

Informe Laboratorio 5

Sección 2

Santiago Larraín Morales
e-mail: Santiago.Larrain@mail.udp.cl

Noviembre 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	5
2.2. Creación de las credenciales para S1	6
2.3. Tráfico generado por C1 (detallado)	11
2.4. Tráfico generado por C2 (detallado)	11
2.5. Tráfico generado por C3 (detallado)	12
2.6. Tráfico generado por C4 (4 (iface lo) (detallado)	13
2.7. Diferencia entre C1 y C2	14
2.8. Diferencia entre C2 y C3	15
2.9. Diferencia entre C3 y C4	15
3. Desarrollo (Parte 2)	16
3.1. Identificación del cliente ssh	16
3.2. Replicación de tráfico (paso por paso)	16
4. Desarrollo (Parte 3)	18
4.1. Replicación de tráfico (paso a paso)	18
5. Conclusiones y comentarios	19
6. Referencias	19

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.

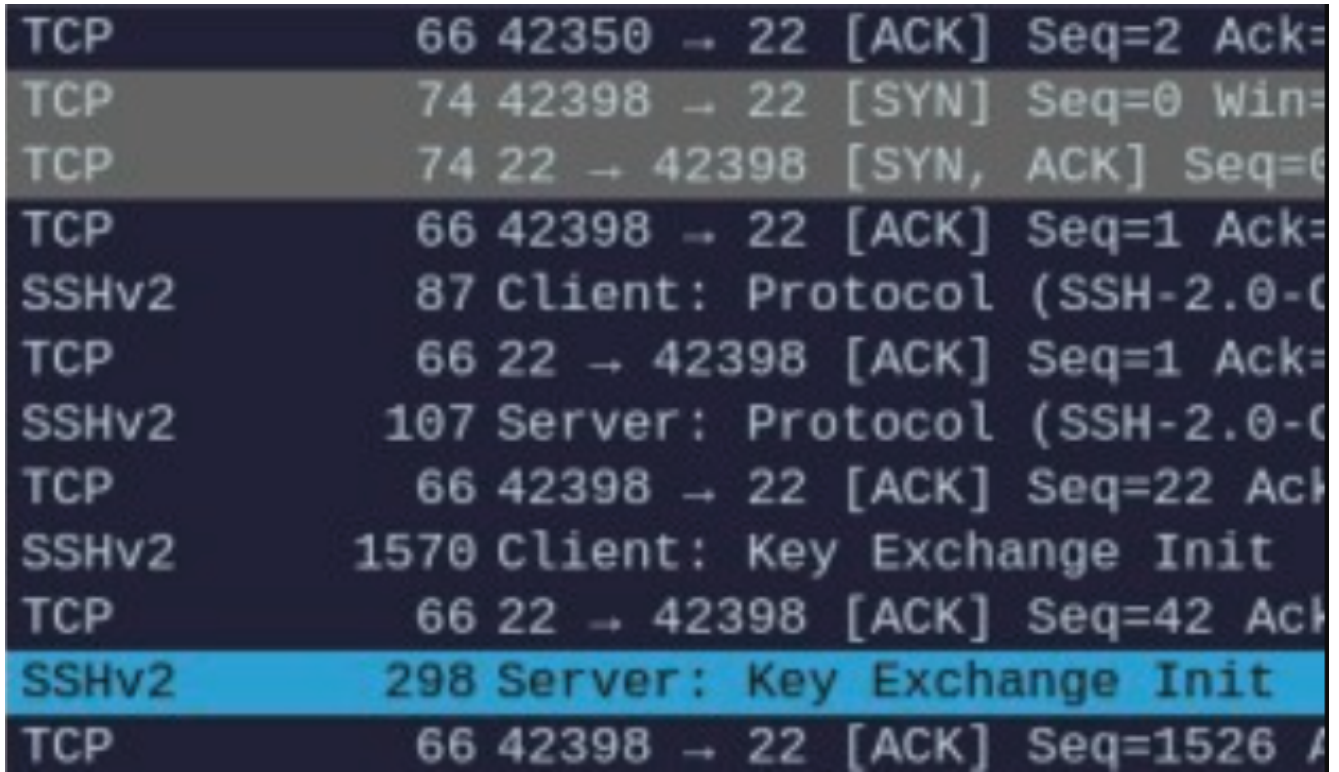
2. Para poder identificar que el usuario efectivamente es el informante, éste utilizará una versión única de cliente. ¿Con qué cliente SSH se habrá generado el siguiente tráfico?

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=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	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

2.1.1. C1

```
FROM ubuntu:14.10

RUN sed -i 's/archive/old-releases/g' /etc/apt/sources.list
RUN sed '/^deb.*security.ubuntu.com/s/^/#/' /etc/apt/sources.list

RUN apt update && apt install -y sudo net-tools openssh-client
```

Listing 1: Dockerfile para el cliente 1 (Ubuntu 14.10).

2.1.2. C2

```
FROM ubuntu:16.10

RUN sed -i 's/archive/old-releases/g' /etc/apt/sources.list
RUN sed '/^deb.*security.ubuntu.com/s/^/#/' /etc/apt/sources.list

RUN apt update && apt install -y sudo net-tools openssh-client
```

Listing 2: Dockerfile para el cliente 2 (Ubuntu 16.10).

2.1.3. C3

```
FROM ubuntu:18.10

RUN sed -i 's/archive/old-releases/g' /etc/apt/sources.list
RUN sed '/^deb.*security.ubuntu.com/s/^/#/' /etc/apt/sources.list

RUN apt update && apt install -y sudo net-tools openssh-client
```

Listing 3: Dockerfile para el cliente 3 (Ubuntu 18.10).

2.1.4. C4/S1

Siguiendo las instrucciones del profesor de laboratorio, se elaboró un Dockerfile destinado a configurar el contenedor C4 para desempeñarse tanto como cliente como servidor. Para lograr

este propósito, se instaló tanto el cliente como el servidor de OpenSSH en dicho contenedor.

```
FROM ubuntu:20.10

RUN sed -i 's/archive/old-releases/g' /etc/apt/sources.list
RUN sed -i '/^deb.*security.ubuntu.com/s/^/#/' /etc/apt/sources.list

# Instalar cliente SSH
RUN apt update && apt install -y sudo net-tools openssh-client

# Instalar servidor SSH
RUN apt-get install -y openssh-server

# Crear usuario test con contraseña test
RUN useradd -m -s /bin/bash test && echo "test:test" | chpasswd

# Exponer el puerto 22 para SSH (puedes cambiarlo segun tus necesidades)
EXPOSE 22

# Iniciar el servidor SSH al arrancar el contenedor
CMD ["/usr/sbin/sshd", "-D"]
```

Listing 4: Dockerfile para el cliente 4 y servidor 1 (Ubuntu 20.10).

2.2. Creación de las credenciales para S1

Para crear las credenciales a utilizar durante la experiencia, se realizó a través del Dockerfile del C4. Se empleó el siguiente comando para crear un usuario:

```
RUN useradd -m -s /bin/bash test && echo "test:test" | chpasswd
```

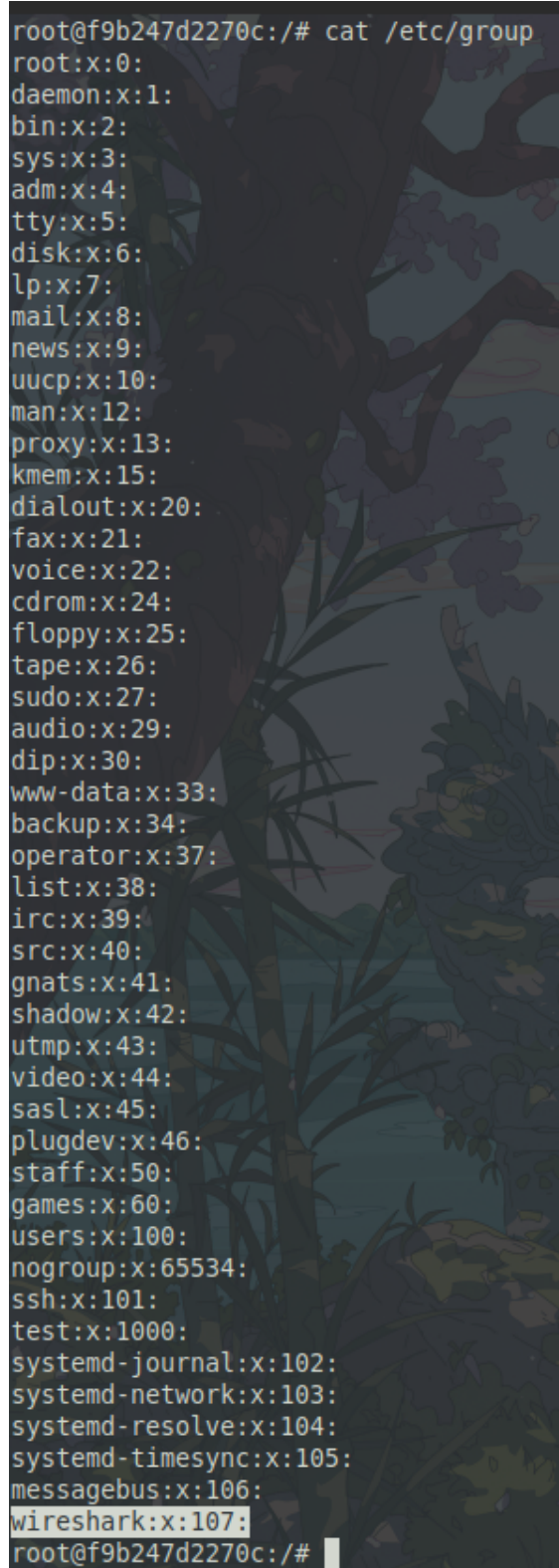
1. `useradd -m -s /bin/bash test`: Crea un nuevo usuario denominado "test".^{en} el sistema del contenedor.
 - `-m`: Crea un directorio de inicio para el nuevo usuario.
 - `-s /bin/bash`: Establece `/bin/bash` como la shell predeterminada para el usuario.
 - `test`: Es el nombre del usuario que se está creando.
2. `echo "test:test"| chpasswd`: Establece la contraseña para el usuario creado anteriormente. Aquí, `echo` imprime la cadena "test:test" el `|` (pipe) redirige la salida de `echo` como entrada a `chpasswd`, que es un comando para cambiar la contraseña. La cadena "test:test" representa el nombre de usuario (test) seguido por dos puntos y la contraseña (test en este caso). Este comando configura la contraseña del usuario test como test.

Posteriormente, para la creación del grupo, se utilizó la instalación por consola de Wireshark (**tshark**), la cual creará un grupo llamado **wireshark**.

Luego se emplearon los siguientes comandos, primero para listar los grupos ya existentes y confirmar la presencia de **wireshark**, y luego para agregar al usuario **test** al grupo **wireshark**.

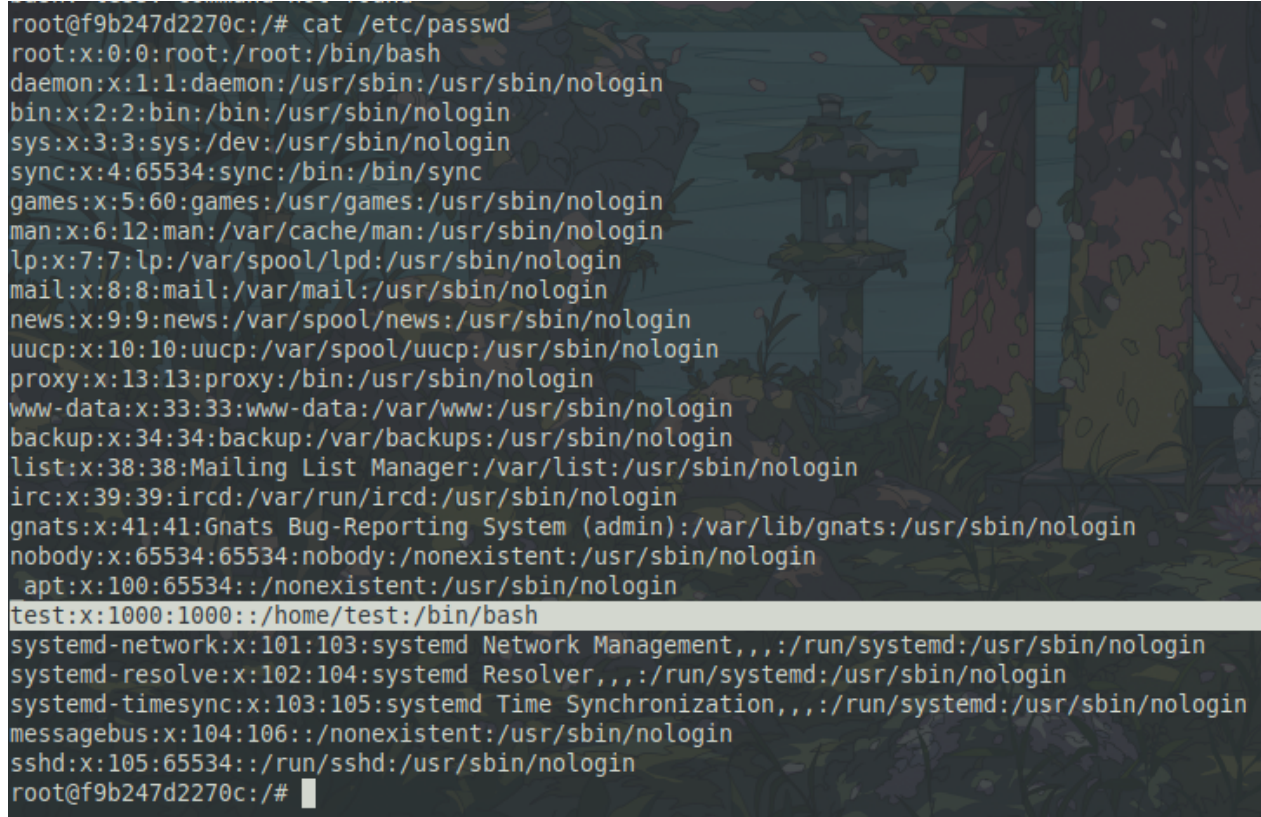
```
cat /etc/group # Lista grupos existentes.
```

```
usermod -a -G wireshark test # Agregar usuario test al grupo wireshark
```

A terminal window with a dark background and a faint, stylized illustration of a tree and foliage. The terminal shows the command 'cat /etc/group' being executed, displaying a list of system groups and their associated users and GIDs. The groups listed are: root, daemon, bin, sys, adm, tty, disk, lp, mail, news, uucp, man, proxy, kmem, dialout, fax, voice, cdrom, floppy, tape, sudo, audio, dip, www-data, backup, operator, list, irc, src, gnats, shadow, utmp, video, sasl, plugdev, staff, games, users, nogroup, ssh, test, systemd-journal, systemd-network, systemd-resolve, systemd-timesync, messagebus, and wireshark. The 'wireshark' group is highlighted with a light blue selection box.

```
root@f9b247d2270c:/# cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:
floppy:x:25:
tape:x:26:
sudo:x:27:
audio:x:29:
dip:x:30:
www-data:x:33:
backup:x:34:
operator:x:37:
list:x:38:
irc:x:39:
src:x:40:
gnats:x:41:
shadow:x:42:
utmp:x:43:
video:x:44:
sasl:x:45:
plugdev:x:46:
staff:x:50:
games:x:60:
users:x:100:
nogroup:x:65534:
ssh:x:101:
test:x:1000:
systemd-journal:x:102:
systemd-network:x:103:
systemd-resolve:x:104:
systemd-timesync:x:105:
messagebus:x:106:
wireshark:x:107:
root@f9b247d2270c:/#
```

Figura 3: Lista de Grupos existentes.



```
root@f9b247d2270c:/# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
apt:x:100:65534:/nonexistent:/usr/sbin/nologin
test:x:1000:1000:/home/test:/bin/bash
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:103:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:106:/nonexistent:/usr/sbin/nologin
sshd:x:105:65534:/run/sshd:/usr/sbin/nologin
root@f9b247d2270c:/#
```

Figura 4: Lista de Usuarios existentes.

```

root@f9b247d2270c:/# usermod -a -G wireshark test
root@f9b247d2270c:/# cat /etc/group
root:x:0:santiago-Aspire-A315-53:~/Documentos/Univ
daemon:x:1:aseña para santiago:
bin:x:2:1a2e6899:/# ssh test@172.17.0.2
sys:x:3:enticity of host '172.17.0.2 (172.17.0.2)'
adm:x:4: fingerprint is SHA256:NyoNmnhrt+Wm+b5P00
tty:x:5: sure you want to continue connecting (yes/
disk:x:6:Permanently added '172.17.0.2' (ECDSA) to
lp:x:7:172.17.0.2's password:
mail:x:8:0 Ubuntu 20.10 (GNU/Linux 5.15.0-88-gener
news:x:9:
uucp:x:10:tation: https://help.ubuntu.com
man:x:12:ment: https://landscape.canonical.com
proxy:x:13: https://ubuntu.com/advantage
kmem:x:15:
dialout:x:20:as been minimized by removing package
fax:x:21:red on a system that users do not log into
voice:x:22:
cdrom:x:24:this content, you can run the 'unminimi
floppy:x:25:wed Nov 22 06:14:49 2023 from 172.17.0
tape:x:26:7d2270c:~$ 
sudo:x:27:
audio:x:29:
dip:x:30:
www-data:x:33:
backup:x:34:
operator:x:37:
list:x:38:
irc:x:39:
src:x:40:
gnats:x:41:
shadow:x:42:
utmp:x:43:
video:x:44:
sasl:x:45:
plugdev:x:46:
staff:x:50:
games:x:60:
users:x:100:
nogroup:x:65534:
ssh:x:101:
test:x:1000:
systemd-journal:x:102:
systemd-network:x:103:
systemd-resolve:x:104:
systemd-timesync:x:105:
messagebus:x:106:
wireshark:x:107:test
root@f9b247d2270c:/#

```

Figura 5: Comando usermod sobre usuario test y lista actualizada de grupos existentes.

2.3. Tráfico generado por C1 (detallado)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.6	172.17.0.2	TCP	74	53788 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2607300009 TSecr=0 WS=128
2	0.000000000	172.17.0.2	172.17.0.6	TCP	74	22 → 53788 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=195660227 TSecr=2607300009 WS=128
3	0.000000000	172.17.0.6	172.17.0.2	TCP	66	53788 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2607300009 TSecr=195660227
4	0.000000000	172.17.0.6	172.17.0.2	SSHv2	190	Client: Protocol (SSH-2.0-OpenSSH_6.6.p1 Ubuntu-8)
5	0.000000000	172.17.0.2	172.17.0.6	TCP	66	22 → 53788 [ACK] Seq=1 Ack=35 Win=65152 Len=0 TSval=195660228 TSecr=2607300010
6	0.015406300	172.17.0.2	172.17.0.6	SSHv2	197	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
7	0.015406300	172.17.0.6	172.17.0.2	TCP	66	53788 → 22 [ACK] Seq=35 Ack=42 Win=64256 Len=0 TSval=2607300025 TSecr=195660242
8	0.016305906	172.17.0.6	172.17.0.2	SSHv2	2034	Client: Key Exchange Init
9	0.017490591	172.17.0.2	172.17.0.6	SSHv2	1122	Server: Key Exchange Init
10	0.021301963	172.17.0.6	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.029030731	172.17.0.2	172.17.0.6	SSHv2	346	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
12	0.035102909	172.17.0.6	172.17.0.2	SSHv2	82	Client: New Keys
13	0.077926746	172.17.0.6	172.17.0.2	TCP	66	22 → 53788 [ACK] Seq=1378 Ack=2067 Win=64128 Len=0 TSval=195660305 TSecr=2607300044
14	0.078010261	172.17.0.6	172.17.0.2	SSHv2	122	Client: Encrypted packet (len=56)
15	0.078038684	172.17.0.2	172.17.0.6	TCP	66	22 → 53788 [ACK] Seq=1378 Ack=2123 Win=64128 Len=0 TSval=195660305 TSecr=2607300087
16	0.078188625	172.17.0.2	172.17.0.6	SSHv2	122	Server: Encrypted packet (len=56)
17	0.078421746	172.17.0.6	172.17.0.2	SSHv2	138	Client: Encrypted packet (len=72)
18	0.086194103	172.17.0.2	172.17.0.6	SSHv2	122	Server: Encrypted packet (len=56)
19	0.129527295	172.17.0.6	172.17.0.2	TCP	66	53788 → 22 [ACK] Seq=2195 Ack=1490 Win=64128 Len=0 TSval=2607300139 TSecr=195660313
20	0.847583832	172.17.0.6	172.17.0.2	SSHv2	218	Client: Encrypted packet (len=152)
21	3.863229132	172.17.0.2	172.17.0.6	SSHv2	106	Server: Encrypted packet (len=40)
22	3.863292868	172.17.0.6	172.17.0.2	TCP	66	53788 → 22 [ACK] Seq=2347 Ack=1530 Win=64128 Len=0 TSval=2607303872 TSecr=195664090
23	3.863492849	172.17.0.6	172.17.0.2	SSHv2	194	Client: Encrypted packet (len=128)
24	3.881960541	172.17.0.2	172.17.0.6	SSHv2	698	Server: Encrypted packet (len=632)
25	3.925539355	172.17.0.6	172.17.0.2	TCP	66	53788 → 22 [ACK] Seq=2475 Ack=2162 Win=64128 Len=0 TSval=2607303935 TSecr=195664109
26	3.925658403	172.17.0.2	172.17.0.6	SSHv2	122	Server: Encrypted packet (len=56)
27	3.925688353	172.17.0.6	172.17.0.2	TCP	66	53788 → 22 [ACK] Seq=2475 Ack=2218 Win=64128 Len=0 TSval=2607303935 TSecr=195664153
28	3.925974220	172.17.0.6	172.17.0.2	SSHv2	466	Client: Encrypted packet (len=400)
29	3.929248337	172.17.0.2	172.17.0.6	SSHv2	186	Server: Encrypted packet (len=120)
30	3.929912842	172.17.0.6	172.17.0.2	SSHv2	570	Server: Encrypted packet (len=504)
31	3.930033393	172.17.0.2	172.17.0.6	TCP	66	53788 → 22 [ACK] Seq=2875 Ack=2842 Win=64128 Len=0 TSval=2607303939 TSecr=195664156
32	3.953415915	172.17.0.6	172.17.0.2	SSHv2	154	Server: Encrypted packet (len=88)
33	4.001952410	172.17.0.6	172.17.0.2	TCP	66	53788 → 22 [ACK] Seq=2875 Ack=2930 Win=64128 Len=0 TSval=2607304011 TSecr=195664180

Figura 6: Captura de tráfico generado mediante la conexión de cliente 1 con el servidor c4s1.

Los paquetes capturados son de tipo TCP y SSHv2. Dentro de ellos, se puede observar cómo los primeros tres paquetes (TCP) corresponden a un "handshake" de tres pasos con el cual se establece la comunicación. Posteriormente, se envían dos mensajes (uno del cliente y otro del servidor) en los que enumeran todos los algoritmos de encriptación que conocen, para luego seleccionar uno de ellos. En nuestro caso, se selecciona el de curva elíptica Diffie-Hellman, intercambiando así sus llaves públicas junto con el mensaje en la primera etapa de cifrado del algoritmo DH. Finalmente, se completan todos los pasos del algoritmo Diffie-Hellman.

El paquete "Key Exchange Init" contiene los siguientes algoritmos: curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1

2.4. Tráfico generado por C2 (detallado)

The screenshot shows a Wireshark capture of network traffic. The top bar indicates the file is 'Paso1_c2.pcapng'. The menu bar includes Archivo, Edición, Visualización, Jr, Captura, Analizar, Estadísticas, Telefonía, Wireless, Herramientas, and Ayuda. The packet list pane shows 33 captured packets. The packet details pane for the selected packet (No. 1) shows the following information:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.4	172.17.0.2	TCP	74	37166 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3366889966 TSecr=0 WS=128
2	0.000122403	172.17.0.2	172.17.0.4	TCP	74	22 → 37166 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2982137504 TSecr=3366889966 WS=128
3	0.000169925	172.17.0.4	172.17.0.2	TCP	66	37166 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3366889966 TSecr=2982137504
4	0.000707459	172.17.0.4	172.17.0.2	SSHv2	100	Client: Protocol (SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-8)
5	0.000727655	172.17.0.2	172.17.0.4	TCP	66	22 → 37166 [ACK] Seq=1 Ack=35 Win=65152 Len=0 TSval=2982137504 TSecr=3366889966
6	0.016872948	172.17.0.2	172.17.0.4	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
7	0.016920698	172.17.0.4	172.17.0.2	TCP	66	37166 → 22 [ACK] Seq=35 Ack=42 Win=64256 Len=0 TSval=3366889983 TSecr=2982137521
8	0.017788462	172.17.0.4	172.17.0.2	SSHv2	2034	Client: Key Exchange Init
9	0.019149670	172.17.0.2	172.17.0.4	SSHv2	1122	Server: Key Exchange Init
10	0.023081896	172.17.0.4	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.030996204	172.17.0.2	172.17.0.4	SSHv2	346	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
12	0.036658662	172.17.0.4	172.17.0.2	SSHv2	82	Client: New Keys
13	0.078878697	172.17.0.2	172.17.0.4	TCP	66	22 → 37166 [ACK] Seq=1378 Ack=2067 Win=64128 Len=0 TSval=2982137583 TSecr=3366890002
14	0.078969977	172.17.0.4	172.17.0.2	SSHv2	122	Client: Encrypted packet (len=56)
15	0.078969905	172.17.0.2	172.17.0.4	TCP	66	22 → 37166 [ACK] Seq=1378 Ack=2123 Win=64128 Len=0 TSval=2982137583 TSecr=3366890045
16	0.079135963	172.17.0.4	172.17.0.2	SSHv2	122	Server: Encrypted packet (len=56)
17	0.079343167	172.17.0.4	172.17.0.2	SSHv2	138	Client: Encrypted packet (len=72)
18	0.087077867	172.17.0.2	172.17.0.4	SSHv2	122	Server: Encrypted packet (len=56)
19	0.130960597	172.17.0.4	172.17.0.2	TCP	66	37166 → 22 [ACK] Seq=2195 Ack=1490 Win=64128 Len=0 TSval=3366890097 TSecr=2982137501
20	3.462668334	172.17.0.4	172.17.0.2	SSHv2	218	Client: Encrypted packet (len=152)
21	3.478174957	172.17.0.4	172.17.0.2	SSHv2	198	Server: Encrypted packet (len=40)
22	3.478221761	172.17.0.4	172.17.0.2	TCP	66	37166 → 22 [ACK] Seq=2347 Ack=1530 Win=64128 Len=0 TSval=3366893444 TSecr=2982140982
23	3.478453610	172.17.0.4	172.17.0.2	SSHv2	194	Client: Encrypted packet (len=128)
24	3.497133740	172.17.0.2	172.17.0.4	SSHv2	698	Server: Encrypted packet (len=632)
25	3.538859903	172.17.0.4	172.17.0.2	TCP	66	37166 → 22 [ACK] Seq=2475 Ack=2162 Win=64128 Len=0 TSval=3366893505 TSecr=2982141001
26	3.538928362	172.17.0.2	172.17.0.4	SSHv2	122	Server: Encrypted packet (len=56)
27	3.538943682	172.17.0.4	172.17.0.2	TCP	66	37166 → 22 [ACK] Seq=2475 Ack=2218 Win=64128 Len=0 TSval=3366893505 TSecr=2982141043
28	3.539106539	172.17.0.4	172.17.0.2	SSHv2	466	Client: Encrypted packet (len=400)
29	3.548922836	172.17.0.2	172.17.0.4	SSHv2	186	Server: Encrypted packet (len=128)
30	3.541266109	172.17.0.2	172.17.0.4	SSHv2	570	Server: Encrypted packet (len=504)
31	3.541355218	172.17.0.4	172.17.0.2	TCP	66	37166 → 22 [ACK] Seq=2875 Ack=2842 Win=64128 Len=0 TSval=3366893507 TSecr=2982141045
32	3.555026224	172.17.0.2	172.17.0.4	SSHv2	154	Server: Encrypted packet (len=88)
33	3.598849844	172.17.0.4	172.17.0.2	TCP	66	37166 → 22 [ACK] Seq=2875 Ack=2939 Win=64128 Len=0 TSval=3366893565 TSecr=2982141059

Figura 7: Captura de tráfico generado mediante la conexión de cliente 2 con el servidor c4s1.

Los paquetes capturados son de tipo TCP y SSHv2. Dentro de ellos, se puede observar cómo los primeros tres paquetes (TCP) corresponden a un "handshake" de tres pasos con el cual se establece la comunicación. Posteriormente, se envían dos mensajes (uno del cliente y otro del servidor) en los que enumeran todos los algoritmos de encriptación que conocen, para luego seleccionar uno de ellos. En nuestro caso, se selecciona el de curva elíptica Diffie-Hellman, intercambiando así sus llaves públicas junto con el mensaje en la primera etapa de cifrado del algoritmo DH. Finalmente, se completan todos los pasos del algoritmo Diffie-Hellman.

El paquete "Key Exchange Init" contiene los siguientes algoritmos: curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1

2.5. Tráfico generado por C3 (detallado)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.5	172.17.0.2	TCP	74	51526 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3926902071 TSecr=0 WS=128
2	0.000134459	172.17.0.2	172.17.0.5	TCP	74	22 → 51526 [SYN, ACK] Seq=9 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3289788114 TSecr=3926902071 WS=128
3	0.000204323	172.17.0.5	172.17.0.2	TCP	66	51526 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3926902071 TSecr=3289788114
4	0.001409045	172.17.0.5	172.17.0.2	SSHv2	180	Client: Protocol (SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-8)
5	0.001469778	172.17.0.2	172.17.0.5	TCP	66	22 → 51526 [ACK] Seq=1 Ack=35 Win=65152 Len=0 TSval=3289788115 TSecr=3926902072
6	0.020733639	172.17.0.2	172.17.0.5	SSHv2	187	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
7	0.020781744	172.17.0.5	172.17.0.2	TCP	66	51526 → 22 [ACK] Seq=35 Ack=42 Win=64256 Len=0 TSval=3926902092 TSecr=3289788135
8	0.021817730	172.17.0.5	172.17.0.2	SSHv2	2034	Client: Key Exchange Init
9	0.022942691	172.17.0.2	172.17.0.5	SSHv2	1122	Server: Key Exchange Init
10	0.026857935	172.17.0.5	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.035767566	172.17.0.2	172.17.0.5	SSHv2	346	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
12	0.041256839	172.17.0.5	172.17.0.2	SSHv2	82	Client: New Keys
13	0.081647452	172.17.0.2	172.17.0.5	TCP	66	22 → 51526 [ACK] Seq=1378 Ack=2067 Win=64128 Len=0 TSval=3926902112 TSecr=3926902153
14	0.081735379	172.17.0.5	172.17.0.2	SSHv2	122	Client: Encrypted packet (len=56)
15	0.081762191	172.17.0.2	172.17.0.5	TCP	66	22 → 51526 [ACK] Seq=1378 Ack=2123 Win=64128 Len=0 TSval=3289788196 TSecr=3926902153
16	0.081921013	172.17.0.2	172.17.0.5	SSHv2	122	Server: Encrypted packet (len=56)
17	0.082125336	172.17.0.5	172.17.0.2	SSHv2	138	Client: Encrypted packet (len=72)
18	0.089829105	172.17.0.2	172.17.0.5	SSHv2	122	Server: Encrypted packet (len=56)
19	0.133905195	172.17.0.5	172.17.0.2	TCP	66	51526 → 22 [ACK] Seq=2195 Ack=1490 Win=64128 Len=0 TSval=3926902205 TSecr=3289788204
20	2.536181581	172.17.0.5	172.17.0.2	SSHv2	218	Client: Encrypted packet (len=152)
21	2.566940301	172.17.0.2	172.17.0.5	SSHv2	186	Server: Encrypted packet (len=40)
22	2.567031981	172.17.0.5	172.17.0.2	TCP	66	51526 → 22 [ACK] Seq=2347 Ack=1530 Win=64128 Len=0 TSval=3926904638 TSecr=3289790681
23	2.567381670	172.17.0.5	172.17.0.2	SSHv2	194	Client: Encrypted packet (len=128)
24	2.591811362	172.17.0.2	172.17.0.5	SSHv2	698	Server: Encrypted packet (len=632)
25	2.633665279	172.17.0.5	172.17.0.2	TCP	66	51526 → 22 [ACK] Seq=2475 Ack=2162 Win=64128 Len=0 TSval=3926904705 TSecr=3289790706
26	2.633740687	172.17.0.2	172.17.0.5	SSHv2	122	Server: Encrypted packet (len=56)
27	2.633774027	172.17.0.5	172.17.0.2	TCP	66	51526 → 22 [ACK] Seq=2475 Ack=2218 Win=64128 Len=0 TSval=3926904705 TSecr=3289790748
28	2.634932584	172.17.0.5	172.17.0.2	SSHv2	466	Client: Encrypted packet (len=400)
29	2.637313484	172.17.0.2	172.17.0.5	SSHv2	186	Server: Encrypted packet (len=120)
30	2.637876349	172.17.0.2	172.17.0.5	SSHv2	570	Server: Encrypted packet (len=504)
31	2.637991443	172.17.0.5	172.17.0.2	TCP	66	51526 → 22 [ACK] Seq=2875 Ack=2842 Win=64128 Len=0 TSval=3926904709 TSecr=3289790751
32	2.659941506	172.17.0.2	172.17.0.5	SSHv2	154	Server: Encrypted packet (len=88)
33	2.701595421	172.17.0.5	172.17.0.2	TCP	66	51526 → 22 [ACK] Seq=2875 Ack=2930 Win=64128 Len=0 TSval=3926904773 TSecr=3289790774

Figura 8: Captura de tráfico generado mediante la conexión de cliente 3 con el servidor c4s1.

Los paquetes capturados son de tipo TCP y SSHv2. Dentro de ellos, se puede observar cómo los primeros tres paquetes (TCP) corresponden a un "handshake" de tres pasos con el cual se establece la comunicación. Posteriormente, se envían dos mensajes (uno del cliente y otro del servidor) en los que enumeran todos los algoritmos de encriptación que conocen, para luego seleccionar uno de ellos. En nuestro caso, se selecciona el de curva elíptica Diffie-Hellman, intercambiando así sus llaves públicas junto con el mensaje en la primera etapa de cifrado del algoritmo DH. Finalmente, se completan todos los pasos del algoritmo Diffie-Hellman.

El paquete "Key Exchange Init" contiene los siguientes algoritmos: curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1

2.6. Tráfico generado por C4 (4 (iface lo) (detallado))

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.2	172.17.0.2	TCP	74	46282 → 22 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=4152628826 TSecr=0 WS=128
2	0.000000000	172.17.0.2	172.17.0.2	TCP	74	22 → 46282 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=4152628826 TSecr=4152628826 WS=128
3	0.000014569	172.17.0.2	172.17.0.2	TCP	66	46282 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=4152628826 TSecr=4152628826
4	0.000155581	172.17.0.2	172.17.0.2	SSHv2	187	Client: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
5	0.000160936	172.17.0.2	172.17.0.2	TCP	66	22 → 46282 [ACK] Seq=1 Ack=42 Win=65536 Len=0 TSval=4152628826 TSecr=4152628826
6	0.007372064	172.17.0.2	172.17.0.2	SSHv2	187	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
7	0.007388096	172.17.0.2	172.17.0.2	TCP	66	46282 → 22 [ACK] Seq=42 Ack=42 Win=65536 Len=0 TSval=4152628834 TSecr=4152628834
8	0.007727926	172.17.0.2	172.17.0.2	SSHv2	1578	Client: Key Exchange Init
9	0.008355904	172.17.0.2	172.17.0.2	SSHv2	1122	Server: Key Exchange Init
10	0.010282210	172.17.0.2	172.17.0.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.016271446	172.17.0.2	172.17.0.2	SSHv2	574	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=228)
12	0.019936866	172.17.0.2	172.17.0.2	SSHv2	82	Client: New Keys
13	0.060313517	172.17.0.2	172.17.0.2	TCP	66	22 → 46282 [ACK] Seq=1606 Ack=1618 Win=65536 Len=0 TSval=4152628887 TSecr=4152628887
14	0.060334770	172.17.0.2	172.17.0.2	SSHv2	110	Client: Encrypted packet (len=44)
15	0.060346049	172.17.0.2	172.17.0.2	TCP	66	22 → 46282 [ACK] Seq=1606 Ack=1662 Win=65536 Len=0 TSval=4152628887 TSecr=4152628887
16	0.060535413	172.17.0.2	172.17.0.2	SSHv2	110	Server: Encrypted packet (len=44)
17	0.060617496	172.17.0.2	172.17.0.2	SSHv2	126	Client: Encrypted packet (len=60)
18	0.060646437	172.17.0.2	172.17.0.2	SSHv2	118	Server: Encrypted packet (len=52)
19	0.109295008	172.17.0.2	172.17.0.2	TCP	66	46282 → 22 [ACK] Seq=1722 Ack=1702 Win=65536 Len=0 TSval=4152628936 TSecr=4152628895
20	2.243266888	172.17.0.2	172.17.0.2	SSHv2	150	Client: Encrypted packet (len=84)
21	2.251334640	172.17.0.2	172.17.0.2	SSHv2	94	Server: Encrypted packet (len=28)
22	2.251341702	172.17.0.2	172.17.0.2	TCP	66	46282 → 22 [ACK] Seq=1806 Ack=1730 Win=65536 Len=0 TSval=4152631078 TSecr=4152631078
23	2.251398487	172.17.0.2	172.17.0.2	SSHv2	178	Client: Encrypted packet (len=112)
24	2.260457338	172.17.0.2	172.17.0.2	SSHv2	694	Server: Encrypted packet (len=628)
25	2.301309876	172.17.0.2	172.17.0.2	TCP	66	46282 → 22 [ACK] Seq=1918 Ack=2358 Win=65536 Len=0 TSval=4152631128 TSecr=4152631087
26	2.301330331	172.17.0.2	172.17.0.2	SSHv2	110	Server: Encrypted packet (len=44)
27	2.301344827	172.17.0.2	172.17.0.2	TCP	66	46282 → 22 [ACK] Seq=1918 Ack=2402 Win=65536 Len=0 TSval=4152631128 TSecr=4152631128
28	2.301582916	172.17.0.2	172.17.0.2	SSHv2	442	Client: Encrypted packet (len=376)
29	2.303712728	172.17.0.2	172.17.0.2	SSHv2	174	Server: Encrypted packet (len=108)
30	2.304078677	172.17.0.2	172.17.0.2	SSHv2	566	Server: Encrypted packet (len=500)
31	2.304106593	172.17.0.2	172.17.0.2	TCP	66	46282 → 22 [ACK] Seq=2294 Ack=3010 Win=65536 Len=0 TSval=4152631130 TSecr=4152631130
32	2.313340294	172.17.0.2	172.17.0.2	SSHv2	150	Server: Encrypted packet (len=84)
33	2.354197770	172.17.0.2	172.17.0.2	TCP	66	46282 → 22 [ACK] Seq=2294 Ack=3094 Win=65536 Len=0 TSval=4152631181 TSecr=4152631140

Figura 9: Captura de tráfico generado mediante la conexión de cliente 4-v2 con el servidor c4s1.

Los paquetes capturados son de tipo TCP y SSHv2. Dentro de ellos, se puede observar cómo los primeros tres paquetes (TCP) corresponden a un "handshake" de tres pasos con el cual se establece la comunicación. Posteriormente, se envían dos mensajes (uno del cliente y otro del servidor) en los que enumeran todos los algoritmos de encriptación que conocen, para luego seleccionar uno de ellos. En nuestro caso, se selecciona el de curva elíptica Diffie-Hellman, intercambiando así sus llaves públicas junto con el mensaje en la primera etapa de cifrado del algoritmo DH. Finalmente, se completan todos los pasos del algoritmo Diffie-Hellman.

A diferencia de las tres capturas anteriores, en esta, aunque también tenemos paquetes de tipo TCP y SSHv2, los tamaños varían notablemente. Por ejemplo, en la línea 8 de la captura, el tamaño del paquete "Key Exchange Init.^{es} de 1578, a diferencia de los 2034 de las otras tres capturas. Estos cambios se deben a que el contenido es distinto; por ejemplo, en la parte del "Key Exchange Init", ahora contiene menos algoritmos que se pueden usar, lo que causa una disminución en la longitud de estos paquetes.

El paquete "Key Exchange Init" contiene los siguientes algoritmos: curve25519-sha256, curve25519-sha256@libssh.org, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group-exchange-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, diffie-hellman-group14-sha256, ext-info-c

2.7. Diferencia entre C1 y C2

Las diferencias encontradas fueron las siguientes:

- Los valores del Tsval y el Tserc son distintos.

- No se encontraron grandes diferencias a simple vista.

2.8. Diferencia entre C2 y C3

Las diferencias encontradas fueron las siguientes:

- Los valores del T_{sval} y el T_{serc} son distintos.
- No se encontraron grandes diferencias a simple vista.

2.9. Diferencia entre C3 y C4

Las diferencias encontradas son las siguientes:

- La primera diferencia que se encuentra es el largo (Lenght) de algunos paquetes en c4 es diferente a c3, como el por ejemplo el paquete 4 y 8.
- Los valores del T_{sval} y el T_{serc} son distintos.
- No se encontraron grandes diferencias a simple vista.

3. Desarrollo (Parte 2)

3.1. Identificación del cliente ssh

Comparamos las 4 capturas obtenidas con la imagen del paso 2, analizando el tamaño (Length) de los paquetes. Observamos que muchas de ellas comparten tamaños. Sin embargo, al considerar las diferencias del punto anterior, notamos que los clientes 1, 2 y 3 tienen un tamaño de 2034, mientras que el cliente 4 tiene un tamaño de 1578. Usando esto como referencia, se presume que la captura del paso 2 mostrada en las instrucciones corresponde al cliente 4 (c4) y, por ende, debería mostrar "(SSH-2.0-OpenSSH_8.3pi)".

3.2. Replicación de tráfico (paso por paso)

Ahora que identificamos a qué cliente pertenece la imagen, vamos a replicar el mensaje. Para ello, ejecutaremos los siguientes comandos necesarios:

```
apt-get install git
apt-get install vim
apt-get install autoconf
apt-get install libssl-dev
apt-get install zlib1g-dev
apt-get install gcc
apt-get install make
```

Luego, realizamos un `git clone` del siguiente repositorio de SSH: <https://github.com/openssh/openssh-portable>. Una vez clonado, nos movemos a la carpeta y usamos `vim version.h` para modificar el archivo, que inicialmente se veía así:

```
#define SSH_VERSION "OpenSSH_9.5"

#define SSH_PORTABLE "p1"
#define SSH_RELEASE SSH_VERSION SSH_PORTABLE
```

Resultando en la siguiente modificación:


```

1  /* $OpenBSD: version.h,v 1.99 2023/10/04 04:04:09 djm Exp $ */
2
3  #define SSH_VERSION "OpenSSH_?"
4
5  #define SSH_PORTABLE "p1"
6  #define SSH_RELEASE SSH_VERSION

```

Figura 10: Modificación del archivo `versión.h`.

Luego se realizan los siguientes comandos:

```

cd openssh-portable
autoreconf
./configure
make && make tests

```

Por último, se comienza a capturar tráfico con el servidor y con el cliente nos conectamos usando `ssh test@172.17.0.2`.

19	21.847777030	172.17.0.3	172.17.0.2	TCP	74 55710 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
20	21.847795049	172.17.0.2	172.17.0.3	TCP	74 22 → 55710 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
21	21.847809436	172.17.0.3	172.17.0.2	TCP	66 55710 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSv
22	21.848095226	172.17.0.3	172.17.0.2	SSHv2	85 Client: Protocol (SSH-2.0-OpenSSH_?)
23	21.848104166	172.17.0.2	172.17.0.3	TCP	66 22 → 55710 [ACK] Seq=1 Ack=20 Win=65152 Len=0 TSv
24	21.855930226	172.17.0.2	172.17.0.3	SSHv2	107 Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-10
25	21.855942588	172.17.0.3	172.17.0.2	TCP	66 55710 → 22 [ACK] Seq=20 Ack=42 Win=64256 Len=8 TS
26	21.856078429	172.17.0.3	172.17.0.2	SSHv2	1578 Client: Key Exchange Init
27	21.856518778	172.17.0.2	172.17.0.3	SSHv2	1122 Server: Key Exchange Init
28	21.857703774	172.17.0.3	172.17.0.2	SSHv2	114 Client: Elliptic Curve Diffie-Hellman Key Exchange

Figura 11: Captura de paquetes con (SSH-2.0-OpenSSH_?).

4. Desarrollo (Parte 3)

4.1. Replicación de tráfico (paso a paso)

Para reducir la longitud del paquete "Key Exchange Init."^a menos de 300, modificaremos el archivo *ssh_config*, eliminando los algoritmos conocidos que se envían al cliente para acordar cuál utilizar. Esto nos permitirá reducir la longitud del paquete.

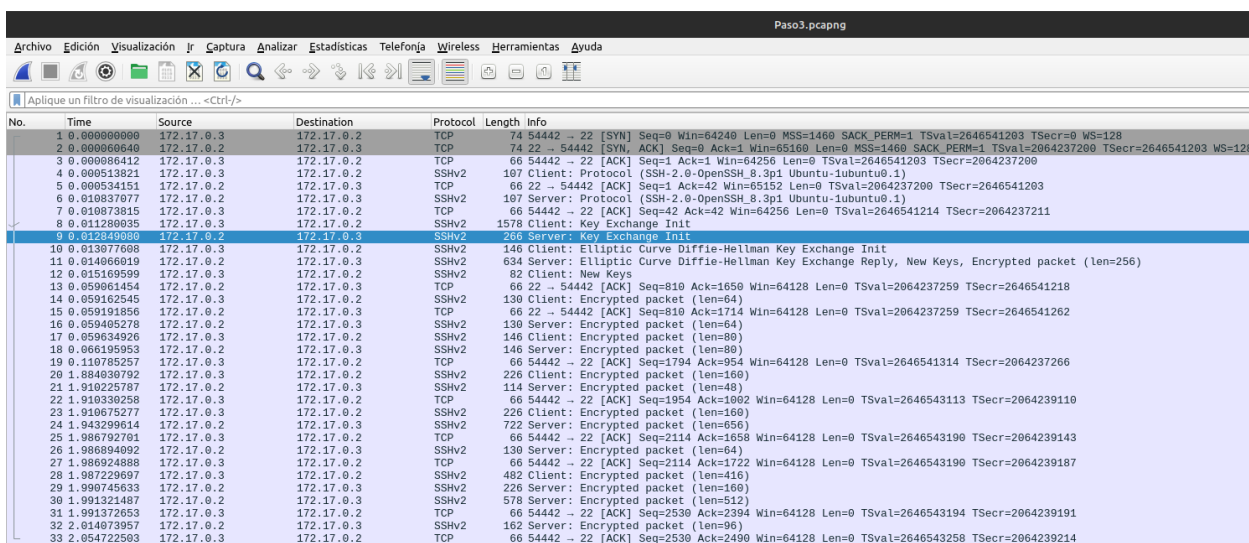
Para llevar a cabo este proceso, se siguieron los siguientes pasos:

1. Nos dirigimos a la ruta `cd /etc/ssh`.
2. Ejecutamos `ls` para visualizar todos los elementos en la carpeta.
3. Utilizamos `vim sshd_config` para editar el archivo y agregamos lo siguiente al final del mismo:

```
Ciphers aes128-ctr
HostKeyAlgorithms ecdsa-sha2-nistp256
KexAlgorithms ecdh-sha2-nistp256
MACs hmac-sha2-256
```

4. Guardamos los cambios y salimos del archivo.
5. Reiniciamos SSH con `sudo service ssh restart`.
6. Configuramos el servidor para capturar tráfico y nos dirigimos a nuestro cliente 4 (c4) para conectarnos con `ssh test@172.17.0.2`.

Estos pasos resultan en lo siguiente:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.3	172.17.0.2	TCP	74	54442 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2646541203 TSecr=0 WS=128
2	0.000000040	172.17.0.2	172.17.0.3	TCP	74	22 → 54442 [SYN, ACK] Seq=9 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2064237209 TSecr=2646541203 WS=128
3	0.000000412	172.17.0.3	172.17.0.2	TCP	66	54442 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2646541203 TSecr=2064237209
4	0.000513821	172.17.0.3	172.17.0.2	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
5	0.000534151	172.17.0.2	172.17.0.3	TCP	66	22 → 54442 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=2064237209 TSecr=2646541203
6	0.010837077	172.17.0.2	172.17.0.3	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.3p1 Ubuntu-1ubuntu0.1)
7	0.010873815	172.17.0.3	172.17.0.2	TCP	66	54442 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=2064237211 TSecr=2064237211
8	0.011289035	172.17.0.3	172.17.0.2	SSHv2	1578	Client: Key Exchange Init
9	0.012849080	172.17.0.2	172.17.0.3	SSHv2	266	Server: Key Exchange Init
10	0.013077608	172.17.0.3	172.17.0.2	SSHv2	146	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.014066019	172.17.0.2	172.17.0.3	SSHv2	634	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=256)
12	0.015169599	172.17.0.3	172.17.0.2	SSHv2	82	Client: New Keys
13	0.059061454	172.17.0.2	172.17.0.3	TCP	66	22 → 54442 [ACK] Seq=810 Ack=1650 Win=64128 Len=0 TSval=2064237259 TSecr=2646541218
14	0.059162545	172.17.0.3	172.17.0.2	SSHv2	130	Client: Encrypted packet (len=64)
15	0.059191856	172.17.0.2	172.17.0.3	TCP	66	22 → 54442 [ACK] Seq=810 Ack=1714 Win=64128 Len=0 TSval=2064237259 TSecr=2646541262
16	0.059405278	172.17.0.2	172.17.0.3	SSHv2	130	Server: Encrypted packet (len=64)
17	0.059634926	172.17.0.3	172.17.0.2	SSHv2	146	Client: Encrypted packet (len=80)
18	0.066195953	172.17.0.2	172.17.0.3	SSHv2	146	Server: Encrypted packet (len=80)
19	0.110785257	172.17.0.3	172.17.0.2	TCP	66	54442 → 22 [ACK] Seq=1794 Ack=954 Win=64128 Len=0 TSval=2064237266 TSecr=2064237266
20	1.884030792	172.17.0.3	172.17.0.2	SSHv2	226	Client: Encrypted packet (len=160)
21	1.910225787	172.17.0.2	172.17.0.3	SSHv2	114	Server: Encrypted packet (len=40)
22	1.910338258	172.17.0.3	172.17.0.2	TCP	66	54442 → 22 [ACK] Seq=1954 Ack=1002 Win=64128 Len=0 TSval=2064239110 TSecr=2064239110
23	1.910675277	172.17.0.3	172.17.0.2	SSHv2	226	Client: Encrypted packet (len=160)
24	1.943299614	172.17.0.2	172.17.0.3	SSHv2	722	Server: Encrypted packet (len=656)
25	1.986792701	172.17.0.3	172.17.0.2	TCP	66	54442 → 22 [ACK] Seq=2114 Ack=1658 Win=64128 Len=0 TSval=2646543190 TSecr=2064239143
26	1.986894092	172.17.0.2	172.17.0.3	SSHv2	130	Server: Encrypted packet (len=64)
27	1.986924888	172.17.0.3	172.17.0.2	TCP	66	54442 → 22 [ACK] Seq=2114 Ack=1722 Win=64128 Len=0 TSval=2646543190 TSecr=2064239187
28	1.987229697	172.17.0.3	172.17.0.2	SSHv2	482	Client: Encrypted packet (len=416)
29	1.990745633	172.17.0.2	172.17.0.3	SSHv2	226	Server: Encrypted packet (len=160)
30	1.991321487	172.17.0.2	172.17.0.3	SSHv2	578	Server: Encrypted packet (len=512)
31	1.991372653	172.17.0.3	172.17.0.2	TCP	66	54442 → 22 [ACK] Seq=2530 Ack=2394 Win=64128 Len=0 TSval=2646543194 TSecr=2064239191
32	2.014073957	172.17.0.2	172.17.0.3	SSHv2	162	Server: Encrypted packet (len=96)
33	2.054722503	172.17.0.3	172.17.0.2	TCP	66	54442 → 22 [ACK] Seq=2530 Ack=2490 Win=64128 Len=0 TSval=2646543258 TSecr=2064239214

Figura 12: Captura de paquetes "Key Exchange Init" del servidor con una longitud de 266 bytes.

5. Conclusiones y comentarios

En general, se logró cumplir el objetivo de la experiencia, aunque se enfrentaron dificultades que resultaron difíciles de superar. Por ejemplo, la captura de datos a través de tshark en c4s1 con el cliente c4 presentó problemas, pero después de varios intentos se logró resolver. También hubo desafíos al modificar el archivo sshd.config, ya que inicialmente no alcanzamos el tamaño solicitado, pero finalmente logramos superarlos.

A pesar de las dificultades, fue una experiencia entretenida y exitosa. Mis disculpas por la conclusión abrupta, pero tengo sueño y un examen el jueves por la mañana, además de otros trabajos que completar :)

6. Referencias

- Repositorio de Github.
- Repositorio OpenSSH.