

Informe Laboratorio 5

Sección 4

Alumno Pablo Lores
e-mail: pablo.lores_s.contacto@mail.udp.cl

Noviembre de 2024

Índice

Descripción de actividades	3
1. Desarrollo (Parte 1)	6
1.1. Códigos de cada Dockerfile	6
1.1.1. C1	6
1.1.2. C2	7
1.1.3. C3	7
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)	9
1.5. Tráfico generado por C3, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)	9
1.6. Tráfico generado por C4 (iface lo), detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)	9
1.7. Compara la versión de HASSH obtenida con la base de datos para validar si el cliente corresponde al mismo	9
1.8. Tipo de información contenida en cada uno de los paquetes generados en texto plano	9
1.8.1. C1	9
1.8.2. C2	9
1.8.3. C3	9
1.8.4. C4/S1	9
1.9. Diferencia entre C1 y C2	9
1.10. Diferencia entre C2 y C3	9
1.11. Diferencia entre C3 y C4	9
2. Desarrollo (Parte 2)	10
2.1. Identificación del cliente SSH con versión “?”	10
2.2. Replicación de tráfico al servidor (paso por paso)	10
3. Desarrollo (Parte 3)	10
3.1. Replicación del KEI con tamaño menor a 300 bytes (paso por paso)	10
4. Desarrollo (Parte 4)	10
4.1. Explicación OpenSSH en general	10
4.2. Capas de Seguridad en OpenSSH	10
4.3. Identificación de que protocolos no se cumplen	10

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

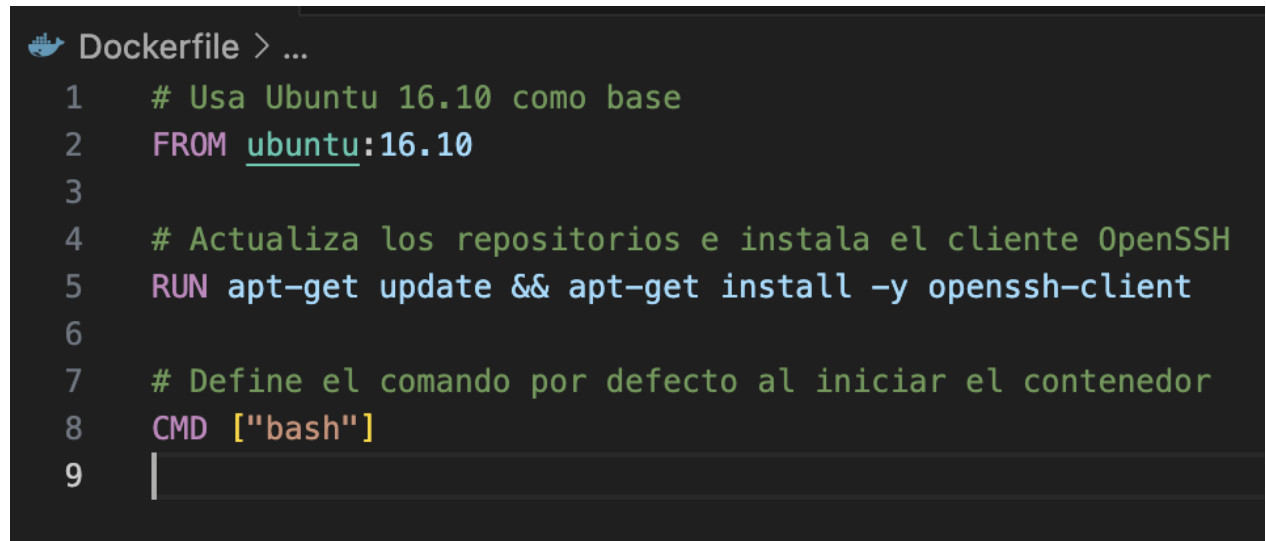
Figura 2: Captura del Key Exchange

- 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

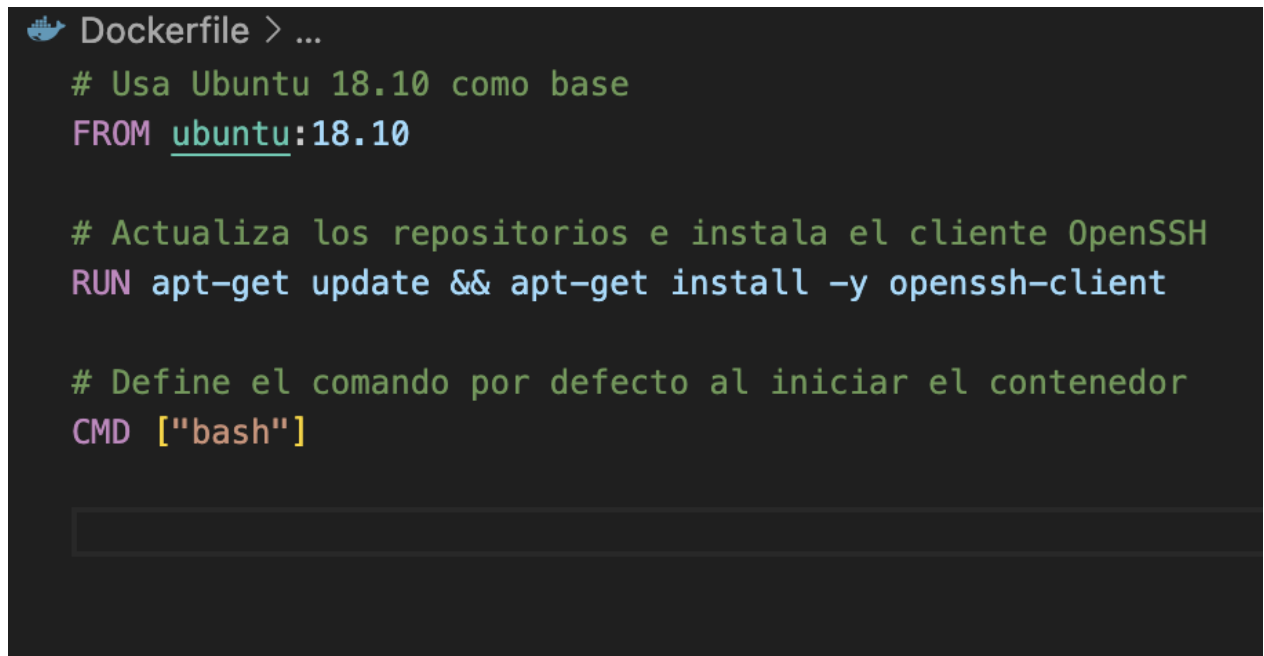
1.1.1. C1



```
Dockerfile > ...
1  # Usa Ubuntu 16.10 como base
2  FROM ubuntu:16.10
3
4  # Actualiza los repositorios e instala el cliente OpenSSH
5  RUN apt-get update && apt-get install -y openssh-client
6
7  # Define el comando por defecto al iniciar el contenedor
8  CMD ["bash"]
9  |
```

Figura 3: Captura Docker 1

1.1.2. C2

A screenshot of a Dockerfile editor interface. The title bar shows a Docker icon and the text 'Dockerfile > ...'. The code is as follows:

```
# Usa Ubuntu 18.10 como base
FROM ubuntu:18.10

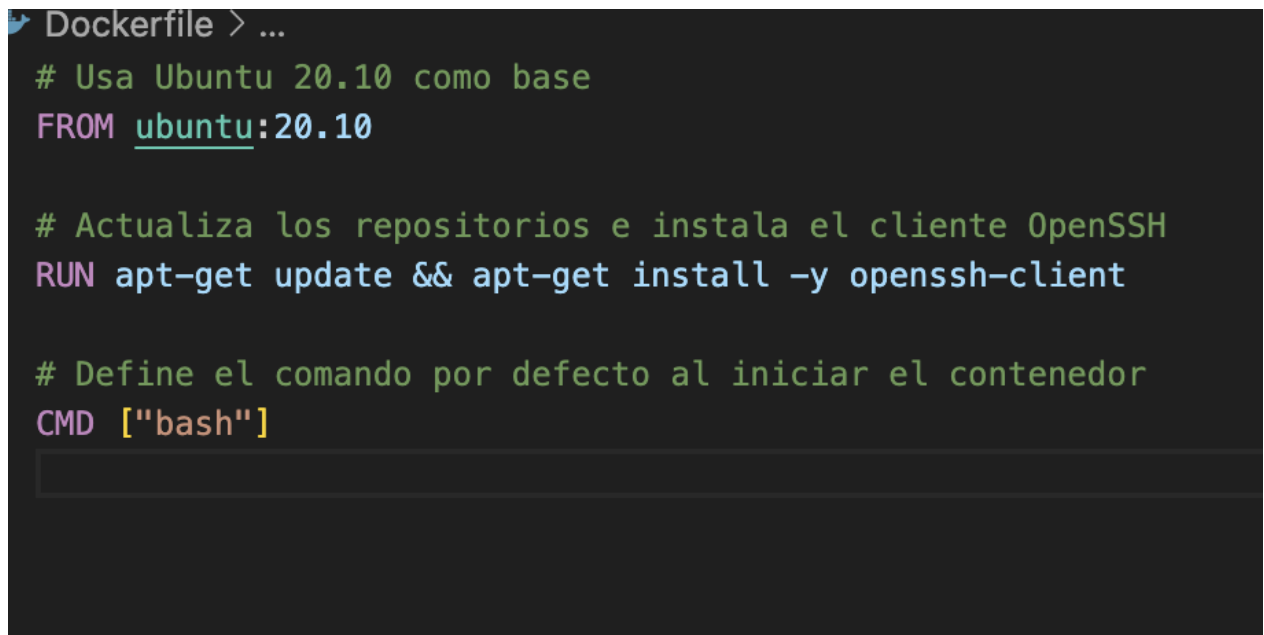
# Actualiza los repositorios e instala el cliente OpenSSH
RUN apt-get update && apt-get install -y openssh-client

# Define el comando por defecto al iniciar el contenedor
CMD ["bash"]
```

There is an empty input field at the bottom of the editor.

Figura 4: Captura Docker 2

1.1.3. C3

A screenshot of a Dockerfile editor interface. The title bar shows a Docker icon and the text 'Dockerfile > ...'. The code is as follows:

```
# Usa Ubuntu 20.10 como base
FROM ubuntu:20.10

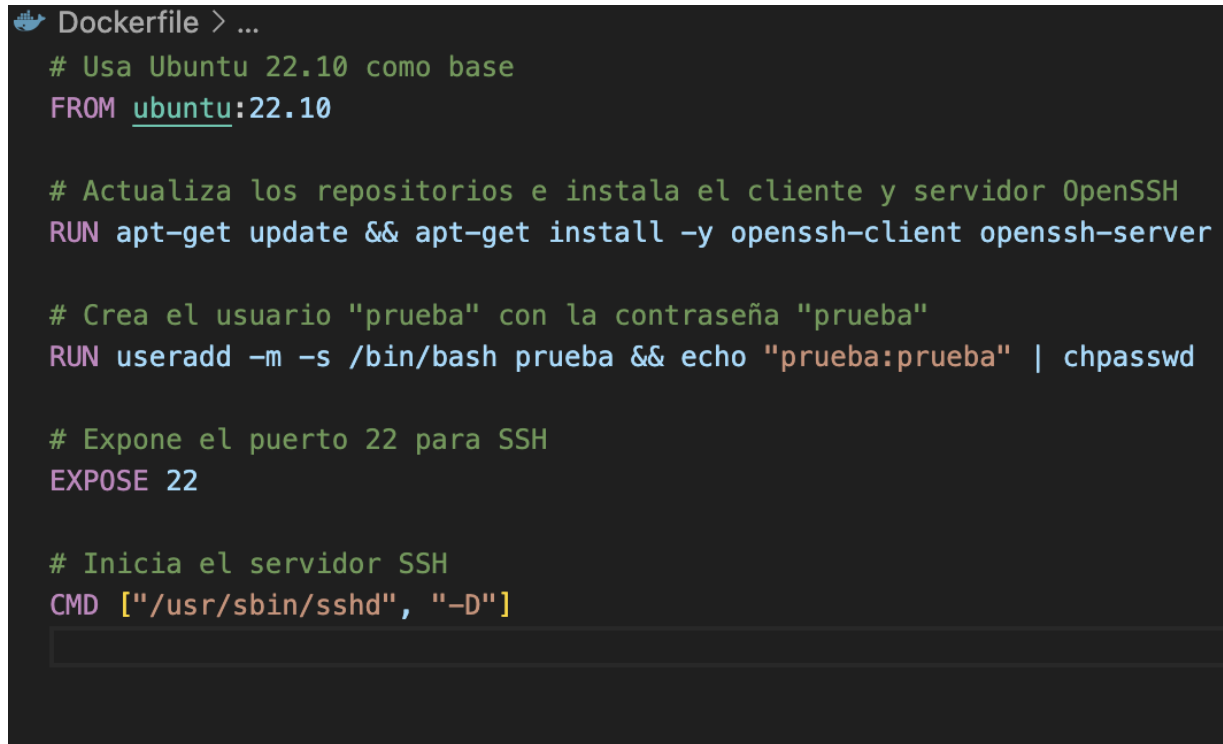
# Actualiza los repositorios e instala el cliente OpenSSH
RUN apt-get update && apt-get install -y openssh-client

# Define el comando por defecto al iniciar el contenedor
CMD ["bash"]
```

There is an empty input field at the bottom of the editor.

Figura 5: Captura Docker 3

1.1.4. C4/S1



```
Dockerfile > ...  
# Usa Ubuntu 22.10 como base  
FROM ubuntu:22.10  
  
# Actualiza los repositorios e instala el cliente y servidor OpenSSH  
RUN apt-get update && apt-get install -y openssh-client openssh-server  
  
# Crea el usuario "prueba" con la contraseña "prueba"  
RUN useradd -m -s /bin/bash prueba && echo "prueba:prueba" | chpasswd  
  
# Expone el puerto 22 para SSH  
EXPOSE 22  
  
# Inicia el servidor SSH  
CMD ["/usr/sbin/sshd", "-D"]
```

Figura 6: Captura Docker 4

- 1.2. Creación de las credenciales para S1
- 1.3. Tráfico generado por C1, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)
- 1.4. Tráfico generado por C2, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)
- 1.5. Tráfico generado por C3, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)
- 1.6. Tráfico generado por C4 (iface lo), detallando tamaño paquetes del flujo y el HASSH respectivo (detallado)
- 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
 - 1.8.1. C1
 - 1.8.2. C2
 - 1.8.3. C3
 - 1.8.4. C4/S1
- 1.9. Diferencia entre C1 y C2
- 1.10. Diferencia entre C2 y C3
- 1.11. Diferencia entre 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)

4. Desarrollo (Parte 4)

- 4.1. Explicación OpenSSH en general
- 4.2. Capas de Seguridad en OpenSSH
- 4.3. Identificación de que protocolos no se cumplen

Conclusiones y comentarios