

Informe Laboratorio 5

Sección x

Alumno x

e-mail: alumno.contacto@mail.udp.cl

Junio de 2023

Índice

1. Descripción de actividades	2
2. Desarrollo (Parte 1)	5
2.1. Códigos de cada Dockerfile	5
2.1.1. C1	5
2.1.2. C2	5
2.1.3. C3	5
2.1.4. C4/S1	6
2.2. Creación de las credenciales para S1	6
2.3. Tráfico generado por C1 (detallado)	6
2.4. Tráfico generado por C2 (detallado)	7
2.5. Tráfico generado por C3 (detallado)	8
2.6. Tráfico generado por C4 (iface lo) (detallado)	8
2.7. Diferencia entre C1 y C2	9
2.8. Diferencia entre C2 y C3	9
2.9. Diferencia entre C3 y C4	9
3. Desarrollo (Parte 2)	9
3.1. Identificación del cliente ssh	9
3.2. Replicación de tráfico (paso por paso)	9
4. Desarrollo (Parte 3)	11
4.1. Replicación de tráfico (paso por paso)	11

1. 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, donde cada uno tendrá el siguiente SO: Ubuntu 14.10, Ubuntu 16.10, Ubuntu 18.10 y Ubuntu 20.10, a los cuales llamaremos C1,C2,C3,C4/S1 respectivamente.
- Para cada uno de ellos, deberá instalar la última versión, disponible en sus repositorios, del cliente y servidor openssh.
- En S1 deberá crear el usuario test con contraseña test, para acceder a él desde los otros contenedores.
- En total serán 4 escenarios, donde cada uno corresponderá a los siguientes equipos:
 - C1 → S1
 - C2 → S1
 - C3 → S1
 - C4 → S1

Pasos:

1. Para cada uno de los 4 escenarios, solo deberá establecer la conexión y no realizar ningún otro comando que pueda generar tráfico (como muestra la Figura). Deberá capturar el tráfico de red generado y analizar el patrón de tráfico generado por cada cliente. De esta forma podrá obtener una huella digital para cada cliente a partir de su tráfico.

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

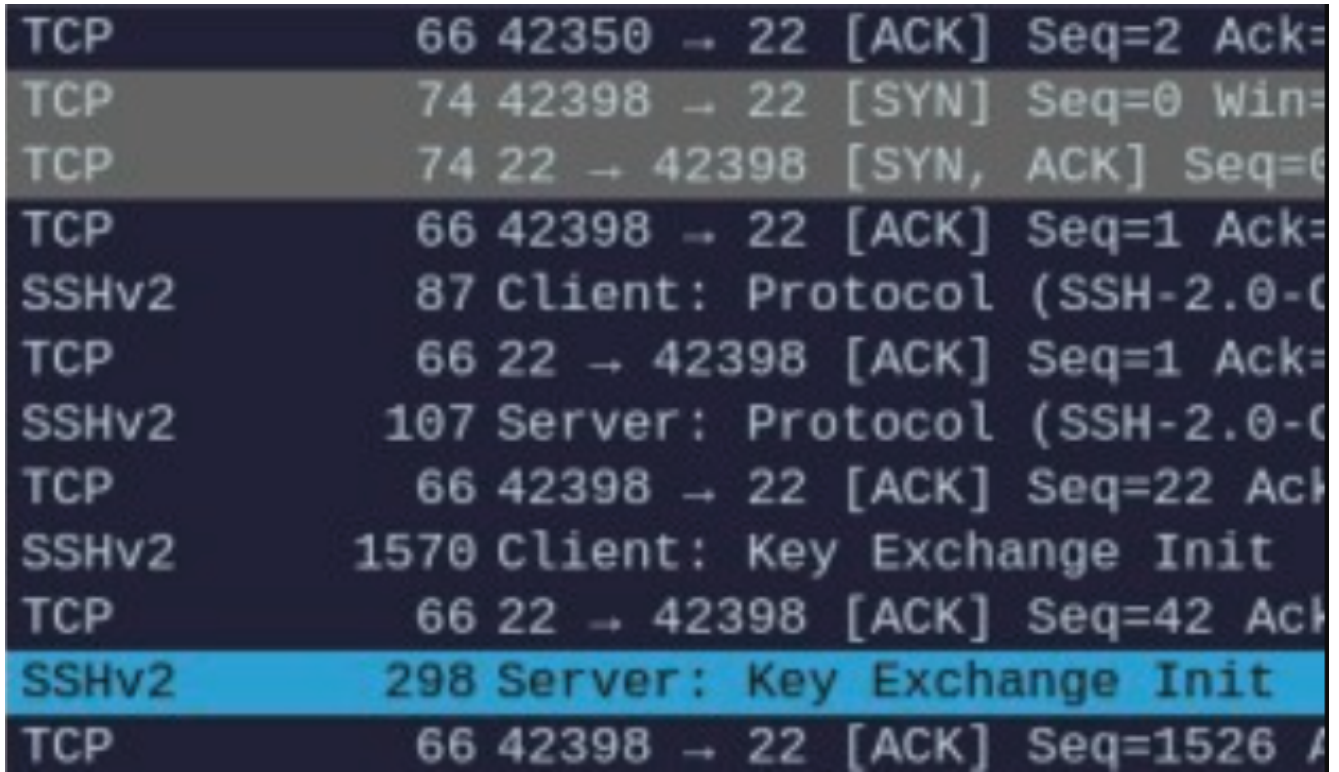
2. Para poder identificar que el usuario efectivamente es el informante, éste utilizará una versión única de cliente. ¿Con qué cliente SSH se habrá generado el siguiente tráfico?

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.

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.



The image shows a network traffic capture with the following entries:

TCP	66	42350 → 22	[ACK]	Seq=2	Ack=
TCP	74	42398 → 22	[SYN]	Seq=0	Win=
TCP	74	22 → 42398	[SYN, ACK]	Seq=6	
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	Ac
SSHv2	1570	Client: Key Exchange Init			
TCP	66	22 → 42398	[ACK]	Seq=42	Ac
SSHv2	298	Server: Key Exchange Init			
TCP	66	42398 → 22	[ACK]	Seq=1526	A

Figura 2: Captura del Key Exchange

2. Desarrollo (Parte 1)

2.1. Códigos de cada Dockerfile

Todos los contenedores se encuentran en versiones de Ubuntu en el estado End of Life (sin soporte activo). Es por esto que los repositorios de `archive.ubuntu.com` y `security.ubuntu.com` para estas versiones no se encuentran disponibles, es por esto que remplazamos ambas URL por `old-releases.ubuntu.com`, además de esto, los 3 primeros contenedores consisten únicamente de un cliente SSH mientras que el cuarto es el único que también cumple rol de servidor SSH.

2.1.1. C1

```
1 FROM ubuntu:14.10
2
3 # Use old-releases.ubuntu.com instead of archive.ubuntu.com or
  security.ubuntu.com
4 RUN sed -i -re 's/([a-z]{2}\.)?archive.ubuntu.com|security.ubuntu.
  com/old-releases.ubuntu.com/g' /etc/apt/sources.list
5
6 RUN apt-get update
7 RUN apt-get install -y openssh-client
```

2.1.2. C2

```
1 FROM ubuntu:16.10
2
3 # Use old-releases.ubuntu.com instead of archive.ubuntu.com or
  security.ubuntu.com
4 RUN sed -i -re 's/([a-z]{2}\.)?archive.ubuntu.com|security.ubuntu.
  com/old-releases.ubuntu.com/g' /etc/apt/sources.list
5
6 RUN apt-get update
7 RUN apt-get install -y openssh-client
```

2.1.3. C3

```
1 FROM ubuntu:18.10
2
3 # Use old-releases.ubuntu.com instead of archive.ubuntu.com or
  security.ubuntu.com
4 RUN sed -i -re 's/([a-z]{2}\.)?archive.ubuntu.com|security.ubuntu.
  com/old-releases.ubuntu.com/g' /etc/apt/sources.list
5
```

```
6 RUN apt-get update
7 RUN apt-get install -y openssh-client
```

2.1.4. C4/S1

```
1 FROM ubuntu:20.10
2
3 # Use old-releases.ubuntu.com instead of archive.ubuntu.com or
  security.ubuntu.com
4 RUN sed -i -re 's/([a-z]{2}\.)?archive.ubuntu.com|security.ubuntu.
  com/old-releases.ubuntu.com/g' /etc/apt/sources.list
5
6 RUN apt-get update
7 RUN apt-get install -y \
8     openssh-client \
9     openssh-server
10
11 # Create "test" user
12 RUN useradd -m -s /bin/bash test
13 # Create password for test user, password: test
14 RUN echo "test:test" | chpasswd
15
16 # Expose port 22 for SSH connection
17 EXPOSE 22
18
19 ENTRYPOINT service ssh restart && bash
```

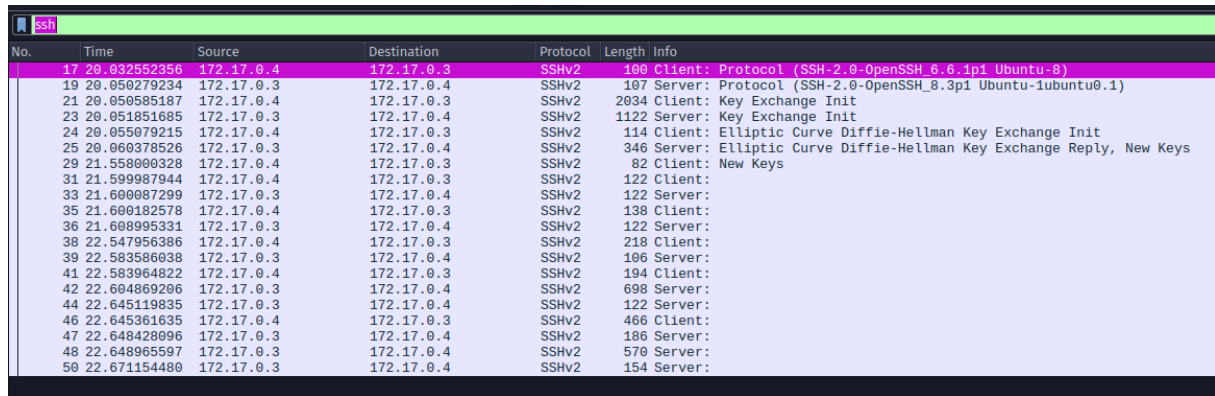
2.2. Creación de las credenciales para S1

Para la creación de credenciales se necesita un usuario y definir una contraseña para el mismo. Esto se logra con los siguientes comandos.

```
1 # Create "test" user
2 RUN useradd -m -s /bin/bash test
3 # Create password for test user, password: test
4 RUN echo "test:test" | chpasswd
```

2.3. Tráfico generado por C1 (detallado)

En todas las capturas de tráfico el protocolo SSH funciona sobre TCP, por lo que contiene un handshake y un ACK en cada paso del protocolo SSH. Por simplicidad se usa el filtro SSH en Wireshark para todas las capturas.

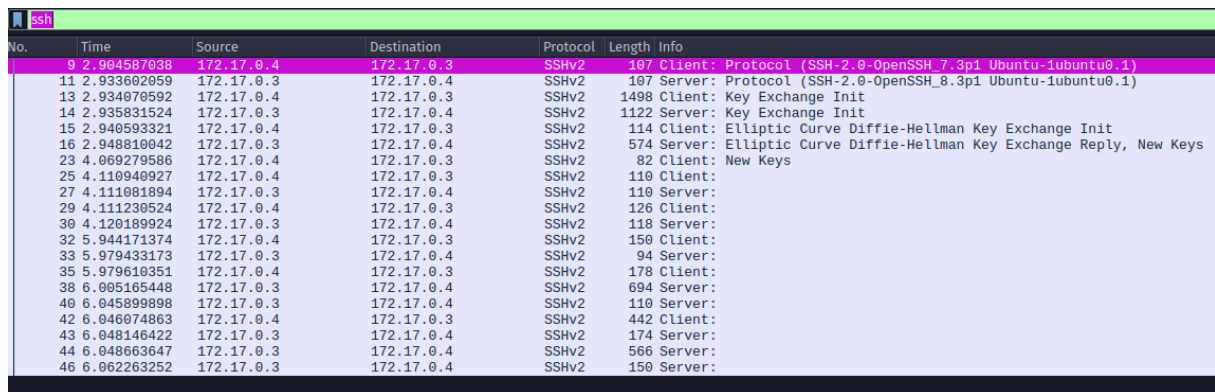


No.	Time	Source	Destination	Protocol	Length	Info
17	20.032552356	172.17.0.4	172.17.0.3	SSHv2	100	Client: Protocol (SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-8)
19	20.050279234	172.17.0.3	172.17.0.4	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
21	20.050585187	172.17.0.4	172.17.0.3	SSHv2	2034	Client: Key Exchange Init
23	20.051851685	172.17.0.3	172.17.0.4	SSHv2	1122	Server: Key Exchange Init
24	20.055079215	172.17.0.4	172.17.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
25	20.060378526	172.17.0.3	172.17.0.4	SSHv2	346	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
29	21.558000328	172.17.0.4	172.17.0.3	SSHv2	82	Client: New Keys
31	21.599987944	172.17.0.3	172.17.0.4	SSHv2	122	Client:
33	21.600087299	172.17.0.3	172.17.0.4	SSHv2	122	Server:
35	21.600182578	172.17.0.4	172.17.0.3	SSHv2	138	Client:
36	21.608995331	172.17.0.3	172.17.0.4	SSHv2	122	Server:
38	22.547956386	172.17.0.4	172.17.0.3	SSHv2	218	Client:
39	22.583586038	172.17.0.3	172.17.0.4	SSHv2	106	Server:
41	22.583964822	172.17.0.4	172.17.0.3	SSHv2	194	Client:
42	22.604869206	172.17.0.3	172.17.0.4	SSHv2	698	Server:
44	22.645119835	172.17.0.4	172.17.0.3	SSHv2	122	Server:
46	22.645361635	172.17.0.4	172.17.0.3	SSHv2	466	Client:
47	22.648428096	172.17.0.3	172.17.0.4	SSHv2	186	Server:
48	22.648965597	172.17.0.3	172.17.0.4	SSHv2	570	Server:
50	22.671154480	172.17.0.3	172.17.0.4	SSHv2	154	Server:

Figura 3: Captura c1 (Ubuntu 14.10) a s1 (Ubuntu 20.10).

Desde la captura se puede desprender la versión de OpenSSH del cliente (v6.6.1) y del servidor (v8.3). El protocolo se puede conocer en el intercambio de llaves, en este caso el protocolo utilizado es Diffie-Hellman. Y luego, 13 mensajes cifrados de distintos tamaños.

2.4. Tráfico generado por C2 (detallado)



No.	Time	Source	Destination	Protocol	Length	Info
9	2.904587038	172.17.0.4	172.17.0.3	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.3p1 Ubuntu-1ubuntu0.1)
11	2.933602059	172.17.0.3	172.17.0.4	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
13	2.934070592	172.17.0.4	172.17.0.3	SSHv2	1498	Client: Key Exchange Init
14	2.935831524	172.17.0.3	172.17.0.4	SSHv2	1122	Server: Key Exchange Init
15	2.940593321	172.17.0.4	172.17.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
16	2.948810042	172.17.0.3	172.17.0.4	SSHv2	574	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
23	4.069279586	172.17.0.4	172.17.0.3	SSHv2	82	Client: New Keys
25	4.110940927	172.17.0.4	172.17.0.3	SSHv2	110	Client:
27	4.111081894	172.17.0.3	172.17.0.4	SSHv2	110	Server:
29	4.111230524	172.17.0.4	172.17.0.3	SSHv2	126	Client:
30	4.120189924	172.17.0.3	172.17.0.4	SSHv2	118	Server:
32	5.944171374	172.17.0.4	172.17.0.3	SSHv2	150	Client:
33	5.979433173	172.17.0.3	172.17.0.4	SSHv2	94	Server:
35	5.979610351	172.17.0.4	172.17.0.3	SSHv2	178	Client:
38	6.005165448	172.17.0.3	172.17.0.4	SSHv2	694	Server:
40	6.045899898	172.17.0.4	172.17.0.3	SSHv2	110	Server:
42	6.046074863	172.17.0.4	172.17.0.3	SSHv2	442	Client:
43	6.048146422	172.17.0.3	172.17.0.4	SSHv2	174	Server:
44	6.048663647	172.17.0.3	172.17.0.4	SSHv2	566	Server:
46	6.062263252	172.17.0.3	172.17.0.4	SSHv2	150	Server:

Figura 4: Captura c2 (Ubuntu 16.10) a s1 (Ubuntu 20.10).

De la captura se desprende la versión de OpenSSH del cliente (v.7.3) y del servidor (v8.3). Utilizan el mismo protocolo que la captura anterior (Diffie-Hellman). Igualmente termina con 13 mensajes cifrados de distintos tamaños.

2.5. Tráfico generado por C3 (detallado)

No.	Time	Source	Destination	Protocol	Length	Info
15	4.968177769	172.17.0.4	172.17.0.3	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.7p1 Ubuntu-4ubuntu0.3)
17	4.994218754	172.17.0.3	172.17.0.4	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
19	4.994699036	172.17.0.4	172.17.0.3	SSHv2	1426	Client: Key Exchange Init
20	4.996167671	172.17.0.3	172.17.0.4	SSHv2	1122	Server: Key Exchange Init
21	4.999371278	172.17.0.4	172.17.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
22	5.005699432	172.17.0.3	172.17.0.4	SSHv2	574	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
26	6.434356200	172.17.0.4	172.17.0.3	SSHv2	82	Client: New Keys
28	6.474760154	172.17.0.3	172.17.0.4	SSHv2	110	Client:
30	6.474882485	172.17.0.3	172.17.0.4	SSHv2	110	Server:
32	6.474954773	172.17.0.3	172.17.0.4	SSHv2	126	Client:
33	6.483804222	172.17.0.3	172.17.0.4	SSHv2	118	Server:
35	8.303291401	172.17.0.3	172.17.0.4	SSHv2	150	Client:
36	8.338984744	172.17.0.3	172.17.0.4	SSHv2	94	Server:
38	8.339343739	172.17.0.3	172.17.0.4	SSHv2	178	Client:
39	8.366406480	172.17.0.3	172.17.0.4	SSHv2	694	Server:
41	8.406976864	172.17.0.3	172.17.0.4	SSHv2	110	Server:
43	8.407278867	172.17.0.4	172.17.0.3	SSHv2	442	Client:
44	8.410893511	172.17.0.3	172.17.0.4	SSHv2	174	Server:
45	8.411378988	172.17.0.3	172.17.0.4	SSHv2	566	Server:
47	8.432395622	172.17.0.3	172.17.0.4	SSHv2	150	Server:
49	9.253519579	172.17.0.4	172.17.0.3	SSHv2	118	Client:
50	9.253791709	172.17.0.3	172.17.0.4	SSHv2	118	Server:

Figura 5: Captura c3 (Ubuntu 18.10) a s1 (Ubuntu 20.10).

De la captura se desprende la versión de OpenSSH del cliente (v.7.7) y del servidor (v8.3). Utilizan el mismo protocolo que la captura anterior (Diffie-Hellman). Termina con 14 mensajes cifrados de distintos tamaños.

2.6. Tráfico generado por C4 (iface lo) (detallado)

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000685482	172.17.0.3	172.17.0.3	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
6	0.014599146	172.17.0.3	172.17.0.3	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
8	0.014784699	172.17.0.3	172.17.0.3	SSHv2	1578	Client: Key Exchange Init
9	0.016021357	172.17.0.3	172.17.0.3	SSHv2	1122	Server: Key Exchange Init
10	0.018319717	172.17.0.3	172.17.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.023480810	172.17.0.3	172.17.0.3	SSHv2	574	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
13	2.395451286	172.17.0.3	172.17.0.3	SSHv2	82	Client: New Keys
15	2.436504583	172.17.0.3	172.17.0.3	SSHv2	110	Client:
17	2.436631296	172.17.0.3	172.17.0.3	SSHv2	110	Server:
19	2.436774046	172.17.0.3	172.17.0.3	SSHv2	126	Client:
20	2.445663173	172.17.0.3	172.17.0.3	SSHv2	118	Server:
22	4.569866123	172.17.0.3	172.17.0.3	SSHv2	150	Client:
23	4.605067993	172.17.0.3	172.17.0.3	SSHv2	94	Server:
25	4.605216421	172.17.0.3	172.17.0.3	SSHv2	178	Client:
26	4.624553638	172.17.0.3	172.17.0.3	SSHv2	694	Server:
28	4.665550659	172.17.0.3	172.17.0.3	SSHv2	110	Server:
30	4.665644932	172.17.0.3	172.17.0.3	SSHv2	442	Client:
31	4.667557427	172.17.0.3	172.17.0.3	SSHv2	174	Server:
32	4.667753731	172.17.0.3	172.17.0.3	SSHv2	782	Server:
34	4.674094111	172.17.0.3	172.17.0.3	SSHv2	150	Server:

Figura 6: Captura c4 (Ubuntu 20.10) a s1 (Ubuntu 20.10).

De la captura se puede desprender que el cliente y el servidor corresponden al mismo host, ya que, la IP de fuente y de destino es la misma (172.17.0.3) como también, se puede conocer la versión de OpenSSH (v8.3). Se puede notar que aunque el cliente y el servidor se encuentran en el mismos host se realiza el protocolo completo de SSH, es decir, el intercambio de llaves y luego 13 mensajes encriptados.

2.7. Diferencia entre C1 y C2

Entre C1 y C2 existe un salto de versión mayor (de v6.x a v7.x). Ambos contienen la misma serie de pasos: Anuncian su versión, ambos inician el intercambio de llaves, definen un protocolo para enviarla y finalmente el cliente anuncia que tiene una nueva llave; terminan con 13 mensajes cifrados. La única diferencia son los tamaños de los paquetes en cada uno de estos pasos excepto la señal “Client: New Keys”.

2.8. Diferencia entre C2 y C3

Entre C2 y C3 no existe un salto de versión mayor (de v7.3 a v7.7) y en general, todos los paquetes tienen un mismo tamaño excepto el “Client: Key Exchange Init”, además, C3 envía 1 mensaje cifrado más con respecto a C2 (también con respecto a C1 y C4). Esto puede ser circunstancial o no, no se puede inspeccionar el paquete ya que está encriptado.

2.9. Diferencia entre C3 y C4

Entre C3 y C4 sí existe un salto de versión mayor (de v7.7 a v8.3). Siguen invariantes los pasos del protocolo, vuelve a variar el tamaño de los paquetes.

3. Desarrollo (Parte 2)

3.1. Identificación del cliente ssh

Desde la imagen se puede notar que el “Client: Key Exchange Init” tiene exactamente 1578 Bytes. La única versión que coincide con esto es la captura c4 a c4 con OpenSSH v8.3.

- C1 a C4 tiene un “Client: Key Exchange Init” de tamaño 2034 bytes.
- C2 a C4 tiene un “Client: Key Exchange Init” de tamaño 1498 bytes.
- C3 a C4 tiene un “Client: Key Exchange Init” de tamaño 1498 bytes.
- C4 a C4 tiene un “Client: Key Exchange Init” de tamaño 2034 bytes.

Cabe destacar, que desde estos datos se puede intuir que entre versiones mayores se mantienen los tamaños de los mensajes del protocolo SSH.

3.2. Replicación de tráfico (paso por paso)

Para replicar el tráfico se puede compilar desde fuente, para esto se necesita un compilador, una librería para compresión (zlib) y una librería para encriptación (libssl). Antes de compilar basta con modificar el archivo version.h y colocar el string deseado, en este caso OpenSSH_?. A continuación el Dockerfile que genera este contenedor (cliente-modificado):

```

1 FROM ubuntu:20.10
2
3 # Use old-releases.ubuntu.com instead of archive.ubuntu.com or
  security.ubuntu.com
4 RUN sed -i -re 's/([a-z]{2}\.)?archive.ubuntu.com|security.ubuntu.
  com/old-releases.ubuntu.com/g' /etc/apt/sources.list
5
6 RUN apt-get update
7 RUN apt-get install -y \
8     git \
9     autoconf \
10    gcc \
11    zlib1g-dev \
12    libssl-dev \
13    make \
14    wget
15
16 RUN wget https://cdn.openbsd.org/pub/OpenBSD/OpenSSH/portable/
  openssh-8.3p1.tar.gz
17 RUN tar zxvf openssh-8.3p1.tar.gz
18 WORKDIR /openssh-8.3p1
19 RUN sed -i 's/8.3/?/' version.h
20
21 RUN ./configure
22 RUN make
23 RUN make install

```

Generamos el tráfico y comprobamos:

No.	Time	Source	Destination	Protocol	Length	Info
8	0.397408064	172.17.0.4	172.17.0.3	SSHv2	85	Client: Protocol (SSH-2.0-OpenSSH_?)
10	8.425835277	172.17.0.3	172.17.0.4	SSHv2	407	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
12	8.426200555	172.17.0.4	172.17.0.3	SSHv2	1578	Client: Key Exchange Init
14	8.427857293	172.17.0.3	172.17.0.4	SSHv2	1122	Server: Key Exchange Init
15	8.431170768	172.17.0.4	172.17.0.3	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
16	8.437550075	172.17.0.3	172.17.0.4	SSHv2	574	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
18	9.573556976	172.17.0.4	172.17.0.3	SSHv2	82	Client: New Keys
20	9.614710111	172.17.0.4	172.17.0.3	SSHv2	110	Client:
22	9.614808050	172.17.0.3	172.17.0.4	SSHv2	110	Server:
24	9.614948087	172.17.0.4	172.17.0.3	SSHv2	126	Client:
25	9.623787171	172.17.0.3	172.17.0.4	SSHv2	118	Server:
27	10.872811196	172.17.0.4	172.17.0.3	SSHv2	150	Client:
28	10.908424263	172.17.0.3	172.17.0.4	SSHv2	94	Server:
30	10.908784414	172.17.0.4	172.17.0.3	SSHv2	178	Client:
31	10.928468144	172.17.0.3	172.17.0.4	SSHv2	694	Server:
33	10.968761715	172.17.0.4	172.17.0.3	SSHv2	110	Server:
35	10.968972591	172.17.0.4	172.17.0.3	SSHv2	442	Client:
36	10.971719271	172.17.0.3	172.17.0.4	SSHv2	174	Server:
37	10.972054577	172.17.0.3	172.17.0.4	SSHv2	566	Server:
39	10.985894215	172.17.0.3	172.17.0.4	SSHv2	150	Server:

Figura 7: Captura cliente-modificado (Ubuntu 20.10) a s1 (Ubuntu 20.10).

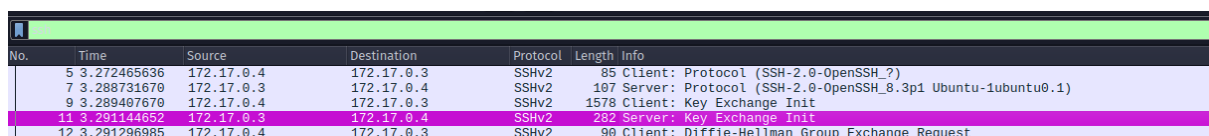
4. Desarrollo (Parte 3)

4.1. Replicación de tráfico (paso por paso)

Gran parte del payload del paquete SSH que envía el servidor hacia el cliente es información sobre los protocolos y cifrados disponibles. Se puede acortar el tamaño del archivo especificando un único parámetro válido. Se genera un Dockerfile llamado “server-modificado” con los siguientes cambios:

```
1 RUN echo "HostKey /etc/ssh/ssh_host_ecdsa_key" >> /etc/ssh/
    sshd_config
2 RUN echo "KexAlgorithms diffie-hellman-group-exchange-sha256" >> /
    etc/ssh/sshd_config
3 RUN echo "MACs hmac-sha2-512" >> /etc/ssh/sshd_config
4 RUN echo "Ciphers aes256-ctr" >> /etc/ssh/sshd_config
```

El servidor SSH ya parte con las condiciones necesarias y se captura el tráfico.



No.	Time	Source	Destination	Protocol	Length	Info
5	3.272465636	172.17.0.4	172.17.0.3	SSHv2	85	Client: Protocol (SSH-2.0-OpenSSH_?)
7	3.288731670	172.17.0.3	172.17.0.4	SSHv2	167	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
9	3.289407670	172.17.0.4	172.17.0.3	SSHv2	1578	Client: Key Exchange Init
11	3.291114052	172.17.0.3	172.17.0.4	SSHv2	262	Server: Key Exchange Init
12	3.291296985	172.17.0.4	172.17.0.3	SSHv2	90	Client: Diffie-Hellman Group Exchange Request

Figura 8: Captura server-modificado con tamaño de paquete reducido.

Conclusiones y comentarios

En el paso informe se puede ver cómo se puede transmitir información de manera sutil manejando las diferencias como versiones de un mismo software que contiene variaciones difíciles de notar. Además se ve cómo esto se puede lograr tanto como del lado del cliente como del servidor.