

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 → 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.

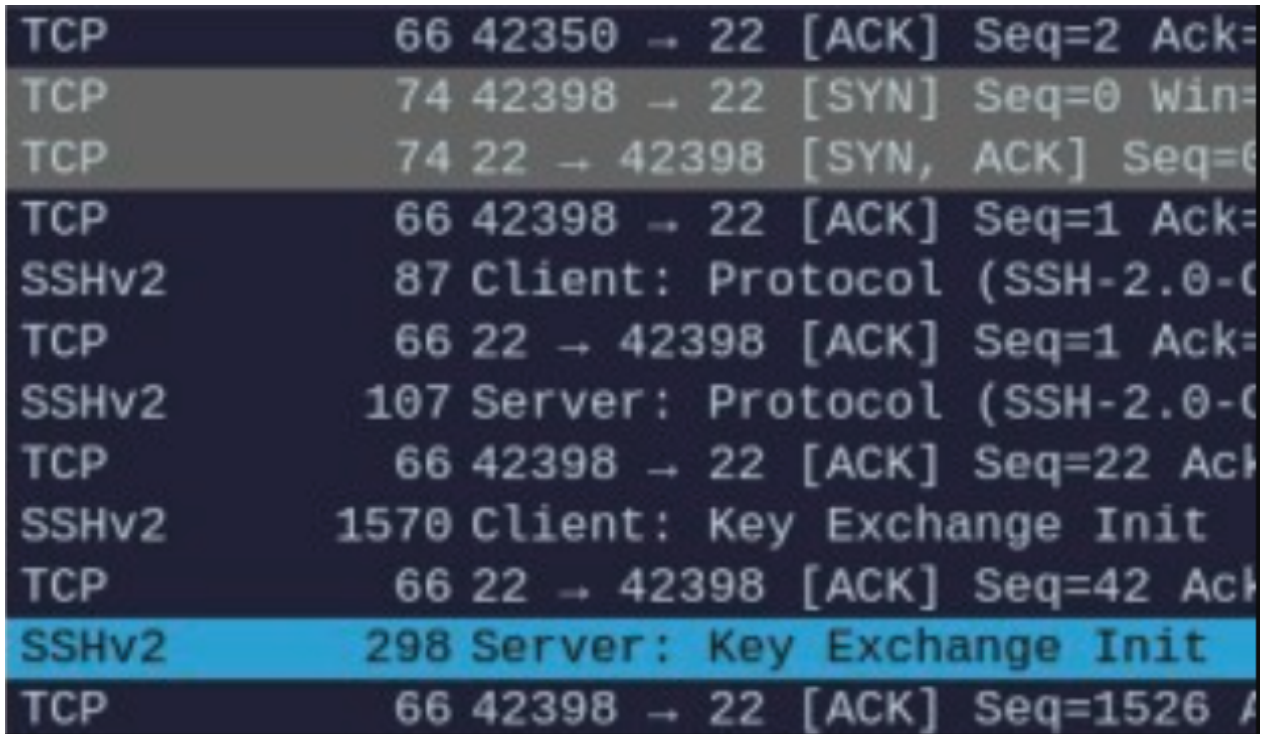


Figura 2: Captura del Key Exchange

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)

No.	Time	Source	Destination	Protocol	Length	Info
4	0.001391	172.21.0.5	172.21.0.2	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-lubuntu7.3)
6	0.001688	172.21.0.2	172.21.0.5	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.3p1 Ubuntu-lubuntu0.1)
8	0.002766	172.21.0.5	172.21.0.2	SSHv2	1146	Server: Key Exchange Init
11	0.0047436	172.21.0.2	172.21.0.5	SSHv2	1146	Client: Key Exchange Init
13	0.049895	172.21.0.5	172.21.0.2	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=310)
14	0.051252	172.21.0.2	172.21.0.5	SSHv2	82	Client: New Keys
16	0.090235	172.21.0.2	172.21.0.5	SSHv2	118	Client: Encrypted packet (len=44)
18	0.090323	172.21.0.5	172.21.0.2	SSHv2	118	Server: Encrypted packet (len=44)
19	0.090364	172.21.0.2	172.21.0.5	SSHv2	134	Client: Encrypted packet (len=68)
20	0.097824	172.21.0.5	172.21.0.2	SSHv2	118	Server: Encrypted packet (len=52)
22	3.347825	172.21.0.2	172.21.0.5	SSHv2	214	Client: Encrypted packet (len=148)
23	3.354738	172.21.0.5	172.21.0.2	SSHv2	94	Server: Encrypted packet (len=28)

compression_algorithms_client_to_server string: none,zlib@openssh.com,zlib
compression_algorithms_server_to_client length: 26
compression_algorithms_server_to_client string: none,zlib@openssh.com,zlib
languages_client_to_server length: 0
languages_client_to_server string:
languages_server_to_client length: 0
languages_server_to_client string:
First KEX Packet Follows: 0
Reserved: 00000000
[hashsh: 0e4584cb9f2dd077dbf8ba0df8112d8e]
Padding String: 000000000000000000000000
[Direction: client-to-server]

Figura 8: Captura realizada por C1

```

[hassh: 0e4584cb9f2dd077dbf8ba0df8112d8e]
padding String: 000000000000000000000000

```

Figura 9: HASSH correspondiente a C1

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

1 DESARROLLO (PARTE 1)

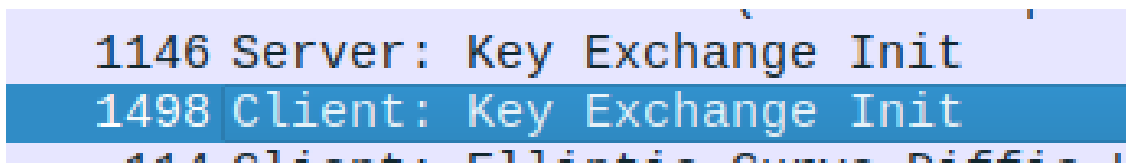


Figura 10: Tamaño paquetes C1

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)

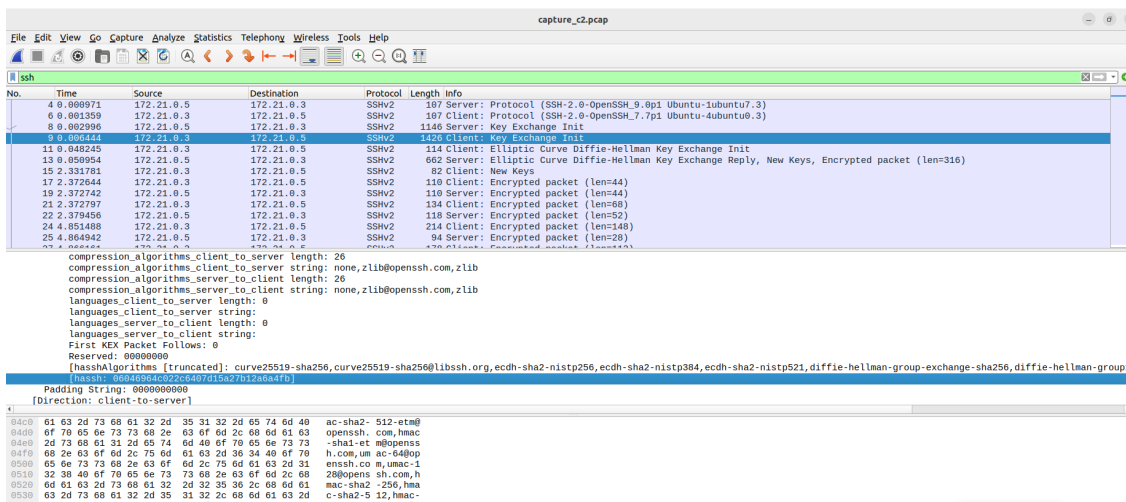


Figura 11: Captura realizada por C2



Figura 12: HASSH correspondiente a C2

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

1 DESARROLLO (PARTE 1)

SSHv2	1146	Server: Key Exchange Init
SSHv2	1426	Client: Key Exchange Init
SSHv2	114	Client: Elliptic Curve Diffie

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)

No.	Time	Source	Destination	Protocol	Length	Info
4	0.001003	172.21.0.5	172.21.0.4	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_9.0p1 Ubuntu-ubuntu7.3)
6	0.001491	172.21.0.4	172.21.0.5	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-ubuntu0.1)
8	0.003458	172.21.0.5	172.21.0.4	SSHv2	1146	Server: Key Exchange Init
9	0.003652	172.21.0.4	172.21.0.5	SSHv2	1578	Client: Key Exchange Init
11	0.004078	172.21.0.4	172.21.0.5	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
12	0.006952	172.21.0.5	172.21.0.4	SSHv2	662	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
13	0.008147	172.21.0.4	172.21.0.5	SSHv2	82	Client: New Keys
15	0.044104	172.21.0.4	172.21.0.5	SSHv2	110	Client: Encrypted packet (len=44)
17	0.044188	172.21.0.5	172.21.0.4	SSHv2	110	Server: Encrypted packet (len=44)
18	0.044232	172.21.0.4	172.21.0.5	SSHv2	134	Client: Encrypted packet (len=68)
19	0.051917	172.21.0.5	172.21.0.4	SSHv2	110	Server: Encrypted packet (len=52)
21	2.531747	172.21.0.4	172.21.0.5	SSHv2	214	Client: Encrypted packet (len=148)
22	2.538694	172.21.0.5	172.21.0.4	SSHv2	94	Server: Encrypted packet (len=28)

compression_algorithms_client_to_server string: none,zlib@openssh.com
compression_algorithms_server_to_client length: 21
compression_algorithms_server_to_client string: none,zlib@openssh.com
languages_client_to_server length: 0
languages_client_to_server string:
languages_server_to_client length: 0
languages_server_to_client string:
First KEX Packet Follows: 0
Reserved: 00000000
[hashServerAlgorithms [truncated]: sntrup761x25519-sha512@openssh.com,curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-gr...
[hashServer]: a904ff004505fabe3cd08f4b3849024a]
Padding String: 0000000000000000
[Direction: server-to-client]

63b0	6d 46 6f 70 65 6e 73 73 68 2e 63 6f 6d 2c 68 6d	m@openssh.h.com,hm
63b0	61 63 2d 73 68 61 32 2d 35 31 32 2d 65 74 6d 40	ac-sha2-512-etc@
63b0	6f 70 65 6e 73 73 68 2e 63 6f 6d 2c 68 6d 61 63	openssh.com,hmac
63b0	2d 73 68 61 31 2d 65 74 6d 40 6f 70 65 6e 73 73	-sha2-etc m@openssh
63b0	68 2e 63 6f 6d 2c 75 6d 61 63 2d 36 34 40 6f 70	h.com,um ac-64@op
63f0	65 6e 73 73 68 2e 63 6f 6d 2c 75 6d 61 63 2d 31	enssh.co n,umac-1
8400	32 38 40 6f 70 65 6e 73 73 68 2e 63 6f 6d 2c 68	28@openssh.com,h
8410	6d 61 63 2d 73 68 61 32 2d 32 35 36 2c 68 6d 61	mac-sha2-256,hma

Figura 14: Captura realizada por C3

[hassh: ae8bd7dd09970555aa4c6ed22adbbf56]

Figura 15: HASSH correspondiente a C3

SSHv2	1146	Server: Key Exchange Init
SSHv2	1578	Client: Key Exchange Init
SSHv2	1144	Client: Elliptic Curve Diffie

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 generados por los clientes C1 y C2.

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

```
root@a16420d9d7f5:/# tcpdump -i lo port 22 -w /tmp/capture3.pcap
tcpdump: listening on lo, link-type EN10MB (Ethernet), snapshot length 262144 bytes
85 packets captured
170 packets received by filter
0 packets dropped by kernel
```

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.

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

1 DESARROLLO (PARTE 1)

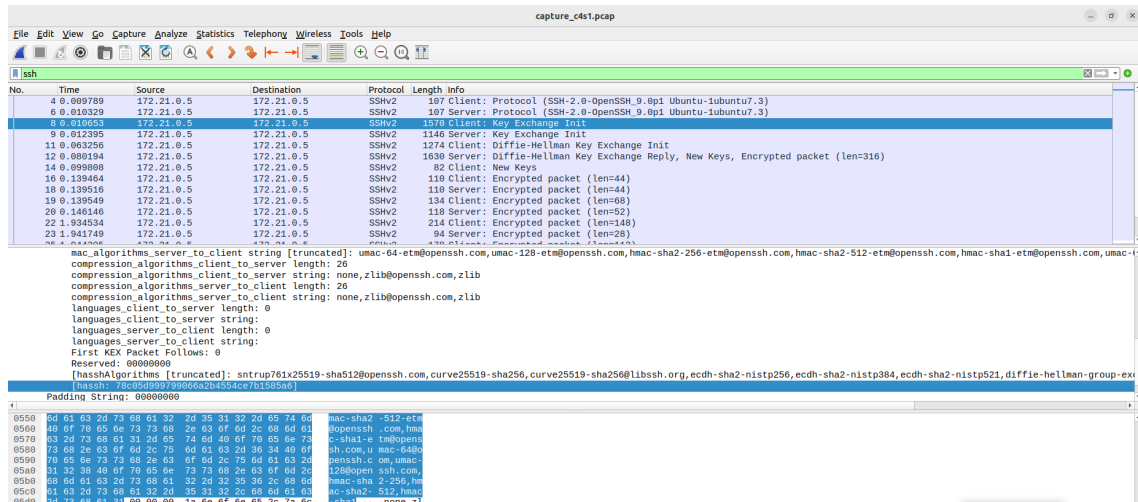


Figura 18: Captura realizada por C4



Figura 19: HASSH correspondiente a C4

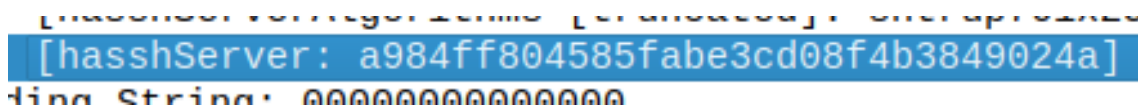


Figura 20: HASSH correspondiente a S1

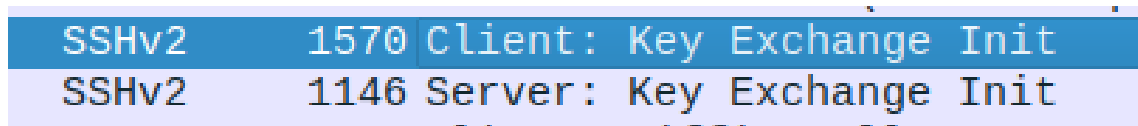


Figura 21: Tamaño paquetes C4

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 generados 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

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

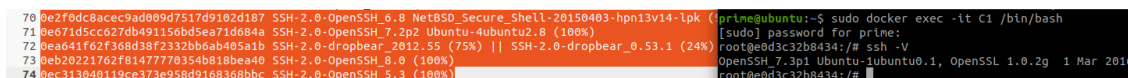


Figura 22: Versión C1.

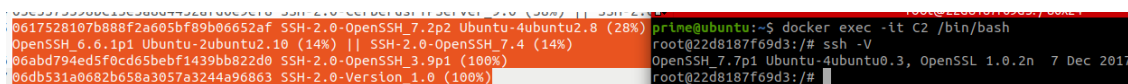


Figura 23: Versión C2.

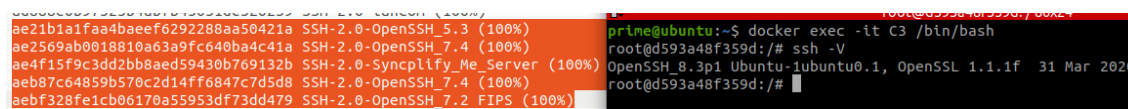


Figura 24: Versión C3.

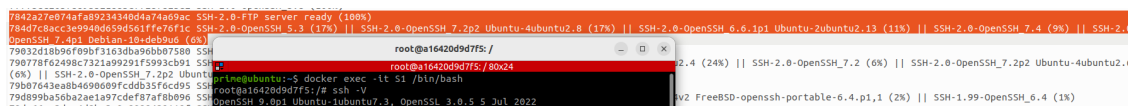


Figura 25: Versión C4.

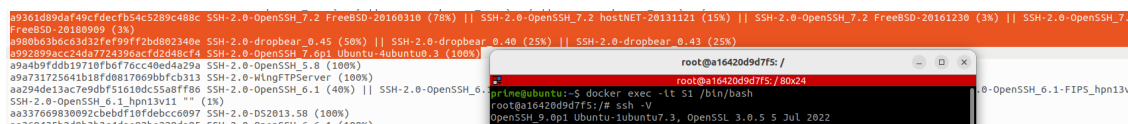


Figura 26: Versión S1.

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

1.8.1. C1

```
[hasshAgorithms [truncated]: curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,  
[hassh: 0e4584cb9f2dd077dbf8ba0df8112d8e]  
Padding String: 00000000000000000000  
Direction: client-to-server]
```

```
01 f5 5d f0 00 00 01 01 08 0a f6 4d 35 43 1c 5e ..].....M5C-A  
fb 3d 00 00 05 94 0b 14 0f 2f 03 20 04 71 5b 0f .-...../.qf.  
9d 10 a2 df 71 93 f9 c2 00 00 01 1e 63 75 72 76 .....q.....curv  
65 32 35 35 31 39 2d 73 68 61 32 35 36 40 6c 69 e25519-s ha256@li  
62 73 73 68 2e 6f 72 67 2c 65 63 64 68 2d 73 68 bssh.org ,ecdh-sh  
61 32 2d 6e 69 73 74 70 32 35 36 2c 65 63 64 68 a2-nistp 256,ecdh  
2d 73 68 61 32 2d 6e 69 73 74 70 33 38 34 2c 65 -sha2-ni stp384,e  
63 64 68 2d 73 68 61 32 2d 6e 69 73 74 70 35 32 cdh-sha2 -nistp52  
31 2c 64 69 66 66 69 65 2d 68 65 6c 6c 6d 61 6e 1,diffie -hellman  
2d 67 72 6f 75 70 2d 65 78 63 68 61 6e 67 65 2d -group-e xchange-  
73 68 61 32 35 36 2c 64 69 66 66 69 65 2d 68 65 sha256,d iffie-he  
6c 6c 6d 61 6e 2d 67 72 6f 75 70 31 36 2d 73 68 llman-gr oup16-sh  
61 35 31 32 2c 64 69 66 66 69 65 2d 68 65 6c 6c a512,dif fie-hell  
6d 61 6e 2d 67 72 6f 75 70 31 38 2d 73 68 61 35 man-grou p18-sha5  
31 32 2c 64 69 66 66 69 65 2d 68 65 6c 6c 6d 61 12,diffi e-hellma
```

Figura 27: Contenido C1.

1.8.2. C2

```
[hasshAlgorithms [truncated]: curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp521]
[hassh: 06046964c022cc6407d15a27b12a6a4fb]
Padding String: 0000000000
Direction: client-to-server]

01 f5 5d a9 00 00 01 01 08 0a bd f9 e0 88 2c 38 ..].....,8
93 f1 00 00 05 4c 05 14 e4 02 87 af a9 fb c4 67 .....L.....g
07 89 90 e4 90 d9 64 3d 00 00 01 30 63 75 72 76 .....d==...0curv
65 32 35 35 31 39 2d 73 68 61 32 35 36 2c 63 75 e25519-s ha256,cu
72 76 65 32 35 31 39 2d 73 68 61 32 35 36 40 rve25519 -sha256@
6c 69 62 73 68 2e 6f 72 67 2c 65 63 64 68 2d libssh.o rg,ecdh-
73 68 61 32 2d 69 73 74 70 32 35 36 2c 65 63 sha2-nis tp256,ec
64 68 2d 73 68 61 32 2d 6e 69 73 74 70 33 38 34 dh-sha2- nistp384
2c 65 63 64 68 2d 73 68 61 32 2d 6e 69 73 74 70 ,ecdh-sh a2-nistp
35 32 31 2c 64 69 66 66 69 65 2d 68 65 6c 6c 6d 521,diff ie-hellm
61 6e 2d 67 72 6f 75 70 2d 65 78 63 68 61 6e 67 an-group -exchang
65 2d 73 68 61 32 35 36 2c 64 69 66 66 69 65 2d e-sha256 ,diffie-
68 65 6c 6c 6d 61 6e 2d 67 72 6f 75 70 31 36 2d hellman- group16-
73 68 61 35 31 32 2c 64 69 66 66 69 65 2d 68 65 sha512,d iffie-he
6c 6c 6d 61 6e 2d 67 72 6f 75 70 31 38 2d 73 68 llman-gr oup18-sh
```

Figura 28: Contenido C2.

1.8.3. C3

```

[hasshAlgorithms [truncated]: curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-ni
[hassh: ae8bd7dd09970555aa4c6ed22adbbf56]
Padding String: 00000000000000000000
Direction: client-to-server]
fd ea 00 00 05 e4 0a 14 df 62 7c 74 47 3a 4e 8f .....b|tG:N.
97 04 d9 00 0e 87 a9 38 00 00 00 f1 63 75 72 76 .....8....curv
65 32 35 35 31 39 2d 73 68 61 32 35 36 2c 63 75 e25519-s ha256,cu
72 76 65 32 35 35 31 39 2d 73 68 61 32 35 36 40 rve25519 -sha256@
6c 69 62 73 73 68 2e 6f 72 67 2c 65 63 64 68 2d libssh.o rg,ecdh-
73 68 61 32 2d 6e 69 73 74 70 32 35 36 2c 65 63 sha2-nis tp256,ec
64 68 2d 73 68 61 32 2d 6e 69 73 74 70 33 38 34 dh-sha2- nistp384
2c 65 63 64 68 2d 73 68 61 32 2d 6e 69 73 74 70 ,ecdh-sh a2-nistp
35 32 31 2c 64 69 66 66 69 65 2d 68 65 6c 6c 6d 521,diff ie-hellm
61 6e 2d 67 72 6f 75 70 2d 65 78 63 68 61 6e 67 an-group -exchang
65 2d 73 68 61 32 35 36 2c 64 69 66 66 69 65 2d e-sha256 ,diffie-
68 65 6c 6c 6d 61 6e 2d 67 72 6f 75 70 31 36 2d hellman- group16-
73 68 61 35 31 32 2c 64 69 66 66 69 65 2d 68 65 sha512,d iffie-he
6c 6c 6d 61 6e 2d 67 72 6f 75 70 31 38 2d 73 68 llman-gr oup18-sh
61 35 31 32 2c 64 69 66 66 69 65 2d 68 65 6c 6c a512,dif fie-hell

```

Figura 29: Contenido C3.

1.8.4. C4/S1

```

[hasshAlgorithms [truncated]: sntrup761x25519-sha512@openssh.com,curve25
[hassh: 78c05d999799066a2b4554ce7b1585a6]
02 00 5e 3b 00 00 01 01 08 0a 70 47 25 79 70 47 ..^;....pG%ypG
25 79 00 00 05 dc 04 14 59 93 aa 47 1d ed a6 7e %y.....Y..G...~
a7 2b dc 23 64 d2 5c bc 00 00 01 14 73 6e 74 72 ..+.#d.\....sntr
75 70 37 36 31 78 32 35 35 31 39 2d 73 68 61 35 up761x25 519-sha5
31 32 40 6f 70 65 6e 73 73 68 2e 63 6f 6d 2c 63 12@opens sh.com,c
75 72 76 65 32 35 35 31 39 2d 73 68 61 32 35 36 urve2551 9-sha256
2c 63 75 72 76 65 32 35 35 31 39 2d 73 68 61 32 ,curve25 519-sha2
35 36 40 6c 69 62 73 73 68 2e 6f 72 67 2c 65 63 56@libss h.org,ec
64 68 2d 73 68 61 32 2d 6e 69 73 74 70 32 35 36 dh-sha2- nistp256
2c 65 63 64 68 2d 73 68 61 32 2d 6e 69 73 74 70 ,ecdh-sh a2-nistp
33 38 34 2c 65 63 64 68 2d 73 68 61 32 2d 6e 69 384,ecdh -sha2-ni
73 74 70 35 32 31 2c 64 69 66 66 69 65 2d 68 65 stp521,d iffie-he
6c 6c 6d 61 6e 2d 67 72 6f 75 70 2d 65 78 63 68 llman-gr oup-exch
61 6e 67 65 2d 73 68 61 32 35 36 2c 64 69 66 66 ange-sha 256,diff
69 65 2d 68 65 6c 6c 6d 61 6e 2d 67 72 6f 75 70 ie-hellm an-group

```

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