Informe Laboratorio 5

Sección 01

Jorge Toro Macías
e-mail: jorge.toro1o@mail.udp.cl

Julio de 2024

Índice

Descripción de actividades **3**

1. Desarrollo (Parte 1) **5**
1.1. Códigos de cada Dockerfile 5
1.1.1. C1 5
1.1.2. C2 6
1.1.3. C3 6
1.1.4. C4/S1 6
1.2. Creación de las credenciales para S1 7
1.3. Tráfico generado por C1, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado) 7
1.4. Tráfico generado por C2, detallando tamaño paquetes del flujo y el HASSH respectivo (detallado) 8
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) 10
1.7. Compara la versión de HASSH obtenida con la base de datos para validar si el cliente corresponde al mismo 12
1.8. Tipo de información contenida en cada uno de los paquetes generados en texto plano 13
1.8.1. C1 13
1.8.2. C2 13
1.8.3. C3 14
1.8.4. C4/S1 14
1.9. Diferencia entre C1 y C2 14
1.10. Diferencia entre C2 y C3 14
1.11. Diferencia entre C3 y C4 14

2. Desarrollo (Parte 2) **15**
2.1. Identificación del cliente SSH con versión "?" 15
2.2. Replicación de tráfico al servidor (paso por paso) 15

3. Desarrollo (Parte 3) **15**
3.1. Replicación del KEI con tamaño menor a 300 bytes (paso por paso) 15

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:

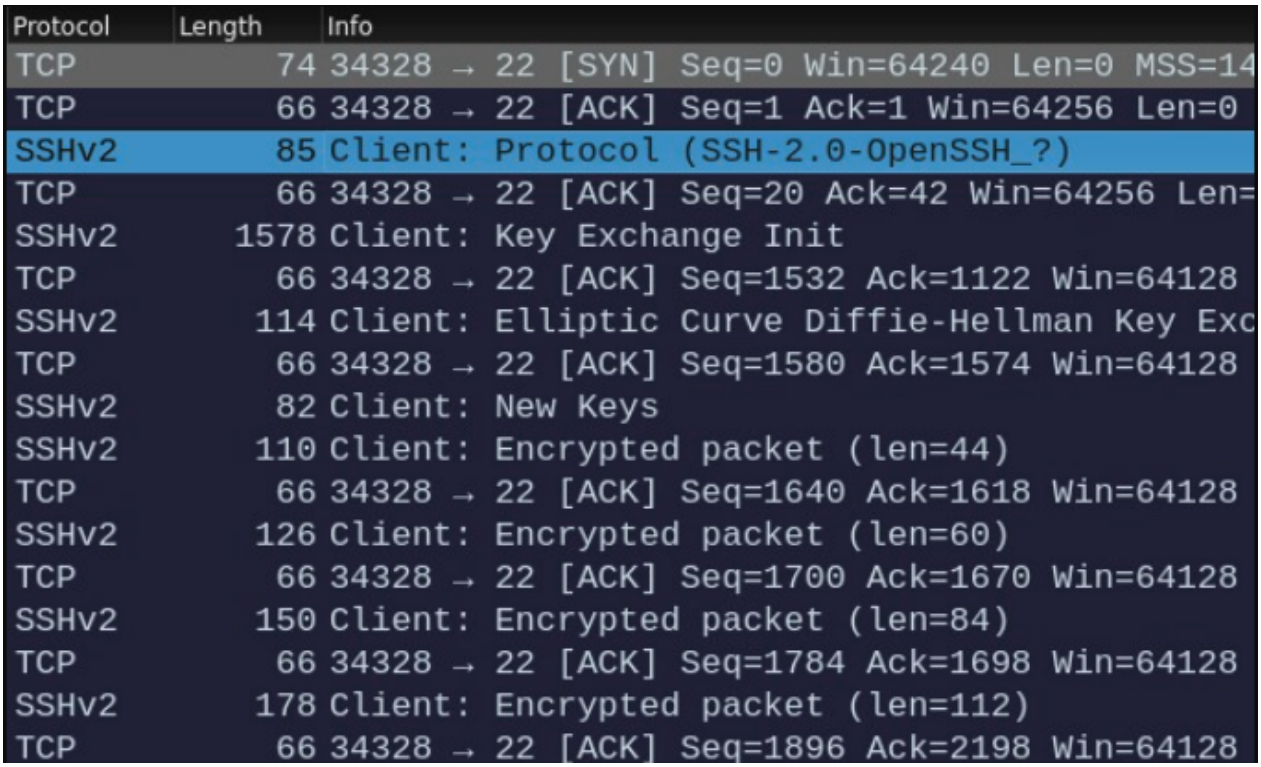
- Crear 4 contenedores en Docker o Podman, donde cada uno tendrá el siguiente SO: Ubuntu 16.10, Ubuntu 18.10, Ubuntu 20.10 y Ubuntu 22.10 a los cuales se llamarán C1, C2, C3 y C4 respectivamente.
  El equipo con Ubuntu 22.10 también será utilizado como S1.
- Para cada uno de ellos, deberá instalar el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

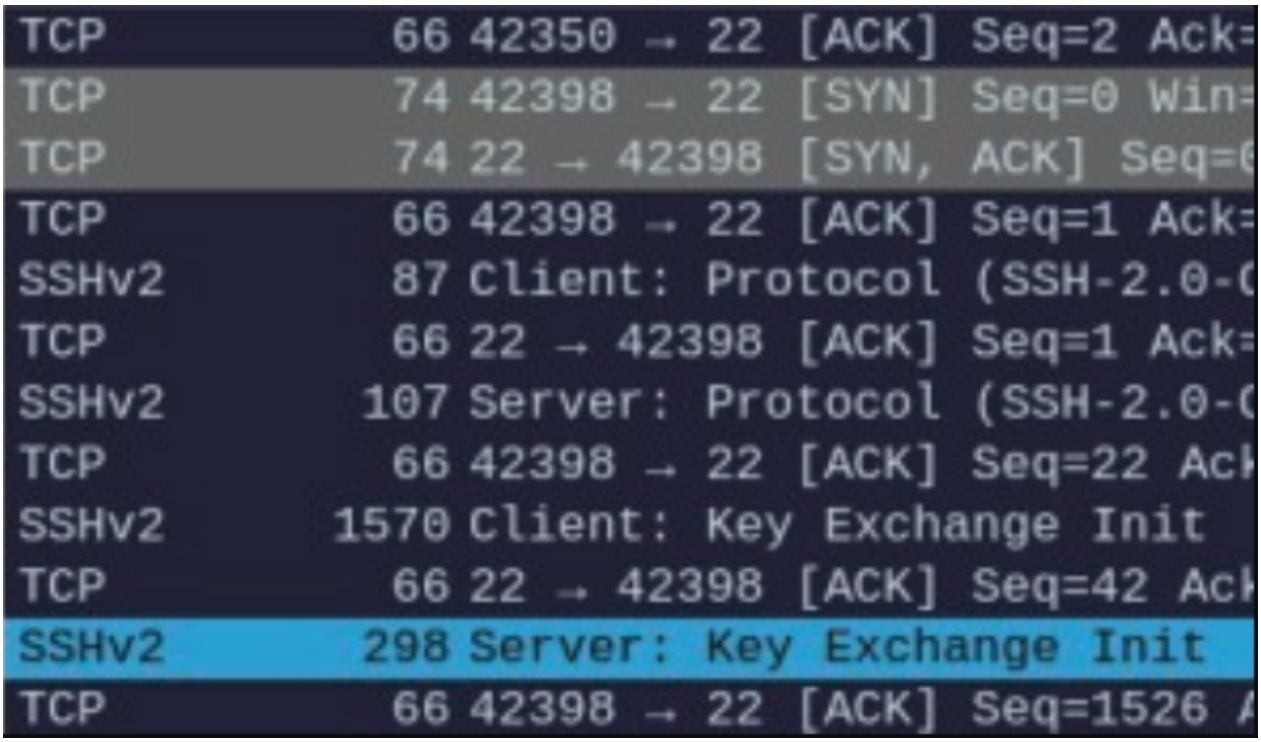


Figura 2: Captura del Key Exchange

1. Desarrollo (Parte 1)

1.1. Códigos de cada Dockerfile

1.1.1. C1

```
FROM ubuntu:16.10

RUN sed -i 's|http://archive.ubuntu.com/ubuntu/|http://old-releases.ubuntu.com/ubuntu/|g' /etc/apt/sources.list && \
    sed -i 's|http://security.ubuntu.com/ubuntu|http://old-releases.ubuntu.com/ubuntu/|g' /etc/apt/sources.list

RUN apt-get update && apt-get install -y openssh-client
```

Figura 3: Código para la imagen de C1

En la figura 3 se puede observar que se especifica la versión 16.10 para el contenedor cliente C1. Además se debe apuntar a los repositorios específicos para versiones antiguas, ya que al ser versiones obsoletas sin vida útil no es posible recuperar los archivos y ficheros con los comandos de manera 'normal'.

1.1.2. C2

```
FROM ubuntu:18.10

RUN sed -i 's|http://archive.ubuntu.com/ubuntu/|http://old-releases.ubuntu.com/ubuntu/|g' /etc/apt/sources.list && \
    sed -i 's|http://security.ubuntu.com/ubuntu|http://old-releases.ubuntu.com/ubuntu/|g' /etc/apt/sources.list

RUN apt-get update && apt-get install -y openssh-client
```

Figura 4: Código para la imagen de C2

En la figura 4 se puede observar que se especifica la versión 18.10 para el contenedor cliente C2. Al igual que para C1 se apunta a los repositorios antiguos de versiones de Ubuntu.

1.1.3. C3

```
FROM ubuntu:20.10

RUN sed -i 's|http://archive.ubuntu.com/ubuntu/|http://old-releases.ubuntu.com/ubuntu/|g' /etc/apt/sources.list && \
    sed -i 's|http://security.ubuntu.com/ubuntu|http://old-releases.ubuntu.com/ubuntu/|g' /etc/apt/sources.list

RUN apt-get update && apt-get install -y openssh-client
```

Figura 5: Código para la imagen de C3

En la figura 5 se puede observar que se especifica la versión 20.10 para el contenedor cliente C3.

1.1.4. C4/S1

```
FROM ubuntu:22.10

RUN sed -i 's|http://archive.ubuntu.com/ubuntu/|http://old-releases.ubuntu.com/ubuntu/|g' /etc/apt/sources.list && \
    sed -i 's|http://security.ubuntu.com/ubuntu|http://old-releases.ubuntu.com/ubuntu/|g' /etc/apt/sources.list

RUN apt-get update && apt-get install -y openssh-client openssh-server

RUN useradd -m -p $(openssl passwd -1 prueba) prueba

RUN mkdir /home/prueba/.ssh && \
    chown -R prueba:prueba /home/prueba/.ssh && \
    echo 'PasswordAuthentication yes' >> /etc/ssh/sshd_config && \
    echo 'PermitRootLogin yes' >> /etc/ssh/sshd_config && \
    service ssh restart
```

Figura 6: Código para la imagen de S1/C4

En la figura 6 se puede observar que se especifica la versión 22.10 para el contenedor servidor/cliente S1/C4. Además se instala OpenSSH tanto para cliente como para servidor, pues este contenedor actuará como ambos.

1.2. Creación de las credenciales para S1

```
FROM ubuntu:22.10

RUN sed -i 's|http://archive.ubuntu.com/ubuntu/|http://old-releases.ubuntu.com/ubuntu/|g' /etc/apt/sources.list && \
    sed -i 's|http://security.ubuntu.com/ubuntu|http://old-releases.ubuntu.com/ubuntu/|g' /etc/apt/sources.list

RUN apt-get update && apt-get install -y openssh-client openssh-server

RUN useradd -m -p $(openssl passwd -1 prueba) prueba

RUN mkdir /home/prueba/.ssh && \
    chown -R prueba:prueba /home/prueba/.ssh && \
    echo 'PasswordAuthentication yes' >> /etc/ssh/sshd_config && \
    echo 'PermitRootLogin yes' >> /etc/ssh/sshd_config && \
    service ssh restart
```

Figura 7: Credenciales para S1

En la figura 7 se puede observar la creación de las credenciales user:'prueba' y password:'prueba' para la conexión al servidor SSH S1. Además se dan los permisos correspondientes para realizar una conexión segura y exitosa.

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

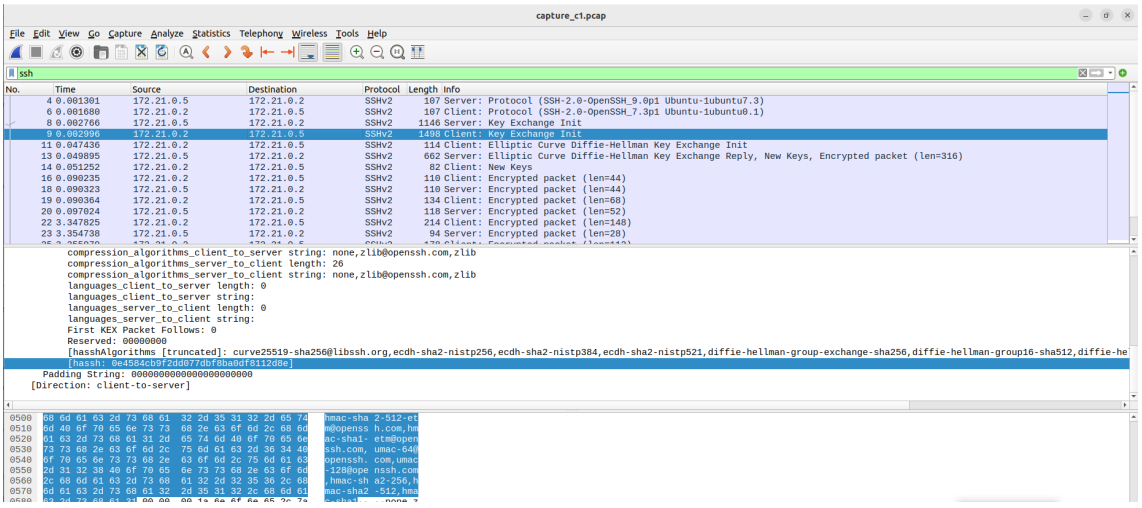


Figura 8: Captura realizada por C1

Figura 9: HASSH correspondiente a C1

Figura 10: Tamaño paquetes C1

En la figura 8 se muestra la captura realizada por la conexión del cliente C1 hacia el servidor S1. En la figura 9 se puede observar el HASSH obtenido para el cliente C1 en el handshake, el cual corresponde a '0e4584cb9f2dd077dbf8ba0df8112d8e'. Este HASSH es calculado aplicando el algoritmo MD5 a la concatenación de todos los algoritmos de encriptación y hash utilizados para la comunicación. En la figura 10 observamos que el tamaño del paquete de Key Exchange Init por parte del cliente es de 1498 bytes.

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

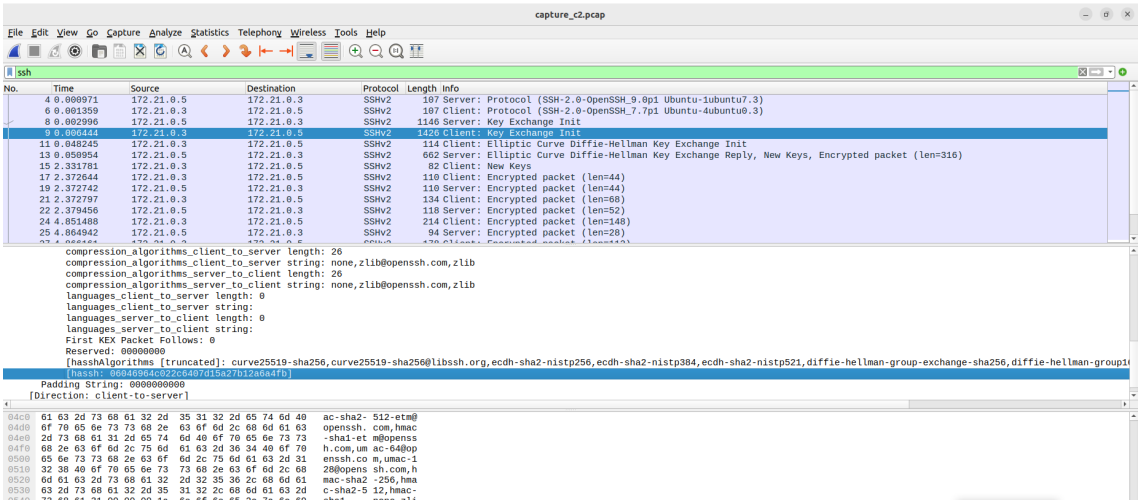


Figura 11: Captura realizada por C2

Figura 12: HASSH correspondiente a C2

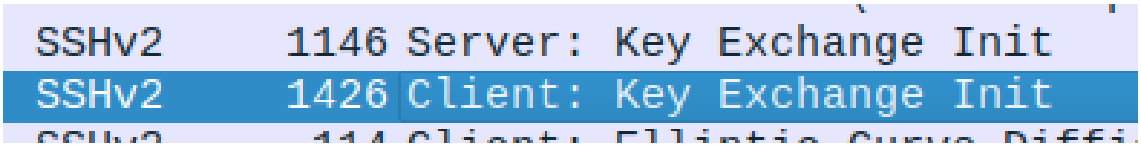


Figura 13: Tamaño paquetes C2

En la figura 11 se muestra la captura realizada por la conexión del cliente C2 hacia el servidor S1. En la figura 12 se puede observar el HASSH obtenido para el cliente C2 en el handshake, el cual corresponde a '0604606964c022c6407d15a27b12a6a4fb'. Este HASSH es calculado aplicando el algoritmo MD5 a la concatenación de todos los algoritmos de encriptación y hash utilizados para la comunicación. En la figura 13 observamos que el tamaño del paquete de Key Exchange Init por parte del cliente C2 es de 1426 bytes. Este tamaño es menor que el del paquete generado por el cliente C1.

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

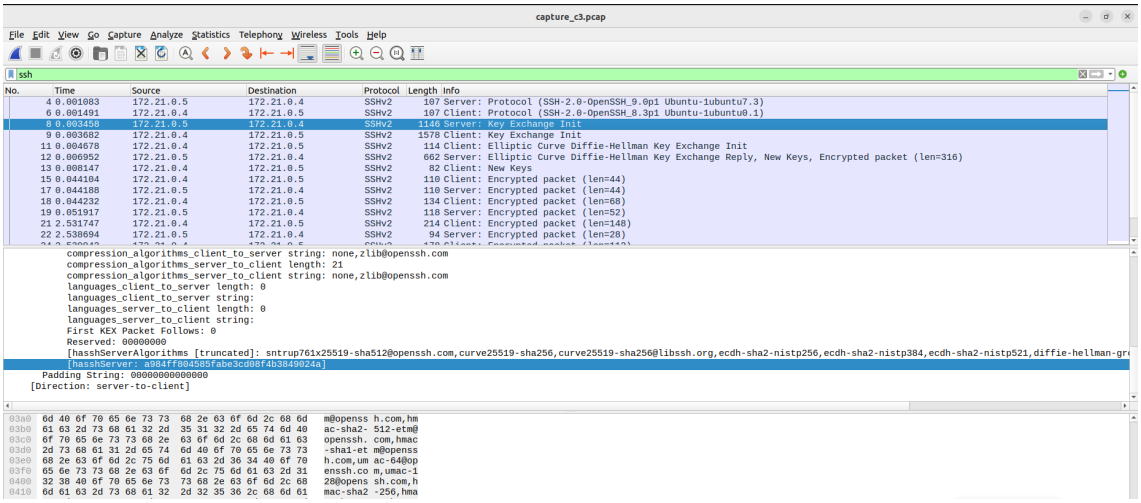


Figura 14: Captura realizada por C3

Figura 15: HASSH correspondiente a C3

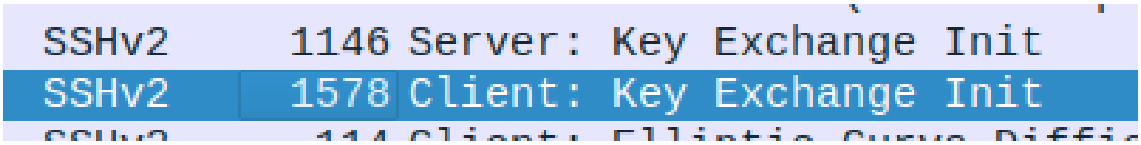


Figura 16: Tamaño paquetes C3

En la figura 14 se muestra la captura realizada por la conexión del cliente C3 hacia el servidor S1. En la figura 15 se puede observar el HASSH obtenido para el cliente C3 en el handshake, el cual corresponde a 'ae8bd7dd09970555aa4c6ed22adbbf56'. Este HASSH es calculado aplicando el algoritmo MD5 a la concatenación de todos los algoritmos de encriptación y hash utilizados para la comunicación. En la figura 16 observamos que el tamaño del paquete de Key Exchange Init por parte del cliente C3 es de 1578 bytes. Este tamaño es mayor que el de los paquetes generads por los clientes C1 y C2.

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

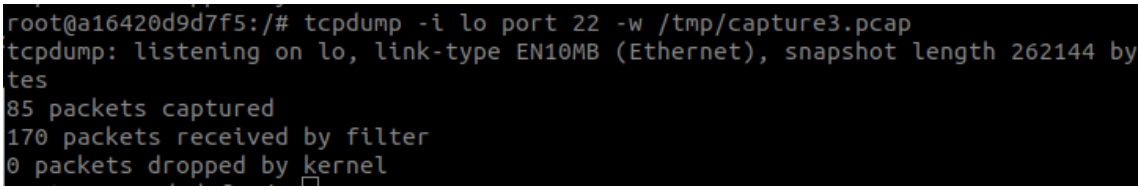


Figura 17: Captura realizada en la interfaz de loopback

En la figura 17 se puede observar el comando con el ajuste realizado para hacer la captura en la interfaz de loopback.

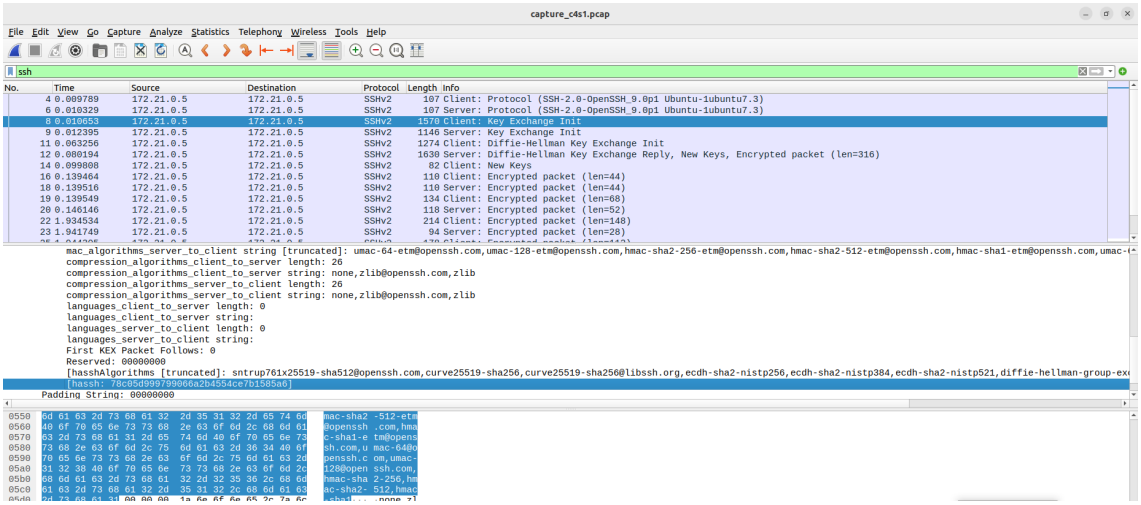


Figura 18: Captura realizada por C4

Figura 19: HASSH correspondiente a C4

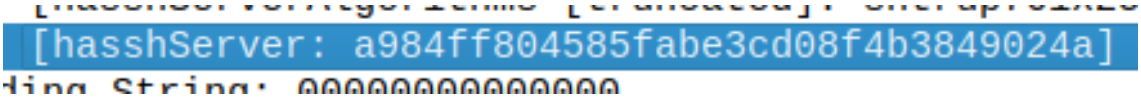


Figura 20: HASSH correspondiente a S1

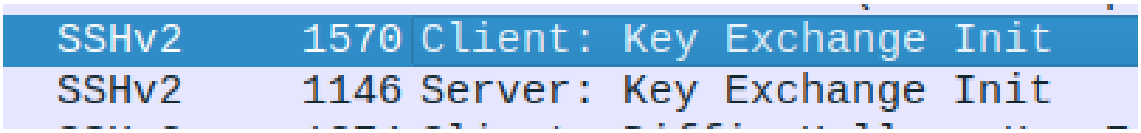


Figura 21: Tamaño paquetes C4

En la figura 18 se muestra la captura realizada por la conexión del cliente C4 hacia sí mismo como servidor S1. En la figura 19 se puede observar el HASSH obtenido para el cliente C4 en el handshake, el cual corresponde a '78c05d999799066a2b4554ce7b1585a6'. Mientras que en la figura 20 podemos observar el HASSH para el servidor, el cual es 'a984ff804585fabe3cd08f4b3849024a'.Estos HASSH son calculados aplicando el algoritmo MD5 a la concatenación de todos los algoritmos de encriptación y hash utilizados para la comunicación. En la figura 21 observamos que el tamaño del paquete de Key Exchange Init por parte del cliente C4 es de 1570 bytes. Este tamaño es mayor que el de los paquetes generads por los clientes C1 y C2 pero menor que el del cliente C3.

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

Figura 22: Versión C1.

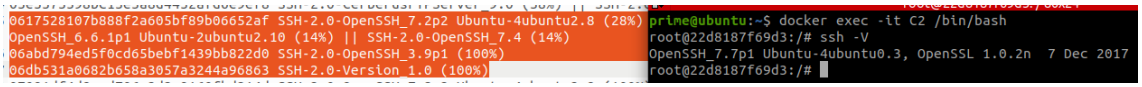


Figura 23: Versión C2.

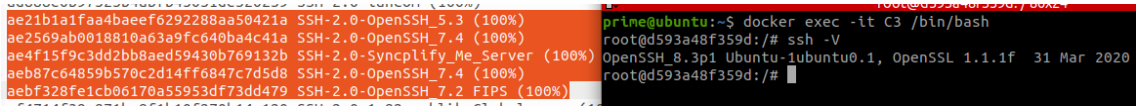


Figura 24: Versión C3.

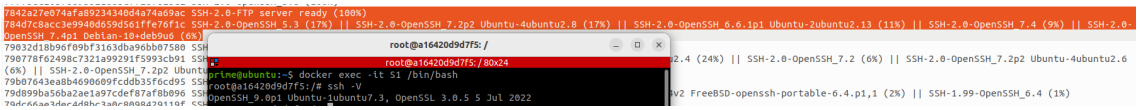


Figura 25: Versión C4.

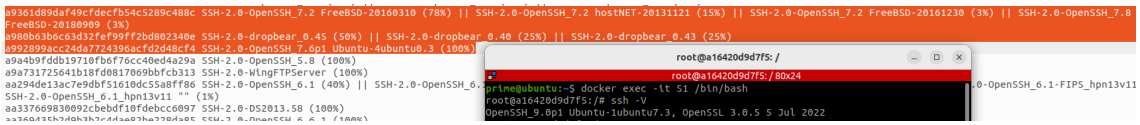


Figura 26: Versión S1.

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

1.8.1. C1

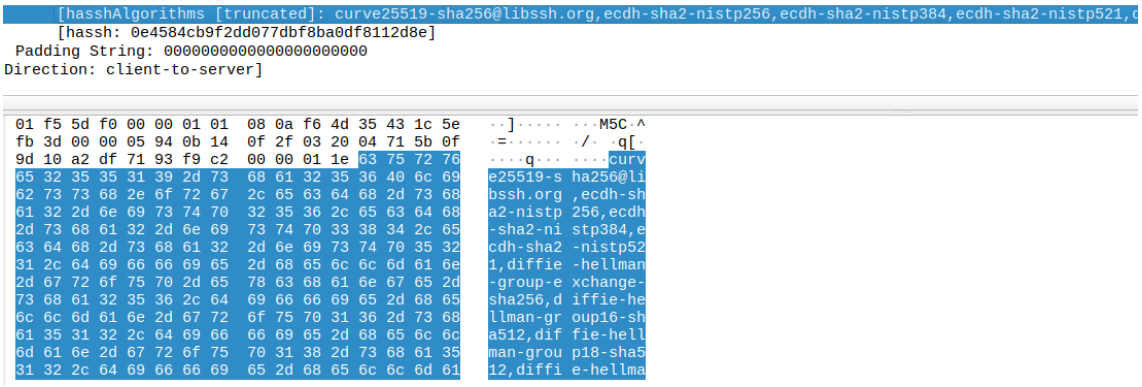


Figura 27: Contenido C1.

1.8.2. C2

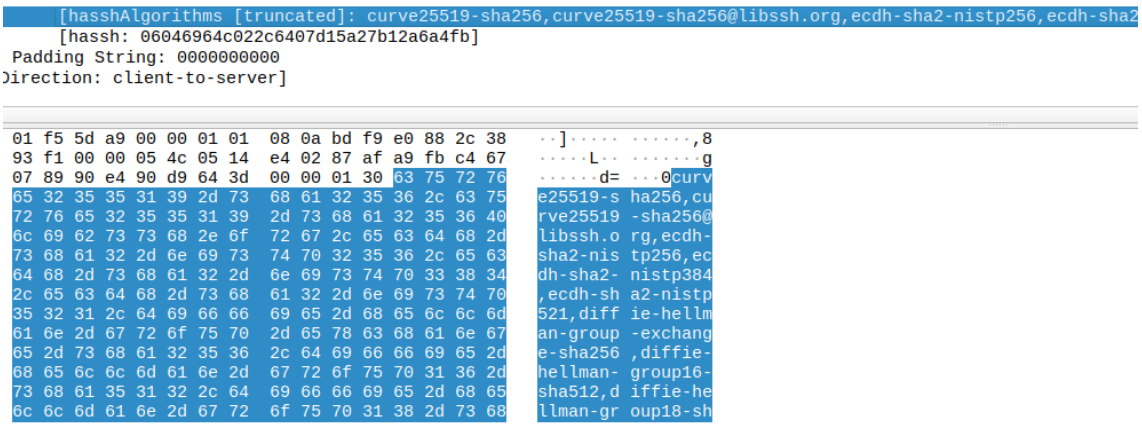


Figura 28: Contenido C2.

1.8.3. C3

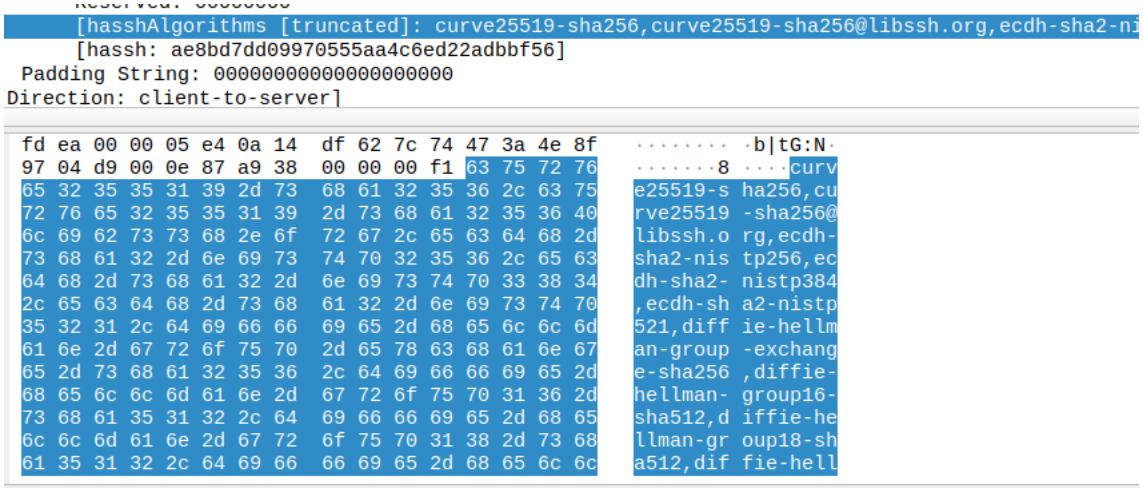


Figura 29: Contenido C3.

1.8.4. C4/S1

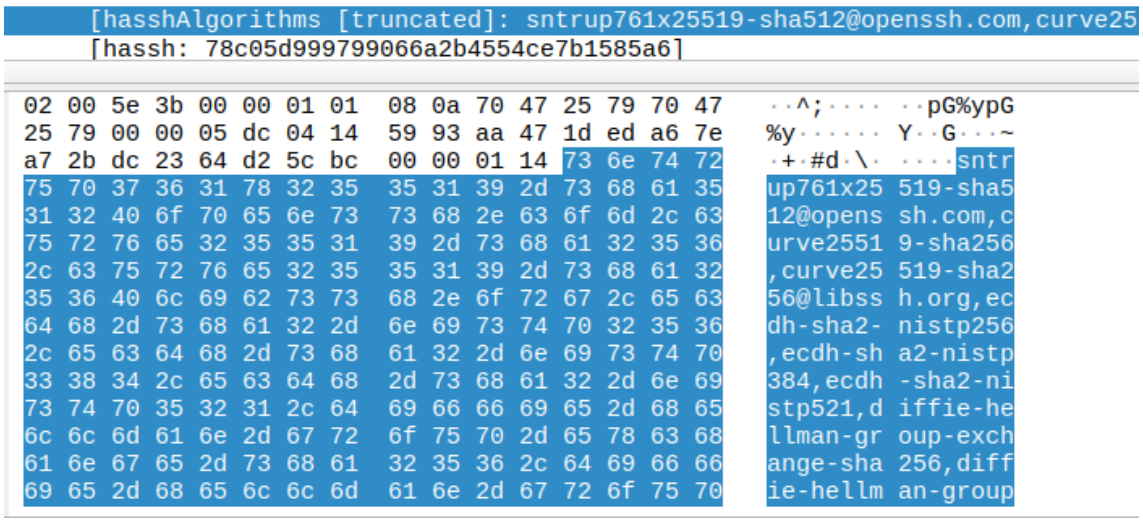


Figura 30: Contenido C4/S1.

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)

Conclusiones y comentarios

Issues