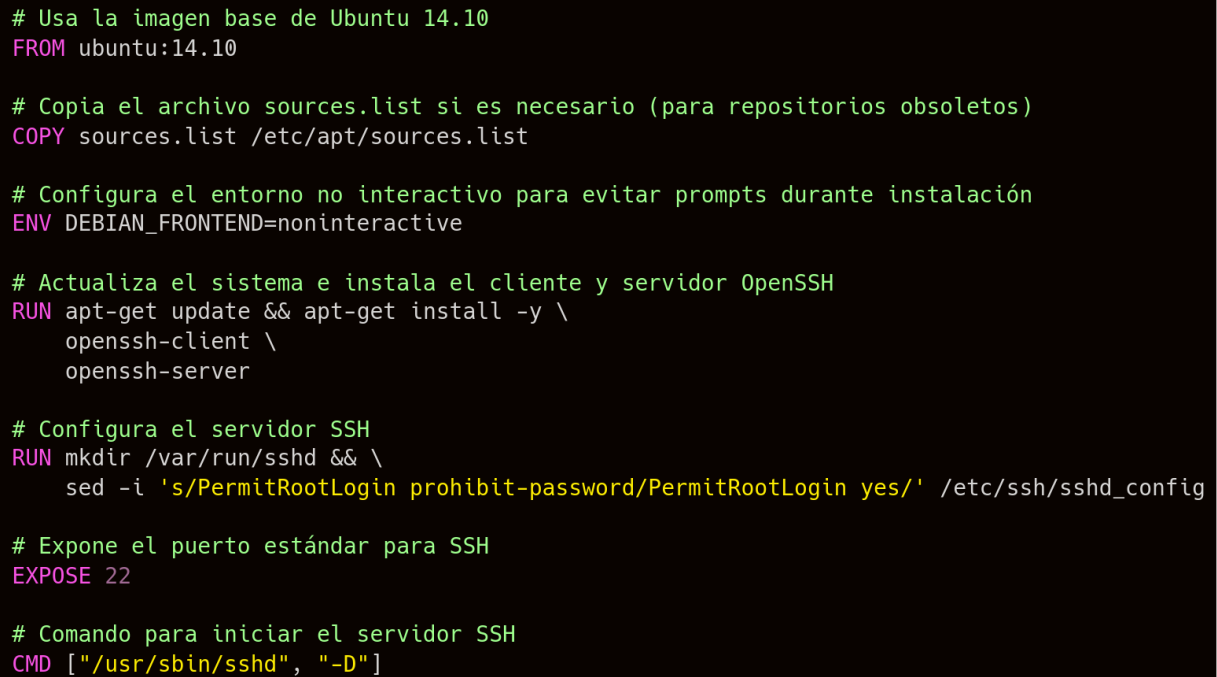
Figura 2: Captura del Key Exchange

- 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

1.1.1. C1

A screenshot of a terminal window with a dark background and light green text. The terminal shows a series of Dockerfile instructions for building a container named C1. The instructions are color-coded: comments are light green, FROM is pink, COPY is pink, ENV is pink, RUN is pink, and EXPOSE is pink. The instructions include setting the base image to Ubuntu 14.10, copying the sources.list file, setting the DEBIAN_FRONTEND to noninteractive, updating the system and installing openssh-client and openssh-server, configuring the SSH server to allow root login without a password, exposing port 22, and setting the command to start the SSH server.

```
# Usa la imagen base de Ubuntu 14.10
FROM ubuntu:14.10

# Copia el archivo sources.list si es necesario (para repositorios obsoletos)
COPY sources.list /etc/apt/sources.list

# Configura el entorno no interactivo para evitar prompts durante instalación
ENV DEBIAN_FRONTEND=noninteractive

# Actualiza el sistema e instala el cliente y servidor OpenSSH
RUN apt-get update && apt-get install -y \
    openssh-client \
    openssh-server

# Configura el servidor SSH
RUN mkdir /var/run/sshd && \
    sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config

# Expone el puerto estándar para SSH
EXPOSE 22

# Comando para iniciar el servidor SSH
CMD ["/usr/sbin/sshd", "-D"]
```

Figura 3: Código Docker para C1.

1.1.2. C2

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays a Dockerfile script for building a container named C2. The script includes comments in Spanish and instructions for using the Ubuntu 16.10 base image, copying sources.list, updating the system, installing OpenSSH, configuring the SSH server to allow root login, exposing port 22, and running the sshd service in the background.

```
# Usa la imagen base de Ubuntu 16.10
FROM ubuntu:16.10

# Copia el archivo sources.list si es necesario
COPY sources.list /etc/apt/sources.list

# Actualiza el sistema e instala el cliente y servidor OpenSSH
RUN apt-get update && apt-get install -y \
    openssh-client \
    openssh-server


# Configura el servidor SSH
RUN mkdir /var/run/sshd && \
    sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config

# Expone el puerto estándar para SSH
EXPOSE 22

# Comando para iniciar el servidor SSH
CMD ["/usr/sbin/sshd", "-D"]
```

Figura 4: Código Docker para C2.

1.1.3. C3

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays a Dockerfile script for building a container named C3. The script includes comments in Spanish and Dockerfile instructions: FROM, COPY, RUN, EXPOSE, and CMD. The instructions set the base image to Ubuntu 18.10, copy the sources.list file, update the system and install OpenSSH, configure the SSH server to allow root login, expose port 22, and set the command to start the SSH daemon in the background.

```
# Usa la imagen base de Ubuntu 18.10
FROM ubuntu:18.10

# Copia el archivo sources.list si es necesario
COPY sources.list /etc/apt/sources.list

# Actualiza el sistema e instala el cliente y servidor OpenSSH
RUN apt-get update && apt-get install -y \
    openssh-client \
    openssh-server

# Configura el servidor SSH
RUN mkdir /var/run/sshd && \
    sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config

# Expone el puerto estándar para SSH
EXPOSE 22

# Comando para iniciar el servidor SSH
CMD ["/usr/sbin/sshd", "-D"]
```

Figura 5: Código Docker para C3

1.1.4. C4/S1

```
# Usa la imagen base de Ubuntu 20.10
FROM ubuntu:20.10

# Copia el archivo sources.list si es necesario
COPY sources.list /etc/apt/sources.list

# Actualiza el sistema e instala el cliente y servidor OpenSSH
RUN apt-get update && apt-get install -y \
    tshark \
    openssh-client \
    openssh-server \
    whois

# Crea el usuario 'test' con contraseña 'test'
RUN useradd -m test && \
    echo "test:test" | chpasswd

# Configura el servidor SSH
RUN mkdir /var/run/sshd && \
    sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config

# Expone el puerto estándar para SSH
EXPOSE 22

# Comando para iniciar el servidor SSH
CMD ["/usr/sbin/sshd", "-D"]
```

Figura 6: Código Docker para C3/S1.

Finalmente, los DockerFiles se construyen mediante los comandos:

```
diegoo@diegoo:~/Escritorio/universidad/criptografia/Lab-5-Criptografia/Lab5_cripto$
docker build -t c1 -f ./c1/c1.dockerfile ./c1
docker build -t c2 -f ./c2/c2.dockerfile ./c2
docker build -t c3 -f ./c3/c3.dockerfile ./c3
docker build -t c4s1 -f ./c4s1/c4s1.dockerfile ./c4s1
```

Figura 7: Creación de las imágenes Docker.

Y las imágenes generadas se ejecutan con los siguientes comandos.

En C4S1 se agregan tales parámetros para las capturas posteriores con **tshark**.

```
diegoo@diegoo:~/local/share/Trash/files/Lab-5-Criptografia/Lab5_cripto$  
sudo docker run -it c1 bashsh  
sudo docker run -it c2 bash  
sudo docker run -it c2 bash  
sudo docker run --cap-add=NET_RAW --cap-add=NET_ADMIN -it c4s1 bash
```

Figura 8: Correr imágenes Docker.

```
root@ab4a... x root@22e0... x root@bd62... x root@5e55... x v  
root@5e55a711badb:/# service ssh start  
* Starting OpenBSD Secure Shell server sshd [ OK ]  
root@5e55a711badb:/# service ssh status  
* sshd is running  
root@5e55a711badb:/#
```

Figura 9: Iniciar servicio SSH.

1.2. Creación de las credenciales para S1

Las credenciales se crean dentro del Dockerfile de C4/S1 con el comando *useradd*. Estas se definen como:

- Usuario: 'test'
- Contraseña: 'test'

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

Una vez levantados los contenedores, se inicia el servicio SSH dentro del contenedor C4 para que el servidor funcione correctamente. Este proceso se detalla a continuación.

Después, se registra la IP del servidor para realizar las conexiones, esta se obtiene mediante el comando *ifconfig*, obteniendo así la ip **172.17.0.5**. Finalmente, en el contenedor del cliente, se realiza la conexión al servidor usando el comando *ssh test@172.17.0.5*.

Para la conexión se debe tener en cuenta las credenciales mencionadas anteriormente y la ip.

- IP: 172.17.0.5
- Usuario: 'test'
- Clave: 'test'

Así, se realiza la conexión para C1.

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

1 DESARROLLO (PARTE 1)

```
test@5e55... x root@22e0... x root@bd62... x root@5e55... x v
root@ab4a5f79ae6d:/# ssh test@172.17.0.5
The authenticity of host '172.17.0.5 (172.17.0.5)' can't be established.
ECDSA key fingerprint is 27:78:d3:d3:3c:f0:a7:16:c1:51:90:0f:03:ec:75:76.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.17.0.5' (ECDSA) to the list of known host
s.
test@172.17.0.5's password:
Welcome to Ubuntu 20.10 (GNU/Linux 6.8.0-49-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.
```

Figura 10: Conectando C1 con S1

A continuación se presenta la captura del tráfico cliente-servidor con wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
26	256.534138145	172.17.0.5	172.17.0.2	TCP	74	22 → 46748 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3055388598 TSecr=3224263971
29	256.534184249	172.17.0.2	172.17.0.5	TCP	66	46748 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3224263969 TSecr=3055388598
30	256.536707314	172.17.0.2	172.17.0.5	SSHv2	100	Client: Protocol (SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-8)
31	256.536707950	172.17.0.5	172.17.0.2	TCP	66	22 → 46748 [ACK] Seq=1 Ack=35 Win=65152 Len=0 TSval=3055388600 TSecr=3224263971
32	256.571279514	172.17.0.5	172.17.0.2	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
33	256.571342517	172.17.0.2	172.17.0.5	TCP	66	46748 → 22 [ACK] Seq=35 Ack=42 Win=64256 Len=0 TSval=3224264006 TSecr=3055388635
34	256.572150677	172.17.0.2	172.17.0.5	SSHv2	2034	Client: Key Exchange Init
35	256.57522712	172.17.0.5	172.17.0.2	SSHv2	1122	Server: Key Exchange Init
36	256.584296509	172.17.0.2	172.17.0.5	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
37	256.599889172	172.17.0.5	172.17.0.2	SSHv2	346	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
38	256.645207678	172.17.0.2	172.17.0.5	TCP	66	46748 → 22 [ACK] Seq=2051 Ack=1378 Win=64128 Len=0 TSval=3224264080 TSecr=3055388663
39	260.626060960	172.17.0.2	172.17.0.5	SSHv2	82	Client: New Keys
40	260.673185533	172.17.0.5	172.17.0.2	TCP	66	22 → 46748 [ACK] Seq=1378 Ack=2067 Win=64128 Len=0 TSval=3055392737 TSecr=3224268060
41	260.673235649	172.17.0.2	172.17.0.5	SSHv2	122	Client: Encrypted packet (len=56)
42	260.673274337	172.17.0.5	172.17.0.2	TCP	66	22 → 46748 [ACK] Seq=1378 Ack=2123 Win=64128 Len=0 TSval=3055392737 TSecr=3224268108
43	260.673405257	172.17.0.2	172.17.0.5	SSHv2	122	Server: Encrypted packet (len=56)
44	260.673444028	172.17.0.2	172.17.0.5	TCP	66	46748 → 22 [ACK] Seq=2123 Ack=1434 Win=64128 Len=0 TSval=3224268108 TSecr=3055392737
45	260.673612477	172.17.0.2	172.17.0.5	SSHv2	138	Client: Encrypted packet (len=72)
46	260.681580755	172.17.0.5	172.17.0.2	SSHv2	122	Server: Encrypted packet (len=56)
47	260.725254896	172.17.0.2	172.17.0.5	TCP	66	46748 → 22 [ACK] Seq=2195 Ack=1490 Win=64128 Len=0 TSval=3224268160 TSecr=3055392745
48	260.485620755	172.17.0.2	172.17.0.5	SSHv2	218	Client: Encrypted packet (len=152)
49	260.517604684	172.17.0.5	172.17.0.2	SSHv2	106	Server: Encrypted packet (len=40)
50	260.517673156	172.17.0.2	172.17.0.5	TCP	66	46748 → 22 [ACK] Seq=2347 Ack=1530 Win=64128 Len=0 TSval=3224275952 TSecr=3055400581
51	260.518146098	172.17.0.2	172.17.0.5	SSHv2	194	Client: Encrypted packet (len=128)
52	260.557908120	172.17.0.5	172.17.0.2	SSHv2	698	Server: Encrypted packet (len=632)
53	260.601266110	172.17.0.2	172.17.0.5	TCP	66	46748 → 22 [ACK] Seq=2475 Ack=2162 Win=64128 Len=0 TSval=3224276036 TSecr=3055400621
54	260.601317158	172.17.0.5	172.17.0.2	SSHv2	122	Server: Encrypted packet (len=56)
55	260.601350444	172.17.0.2	172.17.0.5	TCP	66	46748 → 22 [ACK] Seq=2475 Ack=2218 Win=64128 Len=0 TSval=3224276036 TSecr=3055400665
56	260.601644526	172.17.0.2	172.17.0.5	SSHv2	466	Client: Encrypted packet (len=400)
57	260.61868795	172.17.0.5	172.17.0.2	SSHv2	186	Server: Encrypted packet (len=120)
58	260.619493785	172.17.0.5	172.17.0.2	SSHv2	794	Server: Encrypted packet (len=728)
59	260.630636324	172.17.0.2	172.17.0.5	TCP	66	46748 → 22 [ACK] Seq=2875 Ack=3066 Win=64128 Len=0 TSval=3224276054 TSecr=3055400682
Frame 30: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface docker0, id 0						
Ethernet II, Src: 02:42:ac:11:00:02 (02:42:ac:11:00:02), Dst: 02:42:ac:11:00:05 (02:42:ac:11:00:05)						
Internet Protocol Version 4, Src: 172.17.0.2, Dst: 172.17.0.5						
Transmission Control Protocol, Src Port: 46748, Dst Port: 22, Seq: 1, Ack: 1, Len: 34						
SSH Protocol						
Protocol: SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-8						
[Direction: client-to-server]						
0000	02 42 ac 11 00 05 02 42	ac 11 00 02 08 00	45 00	.B....B.....E.		
0010	00 56 48 f1 40 00 40 06	99 87 ac 11 00 02 ac 11	..VH.@.....			
0020	00 05 b6 9c 00 16 5a 68	e2 cf 72 a6 73 11 00 18Zh...r:s...			
0030	01 f6 58 72 00 00 01 01	00 0a c0 2e 5d 23 b6 1d	..Xr.....]m...			
0040	87 b6 53 53 48 2d 32 2e	30 2d 4f 70 65 0e 53 53	..SSH-2.0-OpenSS			
0050	48 5f 36 2e 36 2e 31 70	31 20 55 62 75 6e 74 75	H.6.6.1p 1 Ubuntu			
0060	2d 38 0d 0a		-8..			

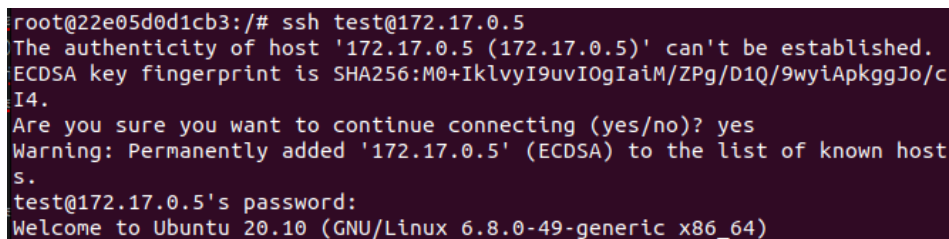
Figura 11: Tráfico generado por la conexión SSH entre C1 y S1.

Durante la conexión entre C1 y el servidor SSH S1, se observa el uso del protocolo SSH versión 2.0. Esta implementación es OpenSSH_6.6.1p1 para Ubuntu-8 y en el server es OpenSSH_9.3p1 Ubuntu-1ubuntu0.1. Esto se debe a las distintas versiones de Ubuntu

instaladas en cada contenedor. La ip del server es 172.17.0.5 y la del cliente 172.17.0.2. Los tamaños de los paquetes varían notablemente, especialmente los paquetes número 34(2034 bytes) y 35(1122 bytes) los que corresponden al inicio de intercambio de claves. Esto sigue el método Diffie-Hellman, después del cual se generan nuevas claves para la sesión. Los paquetes más grandes se deben a la negociación de los algoritmos de cifrado disponibles, paso crucial para poder establecer un canal seguro. Los siguientes paquetes están encriptados, garantizando integridad y confidencialidad de la sesión SSH.

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

Se realiza la conexión SSH para C2 con S1.

A terminal window with a dark background and light-colored text. The text shows the execution of an SSH command from a host named 'root@22e05d0d1cb3' to a host named 'test@172.17.0.5'. The terminal output includes a warning about the authenticity of the host, the ECDSA key fingerprint, a confirmation to continue connecting, a warning about adding the host to the known hosts list, a password prompt, and a final welcome message to Ubuntu 20.10.

```
root@22e05d0d1cb3:/# ssh test@172.17.0.5
The authenticity of host '172.17.0.5 (172.17.0.5)' can't be established.
ECDSA key fingerprint is SHA256:M0+IklvyI9uvIOgIaiM/ZPg/D1Q/9wyiApgggJo/c
I4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.17.0.5' (ECDSA) to the list of known host
s.
test@172.17.0.5's password:
Welcome to Ubuntu 20.10 (GNU/Linux 6.8.0-49-generic x86_64)
```

Figura 12: Conexión SSH entre C2 y S1.

La captura de paquetes se presenta a continuación.

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
7	17.926982872	172.17.0.3	172.17.0.2	TCP	66	37940 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1468404181 TSecr=4014931786
8	17.928362888	172.17.0.3	172.17.0.2	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.3p1 Ubuntu-1ubuntu0.1)
9	17.928415706	172.17.0.2	172.17.0.3	TCP	66	22 → 37940 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=4014931788 TSecr=1468404183
10	17.956388186	172.17.0.2	172.17.0.3	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
11	17.956442337	172.17.0.3	172.17.0.2	TCP	66	37940 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=1468404211 TSecr=4014931816
12	17.957119478	172.17.0.3	172.17.0.2	SSHv2	1498	Client: Key Exchange Init
13	17.969987879	172.17.0.2	172.17.0.3	SSHv2	1122	Server: Key Exchange Init
14	17.966327170	172.17.0.3	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
15	17.989919421	172.17.0.2	172.17.0.3	SSHv2	574	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=228)
16	18.023941661	172.17.0.3	172.17.0.2	TCP	66	37940 → 22 [ACK] Seq=1522 Ack=1606 Win=64128 Len=0 TSval=1468404279 TSecr=4014931840
17	19.684850714	172.17.0.3	172.17.0.2	SSHv2	82	Client: New Keys
18	19.727948167	172.17.0.2	172.17.0.3	TCP	66	22 → 37940 [ACK] Seq=1606 Ack=1538 Win=64128 Len=0 TSval=4014933588 TSecr=1468405939
19	19.728093679	172.17.0.3	172.17.0.2	SSHv2	110	Client: Encrypted packet (len=44)
20	19.728035834	172.17.0.2	172.17.0.3	TCP	66	22 → 37940 [ACK] Seq=1606 Ack=1582 Win=64128 Len=0 TSval=4014933588 TSecr=1468405983
21	19.728222460	172.17.0.2	172.17.0.3	SSHv2	110	Server: Encrypted packet (len=44)
22	19.728264036	172.17.0.3	172.17.0.2	TCP	66	37940 → 22 [ACK] Seq=1582 Ack=1650 Win=64128 Len=0 TSval=1468405983 TSecr=4014933588
23	19.728433513	172.17.0.3	172.17.0.2	SSHv2	126	Client: Encrypted packet (len=60)
24	19.736463977	172.17.0.2	172.17.0.3	SSHv2	118	Server: Encrypted packet (len=52)
25	19.779909522	172.17.0.3	172.17.0.2	TCP	66	37940 → 22 [ACK] Seq=1642 Ack=1702 Win=64128 Len=0 TSval=1468406035 TSecr=4014933596
26	22.536424147	172.17.0.3	172.17.0.2	SSHv2	150	Client: Encrypted packet (len=84)
27	22.568263388	172.17.0.2	172.17.0.3	SSHv2	94	Server: Encrypted packet (len=28)
28	22.568327346	172.17.0.3	172.17.0.2	TCP	66	37940 → 22 [ACK] Seq=1726 Ack=1738 Win=64128 Len=0 TSval=1468408823 TSecr=4014936428
29	22.568638711	172.17.0.3	172.17.0.2	SSHv2	178	Client: Encrypted packet (len=112)
30	22.608138390	172.17.0.2	172.17.0.3	SSHv2	694	Server: Encrypted packet (len=628)
31	22.651943293	172.17.0.3	172.17.0.2	TCP	66	37940 → 22 [ACK] Seq=1838 Ack=2358 Win=64128 Len=0 TSval=1468408907 TSecr=4014936468
32	22.651999273	172.17.0.2	172.17.0.3	SSHv2	110	Server: Encrypted packet (len=44)
33	22.652044879	172.17.0.3	172.17.0.2	TCP	66	37940 → 22 [ACK] Seq=1838 Ack=2402 Win=64128 Len=0 TSval=1468408907 TSecr=4014936512

▶ Frame 8: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface docker0, id 0
 ▶ Ethernet II, Src: 02:42:ac:11:00:03 (02:42:ac:11:00:03), Dst: 02:42:ac:11:00:02 (02:42:ac:11:00:02)
 ▶ Internet Protocol Version 4, Src: 172.17.0.3, Dst: 172.17.0.2
 ▶ Transmission Control Protocol, Src Port: 37940, Dst Port: 22, Seq: 1, Ack: 1, Len: 41
 ▶ SSH Protocol
 Protocol: SSH-2.0-OpenSSH_7.3p1 Ubuntu-1ubuntu0.1
 [Direction: client-to-server]

0000	02 42 ac 11 00 02 02 42	ac 11 00 03 08 00 45 00	-B- - - -B- - - - -E-
0010	00 5d b9 97 40 00 40 06	28 dc ac 11 00 03 ac 11	-]- @-@- (- - - - -
0020	00 02 94 34 00 16 7a 48	8f fc 0e d9 c6 06 80 18	- - -4- - -zH- - - - -
0030	01 f6 58 77 00 00 01 01	08 0a 57 86 11 d7 ef 4e	- -Xw- - - -W- - - -N
0040	ff 4a 53 53 48 2d 32 2e	30 2d 4f 70 65 66 53 53	-JSSH-2. 0-OpenSS
0050	48 5f 37 2e 33 70 31 20	55 62 75 6e 74 75 2d 31	H_7.3p1 Ubuntu-1
0060	75 62 75 6e 74 75 30 2e	31 0d 0a	ubuntu0. 1:-

Figura 13: Tráfico generado en conexión SSH entre C2 y S1.

La ip para C2 corresponde a 172.17.0.3. La del servidor es 172.17.0.2. Este cambio se debe a que se reasignan las direcciones ip en un reinicio de los contenedores. Aquí se utiliza nuevamente la versión 2 del protocolo SSH. Para el cliente, la implementación es OpenSSH_7.3P1 Ubuntu-1ubuntu0.1, mientras que el servidor ocupa la misma. En los paquetes 12(1498 bytes) y 13(1122 bytes) se observa el intercambio de claves. Se observa que el tamaño del paquete Key Exchange Init del server es del mismo tamaño que para la sesión con C1, esto se debe a que el servidor no varía. Se realiza el intercambio de llaves mediante Diffie-Hellman y los paquetes posteriores están encriptados.

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

Se realiza la conexión SSH para el C3.

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

1 DESARROLLO (PARTE 1)

```
root@b6d2253f0f79:/# ssh test@172.17.0.5
The authenticity of host '172.17.0.5 (172.17.0.5)' can't be established.
ECDSA key fingerprint is SHA256:M0+IklvyI9uvIOgIaiM/ZPg/D1Q/9wyiApgkgJo/c
I4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.17.0.5' (ECDSA) to the list of known hosts.
test@172.17.0.5's password:
Welcome to Ubuntu 20.10 (GNU/Linux 6.8.0-49-generic x86_64)
```

Figura 14: Conexión SSH C3-S1.

A continuación se realiza la captura de paquetes.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.001457457	172.17.0.3	172.17.0.2	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.7p1 Ubuntu-4ubuntu0.3)
5	0.001519397	172.17.0.2	172.17.0.3	TCP	66	22 → 39732 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=4015007344 TSecr=1468479739
6	0.032579242	172.17.0.2	172.17.0.3	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
7	0.032663218	172.17.0.3	172.17.0.2	TCP	66	39732 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=1468479770 TSecr=4015007375
8	0.034408394	172.17.0.3	172.17.0.2	SSHv2	1426	Client: Key Exchange Init
9	0.036820845	172.17.0.2	172.17.0.3	SSHv2	1122	Server: Key Exchange Init
10	0.050227744	172.17.0.3	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.068217591	172.17.0.2	172.17.0.3	SSHv2	574	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=228)
12	0.077843953	172.17.0.3	172.17.0.2	SSHv2	82	Client: New Keys
13	0.121323452	172.17.0.2	172.17.0.3	TCP	66	22 → 39732 [ACK] Seq=1606 Ack=1466 Win=64128 Len=0 TSval=4015007464 TSecr=1468479815
14	0.121393953	172.17.0.3	172.17.0.2	SSHv2	110	Client: Encrypted packet (len=44)
15	0.121428671	172.17.0.2	172.17.0.3	TCP	66	22 → 39732 [ACK] Seq=1606 Ack=1510 Win=64128 Len=0 TSval=4015007464 TSecr=1468479859
16	0.121568184	172.17.0.2	172.17.0.3	SSHv2	110	Server: Encrypted packet (len=44)
17	0.121948969	172.17.0.3	172.17.0.2	SSHv2	126	Client: Encrypted packet (len=60)
18	0.129982288	172.17.0.2	172.17.0.3	SSHv2	118	Server: Encrypted packet (len=52)
19	0.17315928	172.17.0.3	172.17.0.2	TCP	66	39732 → 22 [ACK] Seq=1578 Ack=1702 Win=64128 Len=0 TSval=1468479911 TSecr=4015007472
20	0.234510436	172.17.0.3	172.17.0.2	SSHv2	150	Client: Encrypted packet (len=84)
21	0.266332673	172.17.0.2	172.17.0.3	SSHv2	94	Server: Encrypted packet (len=28)
22	0.266402232	172.17.0.3	172.17.0.2	TCP	66	39732 → 22 [ACK] Seq=1654 Ack=1730 Win=64128 Len=0 TSval=1468483004 TSecr=4015010609
23	0.266709969	172.17.0.3	172.17.0.2	SSHv2	178	Client: Encrypted packet (len=112)
24	0.307012051	172.17.0.2	172.17.0.3	SSHv2	694	Server: Encrypted packet (len=628)
25	0.349383692	172.17.0.3	172.17.0.2	TCP	66	39732 → 22 [ACK] Seq=1766 Ack=2358 Win=64128 Len=0 TSval=1468483087 TSecr=4015010649
26	0.349435753	172.17.0.2	172.17.0.3	SSHv2	110	Server: Encrypted packet (len=44)

Frame 4: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface docker0, id 0
Ethernet II, Src: 02:42:ac:11:00:03 (02:42:ac:11:00:03), Dst: 02:42:ac:11:00:02 (02:42:ac:11:00:02)
Internet Protocol Version 4, Src: 172.17.0.3, Dst: 172.17.0.2
Transmission Control Protocol, Src Port: 39732, Dst Port: 22, Seq: 1, Ack: 1, Len: 41
SSH Protocol
Protocol: SSH-2.0-OpenSSH_7.7p1 Ubuntu-4ubuntu0.3
[direction: client-to-server]

0010	00 5d fb f7 40 00 40 06 e6 7b ac 11 00 03 ac 11].@.-{.....
0020	00 02 9b 34 00 16 b3 cd a0 fe b3 2c b0 58 08 18	...4....X.
0030	01 fe 58 77 00 00 01 01 08 0a 57 87 38 fb ef 50	..w..8.P
0040	26 6e 05 33 40 20 32 2c 30 23 07 00 05 00 53 00	anSSH-2.0-OpenSSH
0050	48 5f 37 2e 37 70 31 20 55 62 75 6e 74 75 2d 34	H.7.7p1 Ubuntu-4
0060	75 62 75 6e 74 75 30 2e 33 0d 0a	ubuntu0.3..

Figura 15: Tráfico generado en conexión SSH entre C3 y S1.

El cliente utiliza protocolo SSH versión 2.0 con implementación OpenSSH_7.7p1 Ubuntu-4ubuntu6.3, la implementación del servidor no varía. En la sesión se realiza el intercambio de claves en los paquetes 8 (1426 bytes) y 9 (1122 bytes). El tamaño de paquete asociado al servidor no varía. Se efectúa el intercambio de claves y el resto de paquetes se presentan encriptados.

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

Para lograr capturar el tráfico, se inicializa el servidor SSH de manera estándar. En otra terminal del mismo contenedor se utiliza el comando *sudo tshark -i lo -w /tmp/trafico.pcapng*. Luego se establece la conexión SSH en la terminal de C4 (que es la misma que S1).

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

1 DESARROLLO (PARTE 1)

```

root@5e55a711badb:/# ssh test@172.17.0.5
The authenticity of host '172.17.0.5 (172.17.0.5)' can't be established.
ECDSA key fingerprint is SHA256:M0+IklvyI9uvIOgIaiM/ZPg/D1Q/9wyiApkggJo/cI4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.17.0.5' (ECDSA) to the list of known hosts.
test@172.17.0.5's password:
Welcome to Ubuntu 20.10 (GNU/Linux 6.8.0-49-generic x86_64)

```

Figura 16: Conexión SSH entre C4 y S1.

La captura del tráfico se presenta a continuación.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	54028 → 22 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=3405828944 TSecr=0 WS=128
2	0.000048154	127.0.0.1	127.0.0.1	TCP	74	22 → 54028 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=3405828944 TSecr=3405828944
3	0.000083210	127.0.0.1	127.0.0.1	TCP	66	54028 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3405828944 TSecr=3405828944
4	0.001733615	127.0.0.1	127.0.0.1	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
5	0.001753831	127.0.0.1	127.0.0.1	TCP	66	22 → 54028 [ACK] Seq=1 Ack=42 Win=65536 Len=0 TSval=3405828946 TSecr=3405828946
6	0.035031718	127.0.0.1	127.0.0.1	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
7	0.035069172	127.0.0.1	127.0.0.1	TCP	66	54028 → 22 [ACK] Seq=42 Ack=42 Win=65536 Len=0 TSval=3405828979 TSecr=3405828979
8	0.035675966	127.0.0.1	127.0.0.1	SSHv2	1578	Client: Key Exchange Init
9	0.039739659	127.0.0.1	127.0.0.1	SSHv2	1122	Server: Key Exchange Init
10	0.046740319	127.0.0.1	127.0.0.1	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.062652962	127.0.0.1	127.0.0.1	SSHv2	574	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=228)
12	0.107761363	127.0.0.1	127.0.0.1	TCP	66	54028 → 22 [ACK] Seq=1602 Ack=1606 Win=65536 Len=0 TSval=3405829052 TSecr=3405829006
13	0.303920282	127.0.0.1	127.0.0.1	SSHv2	82	Client: New Keys
14	0.347778534	127.0.0.1	127.0.0.1	TCP	66	22 → 54028 [ACK] Seq=1606 Ack=1618 Win=65536 Len=0 TSval=3405835292 TSecr=3405835248
15	0.347814041	127.0.0.1	127.0.0.1	SSHv2	110	Client: Encrypted packet (len=44)
16	0.347835148	127.0.0.1	127.0.0.1	TCP	66	22 → 54028 [ACK] Seq=1606 Ack=1662 Win=65536 Len=0 TSval=3405835292 TSecr=3405835292
17	0.347992422	127.0.0.1	127.0.0.1	SSHv2	110	Server: Encrypted packet (len=44)
18	0.348015459	127.0.0.1	127.0.0.1	TCP	66	54028 → 22 [ACK] Seq=1662 Ack=1650 Win=65536 Len=0 TSval=3405835292 TSecr=3405835292
19	0.348168607	127.0.0.1	127.0.0.1	SSHv2	126	Client: Encrypted packet (len=60)
20	0.356170809	127.0.0.1	127.0.0.1	SSHv2	118	Server: Encrypted packet (len=52)
21	0.399784256	127.0.0.1	127.0.0.1	TCP	66	54028 → 22 [ACK] Seq=1722 Ack=1702 Win=65536 Len=0 TSval=3405835344 TSecr=3405835300

▶ Frame 4: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface lo, id 0
 ▶ Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
 ▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 ▶ Transmission Control Protocol, Src Port: 54028, Dst Port: 22, Seq: 1, Ack: 1, Len: 41
 - SSH Protocol
 Protocol: SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1
 [Direction: client-to-server]

0010	00 5d bc a0 40 00 40 06	7f f8 7f 00 00 01 7f 00].@-.....
0020	00 01 d3 0c 00 16 5c c2	ec 06 12 5d 4a c4 00 18	...N...J...
0030	02 00 fe 51 00 00 01 01	08 0a cb 0d d3 52 cb 00	...Q...R...
0040	d3 50 53 53 48 2d 32 2e	30 2d 4f 70 65 0e 53 53	-PSSH-2.0-OpenSS
0050	48 5f 38 2e 33 70 31 20	55 62 75 6e 74 75 2d 31	H.8.3p1 Ubuntu-1
0060	75 62 75 6e 74 75 30 2e	31 0d 0a	ubuntu0.1

Figura 17: Tráfico generado en conexión SSH entre C4 y S1.

Utilizando el archivo .pcapng generado, se puede observar el tráfico. Tanto cliente como servidor utilizan la misma versión de SSH, OpenSSH_8.3p1 Ubuntu-1ubuntu0.1. El intercambio de claves se observa en el paquete 8(1578 bytes) y 9(1122 bytes). Se realiza intercambio de llaves Diffie-Hellman y los paquetes siguientes se muestran cifrados.

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

Durante las conexiones realizadas entre los contenedores clientes y el servidor SSH, se observaron las siguientes versiones de OpenSSH y se derivaron los correspondientes HASSH:

Cliente	Versión de OpenSSH	HASSH derivado	Implementación esperada
C1	OpenSSH 6.6.1p1	<hash_C1>	OpenSSH 6.6.1p1
C2	OpenSSH 7.3p1	<hash_C2>	OpenSSH 7.3p1
C3	OpenSSH 7.7p1	<hash_C3>	OpenSSH 7.7p1
C4	OpenSSH 8.3p1	<hash_C4>	OpenSSH 8.3p1

Los resultados muestran que los HASSH obtenidos para cada cliente coinciden con las versiones de OpenSSH configuradas en los respectivos contenedores. Esto confirma que los clientes corresponden a las implementaciones esperadas y que no hubo alteraciones en los parámetros de los algoritmos negociados durante el intercambio de claves.

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

1.8.1. C1

- Inicio de sesión:
 - Se observa el intercambio de versiones del protocolo SSH (SSH-2.0-OpenSSH_6.6.1p1 para el cliente y OpenSSH_8.3p1 para el servidor).
 - Se intercambian listas de algoritmos soportados por el cliente y el servidor, como aes128-ctr y hmac-sha2-256.
- Intercambio de claves:
 - Los paquetes de intercambio (Key Exchange Init) contienen los parámetros para el algoritmo Diffie-Hellman. Aunque la clave en sí no se transmite en texto plano, los métodos de intercambio sí son visibles.
- Autenticación:
 - Se envía el nombre de usuario test en texto plano.
 - Las contraseñas no se envían en texto plano.

1.8.2. C2

- Inicio de sesión:
 - La versión del cliente SSH-2.0-OpenSSH_7.3p1.

1.8 Tipo de información contenida en cada uno de los paquetes de desarrollo (Parte 1)

- Los algoritmos propuestos son similares a los de C1, con diferencias menores en el orden o en versiones.
- Intercambio de claves:
 - Paquetes de tamaño reducido en comparación con C1 debido a mejoras en la versión de OpenSSH.
 - Se sigue utilizando Diffie-Hellman como método de intercambio de claves.
- Autenticación:
 - Se envía el nombre de usuario test en texto plano.
 - Las contraseñas no se envían en texto plano.

1.8.3. C3

- Inicio de sesión:
 - Cliente con versión SSH-2.0-OpenSSH_7.7p1.
 - Negociación de algoritmos, incluyendo aes128-ctr y ecdh-sha2-nistp256.
- Intercambio de claves:
 - Similar a C2, con paquetes de tamaño ajustado a las optimizaciones de la versión del cliente.
 - La clave de sesión se genera utilizando parámetros transmitidos en texto plano (no la clave en sí).
- Autenticación:
 - Se envía el nombre de usuario test en texto plano.
 - Las contraseñas no se envían en texto plano.

1.8.4. C4/S1

- Inicio de sesión:
 - Cliente y servidor usan la misma versión (SSH-2.0-OpenSSH_8.3p1).
 - Los algoritmos ofrecidos y seleccionados son visibles, pero optimizados para la configuración.
- Intercambio de claves:
 - Paquetes más grandes en comparación con otros clientes debido a la inclusión de parámetros adicionales para Diffie-Hellman.

- Similar a los otros clientes, los detalles clave no están en texto plano, pero los métodos sí.
- Autenticación:
 - Se envía el nombre de usuario test en texto plano.
 - Las contraseñas no se envían en texto plano.

1.9. Diferencia entre C1 y C2

La diferencia se encuentra en las versiones de OpenSSH y el tamaño de paquetes al intercambio de claves.

- C1: Ocupa OpenSSH_6.6.1p1 para Ubuntu-8, con paquetes de intercambio de claves de 2034 y 1122 bytes.
- C2: Ocupa OpenSSH_7.3p1 Ubuntu-1ubuntu0.1, con paquetes de 1498 y 1122 bytes.

Las diferencias reflejan cambios en implementaciones y configuraciones en las versiones de SSH.

1.10. Diferencia entre C2 y C3

Acá la diferencia también está en las versiones y tamaños de los paquetes de intercambio de claves.

- C2: Ocupa OpenSSH_7.3p1 Ubuntu-1ubuntu0.1, con paquetes de intercambio de claves de 1498 y 1122 bytes.
- C3: Ocupa OpenSSH_7.7p1 Ubuntu-4ubuntu6.3, con paquetes de 1426 y 1122 bytes.

Las variaciones indican que las diferencias en las versiones de OpenSSH radican en la cantidad de datos transmitidos en el intercambio de claves.

1.11. Diferencia entre C3 y C4

Acá la diferencia también está en las versiones y tamaños de los paquetes de intercambio de claves.

- C3: Ocupa OpenSSH_7.7p1 Ubuntu-4ubuntu6.3, con paquetes de 1426 y 1122 bytes.
- C4: Ocupa OpenSSH_8.3p1 Ubuntu-1ubuntu0.21. Los paquetes de intercambios de claves son de 1578 y 1122 bytes.

2. Desarrollo (Parte 2)

2.1. Identificación del cliente SSH con versión “?”

Se busca identificar a qué cliente corresponde la siguiente captura de 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 18: Tráfico generado del informante

Es clave observar el tamaño del paquete **Key Exchange Init**. Este tiene un tamaño de 1578 bytes. Esto coincide con la información obtenida del C4. Por esto, se concluye que la captura corresponde al cliente 4.

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

Esto se realiza en el contenedor C4S1. Para replicar el tráfico, es necesario recompilar el kernel de SSH modificando el valor de la versión a '?'. Esto se realiza en el contenedor C4S1 y comienza con la instalación de OpenSSH portable. Se realiza clonando el siguiente repositorio:

```
textbfgit clone https://github.com/openssh/openssh-portable
```

Además se deben instalar las dependencias: autoconf, libssl-dev, zlib1g-dev, gcc, make, git y vim.

```
root@5e55a711badb:/# apt install -y autoconf libssl-dev zlib1g-dev gcc make git vim
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  automake autotools-dev binutils binutils-common binutils-x86-64-linux-gnu
  cpp cpp-10 gcc-10 git-man less libasan6 libasn1-8-heimdal libatomic1
  libbinutils libc-dev-bin libc6-dev libcannaera0 libcc1-0 libcrypt-dev
  libctf-nobfd0 libctf0 libcurl3-gnutls liberror-perl libgcc-10-dev
  libgdbm-compat4 libgdbm6 libgomp1 libgpm2 libgssapi3-heimdal
  libhcrypto4-heimdal libheimbase1-heimdal libheimntlm0-heimdal
  libhx509-5-heimdal libisl22 libitm1 libkrb5-26-heimdal libldap2.4-2
```

Figura 19: Instalación de las dependencias necesarias.

Acá se muestra cómo se clona el repositorio, además del acceso al archivo `version.h` para su edición.

```
root@5e55a711badb:/# git clone https://github.com/openssh/openssh-portable
Cloning into 'openssh-portable'...
remote: Enumerating objects: 67743, done.
remote: Counting objects: 100% (3848/3848), done.
remote: Compressing objects: 100% (118/118), done.
remote: Total 67743 (delta 3770), reused 3760 (delta 3730), pack-reused 63895 (from 1)
Receiving objects: 100% (67743/67743), 28.43 MiB | 3.71 MiB/s, done.
Resolving deltas: 100% (52548/52548), done.
root@5e55a711badb:/# cd openssh-portable
root@5e55a711badb:/openssh-portable# vim version.h
```

Figura 20: Consola para C4S1, acceso al archivo version.h.

A continuación se presenta el archivo inicialmente.

```
/* $OpenBSD: version.h,v 1.103 2024/09/19 22:17:44 djm Exp $ */

#define SSH_VERSION      "OpenSSH_9.9"

#define SSH_PORTABLE     "p1"
#define SSH_RELEASE      SSH_VERSION SSH_PORTABLE

...

"version.h" 6L, 171C                                1,1                All
```

Figura 21: Archivo original version.h

Luego, este se modifica. Se cambia la versión de OpenSSH a '?' y además se elimina el valor **SSHPORTABLE**.

[illegible]

Figura 22: Archivo version.h editado.

A continuación:

- Se ejecuta ***autoreconf*** para generar el archivo './configure'.
- Una vez generado tal archivo, se ejecuta para configurar el entorno para compilar OpenSSH.
- Se utiliza el comando ***make*** y ***make install*** para compilar el proyecto. Así se construyen ejecutables y bibliotecas necesarias.
- Se busca la ubicación de ***sshd*** en el server SSH, con el comando ***which sshd*** que entrega su ruta. Se ejecuta ***sshd*** para verificar su funcionamiento.

```
root@5e55a711badb:/openssh-portable# which sshd
/usr/local/sbin/sshd
root@5e55a711badb:/openssh-portable# ps -aux | grep sshd
root      12079  0.0  0.0   3284   1792 pts/1    S+   19:22   0:00 grep --color=auto sshd
root@5e55a711badb:/openssh-portable#
```

Figura 23: Ejecución y verificación SSH.

- Se captura el tráfico en la interfaz loopback con `tshark` y se realiza la conexión entre C4 y S1.

```

test@Se55a711badb: ~
root@Se55a711badb: /

root@Se55a711badb:/openssh-portable# ssh test@localhost
ssh: connect to host localhost port 22: Connection refused
root@Se55a711badb:/openssh-portable# service ssh status
* sshd is not running
root@Se55a711badb:/openssh-portable# service ssh start
* Starting OpenBSD Secure Shell server sshd
[ OK ]
root@Se55a711badb:/openssh-portable# ssh test@localhost
The authenticity of host 'localhost (::1)' can't be established.

```

Figura 24: Conexión entre C4 y S1.

A continuación se observa en el tráfico que la versión es SSH-2.0-OpenSSH_? al igual que en el tráfico que se replicará.

3	0.000057952	127.0.0.1	127.0.0.1	TCP	66 34686 → 22 [ACK] Seq=1 Ack=1 Win=65535
4	0.000770889	127.0.0.1	127.0.0.1	SSHv2	85 Client: Protocol (SSH-2.0-OpenSSH_?)
5	0.000786172	127.0.0.1	127.0.0.1	TCP	66 22 → 34686 [ACK] Seq=1 Ack=20 Win=65535
6	0.020160576	127.0.0.1	127.0.0.1	SSHv2	85 Server: Protocol (SSH-2.0-OpenSSH_?)
7	0.020190656	127.0.0.1	127.0.0.1	TCP	66 34686 → 22 [ACK] Seq=20 Ack=20 Win=65535
8	0.020982387	127.0.0.1	127.0.0.1	SSHv2	1570 Client: Key Exchange Init
9	0.022614490	127.0.0.1	127.0.0.1	SSHv2	1146 Server: Key Exchange Init
10	0.064854936	127.0.0.1	127.0.0.1	TCP	66 34686 → 22 [ACK] Seq=1524 Ack=1100 Win=65535
11	0.142421485	127.0.0.1	127.0.0.1	SSHv2	1274 Client: Diffie-Hellman Key Exchange
12	0.156596749	127.0.0.1	127.0.0.1	SSHv2	1654 Server: Diffie-Hellman Key Exchange
13	0.156605901	127.0.0.1	127.0.0.1	TCP	66 34686 → 22 [ACK] Seq=2732 Ack=2688 Win=65535
14	2.181521966	127.0.0.1	127.0.0.1	SSHv2	82 Client: New Keys
15	2.224904267	127.0.0.1	127.0.0.1	TCP	66 22 → 34686 [ACK] Seq=2688 Ack=2748 Win=65535
16	2.224926642	127.0.0.1	127.0.0.1	SSHv2	110 Client: Encrypted packet (len=44)
17	2.224941631	127.0.0.1	127.0.0.1	TCP	66 22 → 34686 [ACK] Seq=2688 Ack=2792 Win=65535
18	2.225014682	127.0.0.1	127.0.0.1	SSHv2	110 Server: Encrypted packet (len=44)
19	2.225023128	127.0.0.1	127.0.0.1	TCP	66 34686 → 22 [ACK] Seq=2792 Ack=2732 Win=65535
20	2.225126355	127.0.0.1	127.0.0.1	SSHv2	126 Client: Encrypted packet (len=60)
21	2.226670246	127.0.0.1	127.0.0.1	SSHv2	142 Server: Encrypted packet (len=76)
22	2.226680007	127.0.0.1	127.0.0.1	SSHv2	150 Client: Encrypted packet (len=84)

▶ Frame 4: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface lo, id 0
 ▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
 ▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 ▶ Transmission Control Protocol, Src Port: 34686, Dst Port: 22, Seq: 1, Ack: 1, Len: 19
 ▶ SSH Protocol
 Protocol: SSH-2.0-OpenSSH_?
 [Direction: client-to-server]

Figura 25: Versión SSH observada en Wireshark.

3. Desarrollo (Parte 3)

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

A continuación se quiere replicar el siguiente tráfico.

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

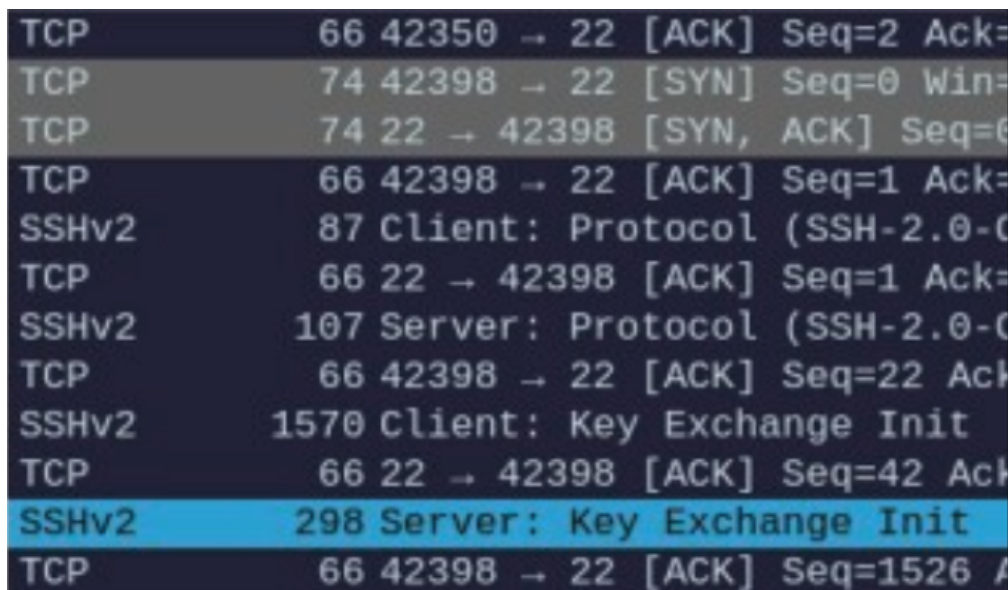


Figura 26: Captura del Key Exchange

Para esto, se modifica la cantidad de datos manejados en el KEI del server SSH. Se logra reduciendo al mínimo los algoritmos disponibles, usando solo uno en cada categoría para disminuir el tamaño del KEI y asegurar una conexión correcta.

- Se descarga OpenSSH Portable.
- Se modifica el archivo **sshd_config**, seleccionando los algoritmos más ligeros soportados por OpenSSH Portable. Estos se observan a continuación.

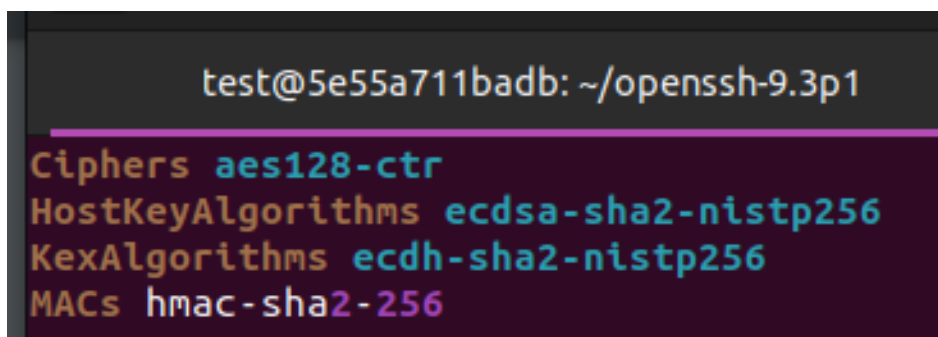


Figura 27: Algoritmos ligeros ocupados.

Luego, en el tráfico se observa que el paquete de Key Exchange del servidor reduce su tamaño a 266 bytes. Esto se debe a la reducción de los algoritmos a trabajar por el servidor.

TCP	66	22 → 46132	[ACK] Seq=1 Ack=22
SSHv2	87	Server: Protocol (SSH-2.0-OpenSSH_7.6p1)	
TCP	66	46132 → 22	[ACK] Seq=22 Ack=87
SSHv2	1570	Client: Key Exchange Init	
SSHv2	266	Server: Key Exchange Init	
SSHv2	146	Client: Elliptic Curve Diffie-Hellman	
SSHv2	746	Server: Elliptic Curve Diffie-Hellman	
TCP	66	46132 → 22	[ACK] Seq=1606 Ack=82
SSHv2	82	Client: New Keys	
TCP	66	22 → 46132	[ACK] Seq=902 Ack=82

Figura 28: Tamaño KEI modificado.

4. Desarrollo (Parte 4)

4.1. Explicación OpenSSH en general

OpenSSH es una suite de herramientas diseñadas para comunicación segura en redes no confiables. Esta sirve para implementar el protocolo SSH en un estándar criptográfico, permitiendo acceder de forma remota a sistemas y transferencia segura de datos.

Su principal propósito recae en:

- Acceso remoto seguro.
- Transferencia segura de archivos.
- Túneles seguros. Permite redirigir tráfico en túneles cifrados, útil para bypass de restricciones o protección de datos.

OpenSSH cuenta con cifrado robusto, utiliza distintas herramientas (como ssh, sshd, scp, sftp y ssh-keygen), flexibilidad en su configuración y seguridad en su diseño. Sirve para Administración de servidores, estionando servidores de forma remota y segura. Sirve para scripts y procesos automatizados, permitiendo copiar archivos y ejecutar comandos de forma remota. También sirve para proteger comunicaciones internas. Empresas lo emplean para cifrar comunicaciones sensibles en sus infraestructuras.

4.2. Capas de Seguridad en OpenSSH

Las principales capas de seguridad son:

- **Capa de transporte:** Asegura que los datos transmitidos entre cliente y servidor se encuentren cifrados y protegidos contra ataques como la interceptación o modificación en tránsito.
- **Capa de Autenticación:** Sirve para garantizar que el cliente (o servidor) sean quienes dicen ser. Esto se puede realizar para el cliente, por contraseña, clave pública/privada y mediante agentes de autenticación (como ssh-agent).

- **Capa de conexión:** Esta capa permite gestionar canales lógicos en la conexión SSH. Estos soportan múltiples tipos de comunicación en paralelo.
- **Capa de gestión de claves:** OpenSSH tiene mecanismos avanzados para manejar claves criptográficas y configuraciones de seguridad que refuerzan sus capas.

4.3. Identificación de que protocolos no se cumplen

Al configurar y usar OpenSSH, se necesita seguir los protocolos establecidos para garantizar seguridad y correcto funcionamiento del sistema. Hay casos donde esto no se cumple, poniendo en riesgo la confidencialidad, integridad o autenticidad de las conexiones.

Un ejemplo es Falta de autenticación adecuada del server. El cliente debe verificar la clave pública del server en el archivo *known_hosts*. Si la clave pública no está registrada allí, o se omite la verificación, podría ocurrir un ataque de intermediario como el cliente no puede confirmar la identidad del server.

Otro ejemplo es la utilización de algoritmos inseguros o obsoletos como DES o 3DES. Esto debilita la seguridad del cifrado y facilita ataques de fuerza bruta o colisión.

También claves de autenticación mal gestionadas, pues las claves privadas deben estar protegidas por contraseñas fuertes y almacenadas en ubicaciones seguras, si no se cumple lo anterior, expone al usuario si se filtra el archivo *id_rsa*.

Conclusiones y comentarios

A lo largo de este laboratorio se observó el comportamiento de distintas versiones de contenedores Docker usando Ubuntu y SSH. Así, se logra una comprensión de cómo las variaciones de versiones pueden afectar la seguridad y rendimiento de las conexiones SSH.

Al observar las diferencias en versiones OpenSSH, se observan cambios notables en los tamaños de paquetes en el intercambio de claves y también en los algoritmos de cifrado empleados. Por esto, es importante mantener actualizadas las herramientas para poder garantizar seguridad y eficiencia.

Para OpenSSH Portable, al seleccionar algoritmos específicos, se puede reducir considerablemente el tamaño de los paquetes. Esto es crucial en entornos donde el rendimiento y eficiencia de la red importan, por lo que se refleja la flexibilidad ofrecida por el software de código abierto.