

MÁSTER EN REVERSING, ANÁLISIS DE MALWARE Y BUG HUNTING

MÁSTER EN *ANÁLISIS DE MALWARE Y REVERSING*

María Sonia Salido Fernández

Módulo 5 - Tarea 2



Campus Internacional
CIBERSEGURIDAD



- 1. Análisis del protocolo
 - Filtración el tráfico específico
 - Localización las conversaciones reales de la app. Sesiones TCP
- 2. Información intercambiada entre el cliente y el servidor

Analizar el tráfico incluido en el fichero actividad2.pcap.

Esa captura de tráfico se ha realizado en la misma máquina (192.168.1.33) donde se ejecuta el servidor de la aplicación a analizar y que corre en el puerto TCP/7000 (que no es el puerto estándar de ese protocolo). El fichero de captura también incluye otro tipo de tráfico, que no está relacionado con la aplicación, por lo que se deberían utilizar los filtros de visualización de Wireshark para encontrar el tráfico de la aplicación.

1. Análisis del protocolo

El objetivo de la práctica es:

- Estudiar el protocolo empleado por la aplicación.
- Identificar el tipo de mensajes intercambiados por el cliente y el servidor.
- Intentar interpretar el contenido y significado de los mismos.
- Obtener la información intercambiada.
- Intentar identificar el protocolo exacto que está siendo empleado por la aplicación.
- Incluir capturas de pantalla de Wireshark:
 - Especificación del protocolo de la aplicación
 - La información intercambiada entre el cliente y el servidor.

Para resolver este ejercicio de análisis de tráfico con Wireshark, necesitamos filtrar el ruido y centrarnos específicamente en la actividad del puerto 7000. Por ello debemos eliminar el tráfico de fondo como TLS e ICMP.

Filtración el tráfico específico

Dado que el enunciado indica que la aplicación corre en el puerto TCP 7000, lo primero es limpiar la vista. En wireshark, aplicamos el filtro: `tcp.port == 7000`:

No.	Time	Source	Destination	Protocol	Length	Info
547	23.658353	192.168.1.110	servidor	TCP	66	62897 → 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
548	23.658377	servidor	192.168.1.110	TCP	66	7000 → 62897 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
549	23.658573	192.168.1.110	servidor	TCP	54	62897 → 7000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
550	23.659351	192.168.1.110	servidor	TCP	56	62897 → 7000 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=2
551	23.659418	servidor	192.168.1.110	TCP	54	7000 → 62897 [ACK] Seq=1 Ack=3 Win=64256 Len=0
552	23.662103	servidor	192.168.1.110	TCP	111	7000 → 62897 [PSH, ACK] Seq=1 Ack=3 Win=64256 Len=57
553	23.662161	servidor	192.168.1.110	TCP	220	7000 → 62897 [FIN, PSH, ACK] Seq=58 Ack=3 Win=64256 Len=166
554	23.662373	192.168.1.110	servidor	TCP	54	62897 → 7000 [ACK] Seq=3 Ack=225 Win=2102016 Len=0
555	23.662554	192.168.1.110	servidor	TCP	54	62897 → 7000 [FIN, ACK] Seq=3 Ack=225 Win=2102016 Len=0
556	23.662565	servidor	192.168.1.110	TCP	54	7000 → 62897 [ACK]

```

Seq=225 Ack=4 Win=64256 Len=0
557 25.852265 192.168.1.110 servidor TCP 66 62900 → 7000 [SYN]
Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
558 25.852296 servidor 192.168.1.110 TCP 66 7000 → 62900 [SYN, ACK]
Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
559 25.852649 192.168.1.110 servidor TCP 54 62900 → 7000 [ACK]
Seq=1 Ack=1 Win=2102272 Len=0
560 25.853468 192.168.1.110 servidor TCP 66 62900 → 7000 [PSH, ACK]
Seq=1 Ack=1 Win=2102272 Len=12
561 25.853479 servidor 192.168.1.110 TCP 54 7000 → 62900 [ACK]
Seq=1 Ack=13 Win=64256 Len=0
562 25.858682 servidor 192.168.1.110 TCP 67 7000 → 62900 [PSH, ACK]
Seq=1 Ack=13 Win=64256 Len=13
563 25.858754 servidor 192.168.1.110 TCP 54 7000 → 62900 [FIN, ACK]
Seq=14 Ack=13 Win=64256 Len=0
564 25.859028 192.168.1.110 servidor TCP 54 62900 → 7000 [ACK]
Seq=13 Ack=15 Win=2102272 Len=0
565 25.859976 192.168.1.110 servidor TCP 54 62900 → 7000 [FIN, ACK]
Seq=13 Ack=15 Win=2102272 Len=0
566 25.859993 servidor 192.168.1.110 TCP 54 7000 → 62900 [ACK]
Seq=15 Ack=14 Win=64256 Len=0
568 31.351551 192.168.1.110 servidor TCP 66 62910 → 7000 [SYN]
Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
569 31.351573 servidor 192.168.1.110 TCP 66 7000 → 62910 [SYN, ACK]
Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
570 31.351799 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=1 Ack=1 Win=2102272 Len=0
571 31.352463 192.168.1.110 servidor TCP 65 62910 → 7000 [PSH, ACK]
Seq=1 Ack=1 Win=2102272 Len=11
572 31.352469 servidor 192.168.1.110 TCP 54 7000 → 62910 [ACK]
Seq=1 Ack=12 Win=64256 Len=0
573 31.355524 servidor 192.168.1.110 TCP 4150 7000 → 62910 [PSH, ACK]
Seq=1 Ack=12 Win=64256 Len=4096 [TCP PDU reassembled in 575]
574 31.355543 servidor 192.168.1.110 TCP 7354 7000 → 62910 [PSH, ACK]
Seq=4097 Ack=12 Win=64256 Len=7300 [TCP PDU reassembled in 575]
575 31.355556 servidor 192.168.1.110 Gryphon 2974
576 31.355954 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=12 Ack=14317 Win=2087936 Len=0
577 31.355963 servidor 192.168.1.110 Gryphon 14654 - Invalid --
Invalid -
578 31.355967 servidor 192.168.1.110 Gryphon 14654 - Invalid -
579 31.355954 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2092032 Len=0
580 31.355954 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2096128 Len=0
581 31.355954 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2100224 Len=0
582 31.355954 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2102272 Len=0
583 31.356253 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=12 Ack=43517 Win=2073088 Len=0
584 31.356259 servidor 192.168.1.110 Gryphon 29254 - Invalid -
585 31.356263 servidor 192.168.1.110 TCP 2974 7000 → 62910 [PSH, ACK]
Seq=72717 Ack=12 Win=64256 Len=2920 [TCP PDU reassembled in 586]

```

```

586 31.356265 servidor 192.168.1.110 Gryphon 26334 - Invalid --
Invalid -
587 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2077184 Len=0
588 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2081280 Len=0
589 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2085376 Len=0
590 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2089472 Len=0
591 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2093568 Len=0
592 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2097664 Len=0
593 31.356274 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2101760 Len=0
594 31.356563 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=12 Ack=101917 Win=2043392 Len=0
595 31.356572 servidor 192.168.1.110 Gryphon 35094 - Invalid --
Invalid -
596 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2048000 Len=0
597 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2052096 Len=0
598 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2056192 Len=0
599 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2060288 Len=0
600 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2064384 Len=0
601 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2068480 Len=0
602 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2072576 Len=0
603 31.356593 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2076672 Len=0
604 31.356593 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2080768 Len=0
605 31.356593 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2084864 Len=0
606 31.356673 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2088960 Len=0
607 31.356676 servidor 192.168.1.110 TCP 22842 7000 → 62910 [PSH,
ACK] Seq=136957 Ack=12 Win=64256 Len=22788 [TCP PDU reassembled in 611]
608 31.356673 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2093056 Len=0
609 31.356673 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2097152 Len=0
610 31.356673 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2101248 Len=0
611 31.356715 servidor 192.168.1.110 Gryphon 23414 - Invalid -
612 31.356816 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=12 Ack=136957 Win=2066176 Len=0
613 31.356822 servidor 192.168.1.110 Gryphon 43854 - Invalid -

```

614 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2071296 Len=0
615 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2075392 Len=0
616 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2079488 Len=0
617 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2083584 Len=0
618 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2087680 Len=0
619 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2091776 Len=0
620 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2095872 Len=0
621 31.356837 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2099968 Len=0
622 31.356837 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2102272 Len=0
623 31.356855 servidor 192.168.1.110 TCP 485 7000 → 62910 [FIN, PSH,
ACK] Seq=226905 Ack=12 Win=64256 Len=431
624 31.356898 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=12 Ack=183105 Win=2056192 Len=0
625 31.356898 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2060288 Len=0
626 31.356898 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2064384 Len=0
627 31.356898 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2068480 Len=0
628 31.356898 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2072576 Len=0
629 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2076672 Len=0
630 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2080768 Len=0
631 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2084864 Len=0
632 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2088960 Len=0
633 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2093056 Len=0
634 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2097152 Len=0
635 31.357037 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=12 Ack=227337 Win=2052864 Len=0
636 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2056960 Len=0
637 31.357057 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2061056 Len=0
638 31.357057 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2065152 Len=0
639 31.357057 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2069248 Len=0
640 31.357058 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2073344 Len=0

```

641 31.357071 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2077440 Len=0
642 31.357087 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2081536 Len=0
643 31.357089 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2085632 Len=0
644 31.357112 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2089728 Len=0
645 31.357114 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2093824 Len=0
646 31.357131 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2097920 Len=0
647 31.357152 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2102016 Len=0
650 34.163277 192.168.1.110 servidor TCP 54 62910 → 7000 [FIN, ACK]
Seq=12 Ack=227337 Win=2102016 Len=0
651 34.163311 servidor 192.168.1.110 TCP 54 7000 → 62910 [ACK]
Seq=227337 Ack=13 Win=64256 Len=0
1074 52.655474 192.168.1.110 servidor TCP 66 62949 → 7000 [SYN]
Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1075 52.655496 servidor 192.168.1.110 TCP 66 7000 → 62949 [SYN,
ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
1076 52.655718 192.168.1.110 servidor TCP 54 62949 → 7000 [ACK]
Seq=1 Ack=1 Win=2102272 Len=0
1077 52.656403 192.168.1.110 servidor TCP 61 62949 → 7000 [PSH,
ACK] Seq=1 Ack=1 Win=2102272 Len=7
1078 52.656409 servidor 192.168.1.110 TCP 54 7000 → 62949 [ACK]
Seq=1 Ack=8 Win=64256 Len=0
1079 52.660410 servidor 192.168.1.110 TCP 101 7000 → 62949 [PSH,
ACK] Seq=1 Ack=8 Win=64256 Len=47
1080 52.660437 servidor 192.168.1.110 TCP 54 7000 → 62949 [FIN,
ACK] Seq=48 Ack=8 Win=64256 Len=0
1081 52.660994 192.168.1.110 servidor TCP 54 62949 → 7000 [ACK]
Seq=8 Ack=49 Win=2102272 Len=0
1082 52.661216 192.168.1.110 servidor TCP 54 62949 → 7000 [FIN,
ACK] Seq=8 Ack=49 Win=2102272 Len=0
1083 52.661226 servidor 192.168.1.110 TCP 54 7000 → 62949 [ACK]
Seq=49 Ack=9 Win=64256 Len=0

```

En el primer paquete que aparece tras aplicar el filtro (547), vamos hacer un seguimiento con TCP stream:
Hacemos click derecho en el paquete 547 y seleccionamos Follow (Seguir) → TCP Stream.

No.	Time	Source	Destination	Protocol	Length	Info
547	23.658353	192.168.1.110	servidor	TCP	66	62897 → 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
548	23.658377	servidor	192.168.1.110	TCP	66	7000 → 62897 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
549	23.658573	192.168.1.110	servidor	TCP	54	62897 → 7000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
550	23.659351	192.168.1.110	servidor	TCP	66	7000 → 62897 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=47
551	23.659418	servidor	192.168.1.110	TCP	54	62897 → 7000 [ACK] Seq=2 Ack=2 Win=64256 Len=0
552	23.662103	servidor	192.168.1.110	TCP	66	7000 → 62897 [PSH, ACK] Seq=2 Ack=2 Win=2102272 Len=0
553	23.662161	servidor	192.168.1.110	TCP	54	62897 → 7000 [ACK] Seq=3 Ack=3 Win=64256 Len=0
554	23.662373	192.168.1.110	servidor	TCP	66	7000 → 62897 [PSH, ACK] Seq=3 Ack=3 Win=2102272 Len=0
555	23.662554	192.168.1.110	servidor	TCP	54	62897 → 7000 [ACK] Seq=4 Ack=4 Win=64256 Len=0
556	23.662565	servidor	192.168.1.110	TCP	66	7000 → 62897 [PSH, ACK] Seq=4 Ack=4 Win=2102272 Len=0

```
iWelcome to the 192.168.1.33:7000 server! fake (NULL) 0
i fake (NULL) 0
0Hello /hello.txt 192.168.1.33 7000 +
1RFCs /dir/ 192.168.1.33 7000 +
gLena /lena.gif 192.168.1.33 7000 +
1Quux Mega Server / 142.4.200.132 7000
```

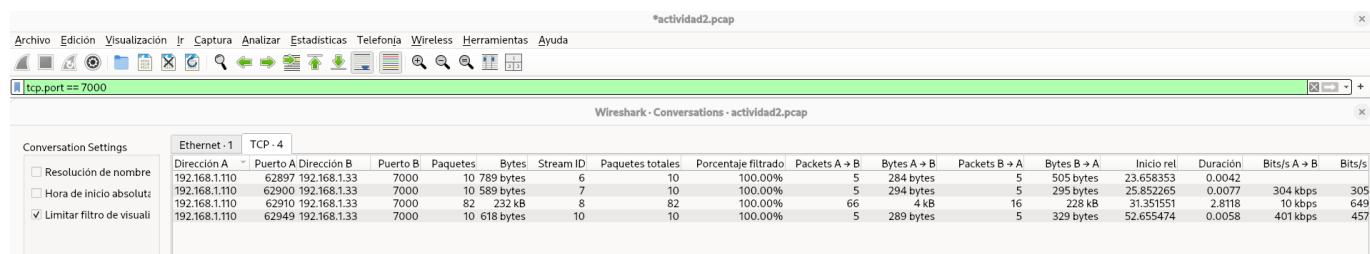
donde:

- Vemos palabras legibles como "Welcome", "Hello", "Mega Server". Esto parece que es un protocolo basado en texto (ASCII), lo cual descarta protocolos complejos cifrados o puramente binarios.**
- Tambien parece una comunicación en la que se piden recursos, como hello.txt, lena.gif**

Localización las conversaciones reales de la app. Sesiones TCP

Aunque filtrremo por 7000, es buena práctica confirmar cuántas conexiones hay y quién habla con quién:

- Vamos a Statistics → Conversations → TCP → Ordenamos por Port o usamos el buscador para encontrar 7000.
- Apuntamos:
 - IP/puerto del cliente
 - IP/puerto del servidor (192.168.1.33:7000)
 - Número de bytes transferidos en cada sentido (te ayuda a identificar la sesión “importante”).



Dirección A, Puerto A, Dirección B, Puerto B, Paquetes, Bytes, Stream ID, Paquetes totales, Porcentaje filtrado, Packets A → B, Bytes A → B, Packets B → A, Bytes B → A, Inicio rel, Duración, Bits/s A → B, Bits/s B → A, Flows
 "192.168.1.110", 62897, "192.168.1.33", 7000, 10, 789, 6, 10, 100, 5, 284, 5, 505, 23.658353, 0.00421200000002547, "", "", 3
 "192.168.1.110", 62900, "192.168.1.33", 7000, 10, 589, 7, 10, 100, 5, 294, 5, 295, 25.852265, 0.00772800000000179, 304347, 305383, 2
 "192.168.1.110", 62910, "192.168.1.33", 7000, 82, 231798, 8, 82, 100, 66, 3587, 16, 228
 211, 31.351551, 2.8117599999999996, 10205, 649304, 14
 "192.168.1.110", 62949, "192.168.1.33", 7000, 10, 618, 10, 10, 100, 5, 289, 5, 329, 52.655474, 0.0057520000000109, 401947, 457579, 2

Esta vista de Conversaciones TCP es fundamental porque nos aporta una visión global de cómo interactuó el cliente con la aplicación. Al filtrar por el puerto 7000, vemos que se establecieron 4 flujos o sesiones independientes entre el cliente (192.168.1.110) y el servidor (192.168.1.33).

Y los volúmenes son muy diferentes:

- Stream 6: 10 paquetes / 789 bytes → muy corto
- Stream 7: 10 paquetes / 589 bytes → muy corto
- Stream 8: 82 paquetes / 231,798 bytes → con diferencia, el principal
- Stream 10: 10 paquetes / 618 bytes → muy corto

Además, en el Stream 8:

- A → B (cliente → servidor): 3,587 bytes
- B → A (servidor → cliente): 228,211 bytes

Esto encaja con un patrón típico: el cliente hace una petición pequeña y el servidor responde con un contenido grande, que puede ser una descarga, listado, datos, o un archivo. Es por esto que vamos a apostar por este stream para seguir investigando `tcp.stream == 8`.

Aplicamos el filtro: `tcp.stream == 8`

No.	Time	Source	Destination	Protocol	Length	Info
568	31.351551	192.168.1.110	servidor	TCP	66	62910 → 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
569	31.351573	servidor	192.168.1.110	TCP	66	7000 → 62910 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
570	31.351799	192.168.1.110	servidor	TCP	54	62910 → 7000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
571	31.352463	192.168.1.110	servidor	TCP	65	62910 → 7000 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=11
572	31.352469	servidor	192.168.1.110	TCP	54	7000 → 62910 [ACK] Seq=1 Ack=12 Win=64256 Len=0
573	31.355524	servidor	192.168.1.110	TCP	4150	7000 → 62910 [PSH, ACK] Seq=1 Ack=12 Win=64256 Len=4096 [TCP PDU reassembled in 575]
574	31.355543	servidor	192.168.1.110	TCP	7354	7000 → 62910 [PSH, ACK] Seq=4097 Ack=12 Win=64256 Len=7300 [TCP PDU reassembled in 575]
575	31.355556	servidor	192.168.1.110	Gryphon	2974	
576	31.355954	192.168.1.110	servidor	TCP	54	62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2087936 Len=0
577	31.355963	servidor	192.168.1.110	Gryphon	14654	- Invalid -- Invalid -
578	31.355967	servidor	192.168.1.110	Gryphon	14654	- Invalid -
579	31.355954	192.168.1.110	servidor	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2092032 Len=0
580	31.355954	192.168.1.110	servidor	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2096128 Len=0
581	31.355954	192.168.1.110	servidor	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2100224 Len=0
582	31.355954	192.168.1.110	servidor	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2102272 Len=0
583	31.356253	192.168.1.110	servidor	TCP	54	62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2073088 Len=0
584	31.356259	servidor	192.168.1.110	Gryphon	29254	- Invalid -
585	31.356263	servidor	192.168.1.110	TCP	2974	7000 → 62910 [PSH, ACK] Seq=72717 Ack=12 Win=64256 Len=2920 [TCP PDU reassembled in 586]

```

586 31.356265 servidor 192.168.1.110 Gryphon 26334 - Invalid --
Invalid -
587 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2077184 Len=0
588 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2081280 Len=0
589 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2085376 Len=0
590 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2089472 Len=0
591 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2093568 Len=0
592 31.356253 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2097664 Len=0
593 31.356274 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2101760 Len=0
594 31.356563 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=12 Ack=101917 Win=2043392 Len=0
595 31.356572 servidor 192.168.1.110 Gryphon 35094 - Invalid --
Invalid -
596 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2048000 Len=0
597 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2052096 Len=0
598 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2056192 Len=0
599 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2060288 Len=0
600 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2064384 Len=0
601 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2068480 Len=0
602 31.356563 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2072576 Len=0
603 31.356593 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2076672 Len=0
604 31.356593 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2080768 Len=0
605 31.356593 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2084864 Len=0
606 31.356673 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2088960 Len=0
607 31.356676 servidor 192.168.1.110 TCP 22842 7000 → 62910 [PSH,
ACK] Seq=136957 Ack=12 Win=64256 Len=22788 [TCP PDU reassembled in 611]
608 31.356673 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2093056 Len=0
609 31.356673 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2097152 Len=0
610 31.356673 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2101248 Len=0
611 31.356715 servidor 192.168.1.110 Gryphon 23414 - Invalid -
612 31.356816 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=12 Ack=136957 Win=2066176 Len=0
613 31.356822 servidor 192.168.1.110 Gryphon 43854 - Invalid -

```

614 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2071296 Len=0
615 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2075392 Len=0
616 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2079488 Len=0
617 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2083584 Len=0
618 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2087680 Len=0
619 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2091776 Len=0
620 31.356816 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2095872 Len=0
621 31.356837 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2099968 Len=0
622 31.356837 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=136957 Win=2102272 Len=0
623 31.356855 servidor 192.168.1.110 TCP 485 7000 → 62910 [FIN, PSH,
ACK] Seq=226905 Ack=12 Win=64256 Len=431
624 31.356898 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=12 Ack=183105 Win=2056192 Len=0
625 31.356898 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2060288 Len=0
626 31.356898 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2064384 Len=0
627 31.356898 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2068480 Len=0
628 31.356898 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2072576 Len=0
629 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2076672 Len=0
630 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2080768 Len=0
631 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2084864 Len=0
632 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2088960 Len=0
633 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2093056 Len=0
634 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=183105 Win=2097152 Len=0
635 31.357037 192.168.1.110 servidor TCP 54 62910 → 7000 [ACK]
Seq=12 Ack=227337 Win=2052864 Len=0
636 31.357037 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2056960 Len=0
637 31.357057 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2061056 Len=0
638 31.357057 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2065152 Len=0
639 31.357057 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2069248 Len=0
640 31.357058 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2073344 Len=0

```

641 31.357071 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2077440 Len=0
642 31.357087 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2081536 Len=0
643 31.357089 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2085632 Len=0
644 31.357112 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2089728 Len=0
645 31.357114 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2093824 Len=0
646 31.357131 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2097920 Len=0
647 31.357152 192.168.1.110 servidor TCP 54 [TCP Window Update]
62910 → 7000 [ACK] Seq=12 Ack=227337 Win=2102016 Len=0
650 34.163277 192.168.1.110 servidor TCP 54 62910 → 7000 [FIN, ACK]
Seq=12 Ack=227337 Win=2102016 Len=0
651 34.163311 servidor 192.168.1.110 TCP 54 7000 → 62910 [ACK]
Seq=227337 Ack=13 Win=64256 Len=0

```

Reconstruimos el diálogo en el primer paquete que muestra este filtro → Analyze → Follow → TCP Stream

*actividad2.pcap

No.	Time	Source	Destination	Protocol	Length	Info
568	31.351551	192.168.1.110	servidor	TCP	66	62910 → 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
569						↑ Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM
570						↓ Ack=1 Win=2102272 Len=0
571						↑ Ack=1 Win=2102272 Len=11
572						↓ Ack=1 Win=64256 Len=0
573						↑ Ack=12 Win=64256 Len=4096 [TCP PDU reas
574						↓ Ack=12 Win=64256 Len=7300 [TCP PDU r
575						:=14317 Win=2087936 Len=0
576						↑ Ack=12 Win=64256 Len=4096 [TCP PDU reas
577						↓ Ack=12 Win=64256 Len=7300 [TCP PDU r
578						:=14317 Win=2087936 Len=0
579						↑ Ack=12 Win=64256 Len=4096 [TCP PDU reas
580						↓ Ack=12 Win=64256 Len=7300 [TCP PDU r
581						:=14317 Win=2087936 Len=0
582						↑ Ack=12 Win=64256 Len=4096 [TCP PDU reas
583						↓ Ack=12 Win=64256 Len=7300 [TCP PDU r
584						:=14317 Win=2087936 Len=0
585						↑ Ack=12 Win=64256 Len=4096 [TCP PDU reas
586						↓ Ack=12 Win=64256 Len=7300 [TCP PDU r
587						:=14317 Win=2087936 Len=0
588						↑ Ack=12 Win=64256 Len=4096 [TCP PDU reas
589						↓ Ack=12 Win=64256 Len=7300 [TCP PDU r

donde:

- El cliente envía la cadena /lena.gif (en rojo).
- El servidor devuelve un contenido que empieza por GIF87a, que es la firma (“magic bytes”) de un fichero GIF.
- La aplicación está sirviendo un recurso llamado lena.gif, y el cliente lo solicita indicando la ruta. Esto es muy compatible con HTTP (GET de un objeto), pero también podría ser un protocolo propio minimalista, en el que el cliente manda la ruta y el servidor devuelve bytes “a pelo”.

Podemos deducir un patrón: el cliente pide un recurso por “ruta/selector” y el servidor devuelve el fichero en bruto.

Vamos a tirar por hacer un “Decode As... COMO HTTP” de un paquete que ofrezca el puerto 7000 a ver que nos dice: De los dos primeros paquetes de este stream 8:

- Paquete 568: 62910 → 7000 [SYN]

Es el SYN del cliente, donde el puerto “destino” es 7000 pero el paquete se origina desde el puerto

62910.

*actividad2.pcap

tcp.stream == 8

No.	Time	Source	Destination	Protocol	Length	Info
568	31.351551	192.168.1.110	servidor	TCP	66	62910 → 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
569	31.351573	servidor	192.168.1.110	TCP	66	7000 → 62910 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
570	31.351799	192.168.1.110	servidor	TCP	54	62910 → 7000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
571	31.352463	192.168.1.110	servidor	TCP	65	62910 → 7000 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=11
572	31.352469	servidor	192.168.1.110	TCP	54	7000 → 62910 [ACK] Seq=1 Ack=12 Win=64256 Len=0
573	31.355524	servidor	192.168.1.110	TCP	4150	7000 → 62910 [PSH, ACK] Seq=1 Ack=12 Win=64256 Len=4096 [TCP PDU reassembled in 575]
574	31.355543	servidor	192.168.1.110	TCP	7354	7000 → 62910 [PSH, ACK] Seq=4097 Ack=12 Win=64256 Len=7300 [TCP PDU reassembled in 575]
575	31.355556	servidor	192.168.1.110	Gryphon	2974	
576	31.355954	192.168.1.110	servidor			
577	31.355963	servidor	192.168.1.110			
578	31.355967	servidor	192.168.1.110			
579	31.355954	192.168.1.110	servidor			
580	31.355954	192.168.1.110	servidor			
581	31.355954	192.168.1.110	servidor			
582	31.355954	192.168.1.110	servidor			

Wireshark - Decodificar como...

Campo	Valor	Tipo	Predeterminado	Actual
TCP port	62910	Integer, base 10	(none)	(none)

- Paquete 569: 7000 → 62910 [SYN, ACK]

Es el SYN/ACK del servidor, y aquí el paquete sale desde el puerto 7000. Al abrir Decode As desde este paquete, es mucho más probable que Wireshark te muestre TCP port = 7000, que es lo que necesitas para decirle: "trata el puerto 7000 como HTTP".

actividad2.pcap

tcp.stream == 8

No.	Time	Source	Destination	Protocol	Length	Info
568	31.351551	192.168.1.110	192.168.1.33	TCP	66	62910 → 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
569	31.351573	192.168.1.33	192.168.1.110	TCP	66	7000 → 62910 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
570	31.351799	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
571	31.352463	192.168.1.110	192.168.1.33	TCP	65	62910 → 7000 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=11
572	31.352469	192.168.1.33	192.168.1.110	TCP	54	7000 → 62910 [ACK] Seq=1 Ack=12 Win=64256 Len=0
573	31.355524	192.168.1.33	192.168.1.110	TCP	4150	7000 → 62910 [PSH, ACK] Seq=1 Ack=12 Win=64256 Len=4096 (GIF87a)[Malformed Packet]
574	31.355543	192.168.1.33	192.168.1.110	TCP	7354	7000 → 62910 [PSH, ACK] Seq=4097 Ack=12 Win=64256 Len=7300
575	31.355556	192.168.1.33	192.168.1.110	TCP	2974	7000 → 62910 [PSH, ACK] Seq=11397 Ack=12 Win=64256 Len=2920
576	31.355954	192.168.1.110	192.168.1.33			
577	31.355963	192.168.1.33	192.168.1.110			
578	31.355967	192.168.1.33	192.168.1.110			
579	31.355954	192.168.1.110	192.168.1.33			
580	31.355954	192.168.1.110	192.168.1.33			
581	31.355954	192.168.1.110	192.168.1.33			
582	31.355954	192.168.1.110	192.168.1.33			
583	31.356253	192.168.1.110	192.168.1.33			

Wireshark - Decodificar como...

Campo	Valor	Tipo	Predeterminado	Actual
TCP port	7000	Integer, base 10	Gryphon	Gryphon

donde:

- "Protocol", Wireshark etiqueta el paquete 575 como Gryphon. Sin embargo, si investigamos este protocolo Gryphon, es un protocolo de red industrial utilizado principalmente en la automoción (para diagnósticos de vehículos y buses CAN). Es un protocolo binario.

Aunque Wireshark intenta interpretarlo como Gryphon debido al puerto 7000, lo que vimos en la captura donde veíamos palabras legibles en texto plano que decían "Welcome to the... server!", "hello.txt", "RFCs", "lena.gif" indica un protocolo de texto. Luego el protocolo que buscamos no es ese Gryphon.

Como parece que es un protocolo de texto, vamos a intentar hacer un decode del paquete 569, obligando a usar HTTP:

actividad2.pcap

tcp.stream == 8

No.	Time	Source	Destination	Protocol	Length	Info
568	31.351551	192.168.1.110	192.168.1.33	TCP	66	62910 → 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
569	31.351573	192.168.1.33	192.168.1.110	TCP	66	7000 → 62910 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
570	31.351799	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
571	31.352463	192.168.1.110	192.168.1.33	TCP	65	62910 → 7000 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=11
572	31.352469	192.168.1.33	192.168.1.110	TCP	54	7000 → 62910 [ACK] Seq=1 Ack=12 Win=64256 Len=0
573	31.355524	192.168.1.33	192.168.1.110	TCP	4150	7000 → 62910 [PSH, ACK] Seq=1 Ack=12 Win=64256 Len=4096 (GIF87a)[Malformed Packet]
574	31.355543	192.168.1.33	192.168.1.110	TCP	7354	7000 → 62910 [PSH, ACK] Seq=4097 Ack=12 Win=64256 Len=7300
575	31.355556	192.168.1.33	192.168.1.110	TCP	2974	7000 → 62910 [PSH, ACK] Seq=11397 Ack=12 Win=64256 Len=2920
576	31.355954	192.168.1.110	192.168.1.33			
577	31.355963	192.168.1.33	192.168.1.110			
578	31.355967	192.168.1.33	192.168.1.110			
579	31.355954	192.168.1.110	192.168.1.33			
580	31.355954	192.168.1.110	192.168.1.33			
581	31.355954	192.168.1.110	192.168.1.33			
582	31.355954	192.168.1.110	192.168.1.33			

Wireshark - Decodificar como...

Campo	Valor	Tipo	Predeterminado	Actual
TCP port	7000	Integer, base 10	Gryphon	HTTP

Pero no cambia nada:

actividad2.pcap								
Archivo	Edición	Visualización	Ir	Captura	Analizar	Estadísticas	Telefonía	Wireless
tcp.stream == 8								
No.	Time	Source	Destination	Protocol	Length	Info		
568	31.351551	192.168.1.110	192.168.1.33	TCP	66	62910 → 7000 [SYN] Seq=0 Win=64246 Len=0 MSS=1460 WS=256 SACK_PERM		
569	31.351573	192.168.1.33	192.168.1.110	TCP	66	7000 → 62910 [SYN, ACK] Seq=0 Win=64246 Len=0 MSS=1460 SACK_PERM WS=128		
570	31.351799	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0		
571	31.352463	192.168.1.110	192.168.1.33	TCP	65	62910 → 7000 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=11		
572	31.352469	192.168.1.33	192.168.1.110	TCP	54	7000 → 62910 [ACK] Seq=1 Ack=12 Win=64256 Len=0		
573	31.355524	192.168.1.33	192.168.1.110	TCP	4150	7000 → 62910 [PSH, ACK] Seq=1 Ack=12 Win=64256 Len=4096 (GIF87a) [Malformed Packet]		
574	31.355543	192.168.1.33	192.168.1.110	TCP	7354	7000 → 62910 [PSH, ACK] Seq=4097 Ack=12 Win=64256 Len=7300		
575	31.355556	192.168.1.33	192.168.1.110	TCP	2974	7000 → 62910 [PSH, ACK] Seq=11397 Ack=12 Win=64256 Len=2920		
576	31.355954	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2087936 Len=0		
577	31.355963	192.168.1.33	192.168.1.110	TCP	14654	7000 → 62910 [PSH, ACK] Seq=14317 Ack=12 Win=64256 Len=14600		
578	31.355967	192.168.1.33	192.168.1.110	TCP	14654	7000 → 62910 [PSH, ACK] Seq=28917 Ack=12 Win=64256 Len=14600		
579	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2092032 Len=0		
580	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2096128 Len=0		
581	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2100224 Len=0		
582	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2102272 Len=0		

No usa el protocolo http. Veamos que responde google:

Google protocolo que puede tener esa respuesta: iWelcome to the 192.168.1.33:7000 server! fake (i × 🔍

Modo IA Todo Imágenes Vídeos Shopping Vídeos cortos Web Más Herramientas

Se muestran resultados de protocolo que puede tener esa respuesta: **Welcome** to the 192.168.1.33:7000 server! fake (NULL) 0 i fake (NULL) 0 iHello /hello.txt 192.168.1.33 7000 + 1RFCs /dir/ 192.168.1.33 7000 + gLena /lena.gif 192.168.1.33 7000 + 1Quux Mega Server / 142.4.200.132 7000

[Mostrar resultados originales](#)

◆ Visión general creada por IA

Esa respuesta es típica de un servidor FTP (File Transfer Protocol) que lista directorios y archivos, usando comandos como LIST o NLST, respondiendo con códigos numéricos (como '0' o '1' para indicar tipo de archivo) y nombres/rutas, mostrando un listado de recursos disponibles en un servidor, posiblemente un servidor de desarrollo o de pruebas, identificándose como "fake" y listando archivos/directorios con sus rutas y metadatos.

Análisis de la respuesta:

- Welcome to the 192.168.1.33:7000 server! fake (NULL) 0 : Mensaje de bienvenida del servidor FTP, que indica la IP y puerto (192.168.1.33:7000), su nombre ("fake"), y un posible indicador de estado o tipo.

192.168.1.33 - Login Admin - Router Network
As touched on above, 192.168.1.33 is a private IP address that your router uses to distinguish itself on the network, and a...
Router Network

192.168.1.0.1: The Complete Guide to Access, Configuration, and ...
13 oct 2025 — Open your browser and type 192.168. Enter your username and password,...
Ruby-Doc.org

Vamos a probar con FTP. Hacemos un decode de ese paquete 569 con el protocolo FTP:

donde:

- FTP (control) usa comandos tipo: USER, PASS, SYST, PWD, TYPE, RETR, PASV, etc.
 - Normalmente en puerto 21, y además abre canales de datos separados.
 - En el stream:
 - El “request” es solo /lena.gif
 - La “response” empieza por GIF87a (cabecera binaria de un GIF)
 - **Tampoco es FTP.**
 - **Lo que sí sabemos es que: Es un protocolo de petición/respuesta donde el cliente envía un “selector/ruta” y el servidor devuelve el recurso binario.**

Preguntamos a google por protocolos similares a http:

dime protocolos similares a http

Modo IA Todo Vídeos Imágenes Vídeos cortos Shopping Noticias Más ▾ Herr

Visión general creada por IA

Protocolos similares a HTTP incluyen **HTTPS** (HTTP seguro), **WebSockets** para comunicación bidireccional, **FTP** para transferencia de archivos, y otros como **SMTP** (correo), **IMAP/POP3** (recuperación de correo), y **Gopher**, que operan en la capa de aplicación para diferentes propósitos, pero todos facilitan la comunicación en redes, basándose a menudo en TCP/IP.

Protocolos web y de aplicación

- **HTTPS (HTTP Secure)**: Es HTTP con una capa de cifrado (TLS/SSL) para seguridad, esencial para transacciones sensibles.
- **WebSockets**: Permite una comunicación full-duplex persistente sobre una única conexión TCP, ideal para aplicaciones en tiempo real, evolucionando desde HTTP.
- **FTP (File Transfer Protocol)**: Diseñado específicamente para transferir archivos entre computadoras, distinto de HTTP que es para hipertexto.
- **Gopher**: Un sistema de información distribuida más antiguo, con un protocolo ligero muy similar a HTTP, permitiendo la navegación en menús.

donde:

- https no puede ser.
- websocket no puede ser. xxxx
- FTP ya hemos visto que tampoco.

Probamos con Gopher: Un sistema de información distribuida más antiguo, con un protocolo ligero muy similar a HTTP, permitiendo la navegación en menús. Volvemos hacer en ese paquete un “Decode As...

COMO Gopher :

tcp.stream eq 8						
No.	Time	Source	Destination	Protocol	Length	Info
568	31.351551	192.168.1.110	192.168.1.33	TCP	66	62910 → 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
569	31.351573	192.168.1.33	192.168.1.110	TCP	66	7000 → 62910 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
570	31.351799	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
571	31.352463	192.168.1.110	192.168.1.33	Gopher	65	Request: /lena.gif
572	31.352469	192.168.1.33	192.168.1.110	TCP	54	7000 → 62910 [ACK] Seq=1 Ack=12 Win=64256 Len=0
573	31.355524	192.168.1.33	192.168.1.110	Gopher	4150	Response
574	31.355543	192.168.1.33	192.168.1.110	Gopher	7354	Response
575	31.355556	192.168.1.33	192.168.1.110	Gopher	2974	Response
576	31.355954	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2087936 Len=0
577	31.355963	192.168.1.33	192.168.1.110	Gopher	14654	Response
578	31.355967	192.168.1.33	192.168.1.110	Gopher	14654	Response
579	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2092032 Len=0
580	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2096128 Len=0
581	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2100224 Len=0
582	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2102272 Len=0
583	31.356253	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2073088 Len=0
584	31.356259	192.168.1.33	192.168.1.110	Gopher	29254	Response
585	31.356263	192.168.1.33	192.168.1.110	Gopher	2974	Response
586	31.356265	192.168.1.33	192.168.1.110	Gopher	26334	Response
587	31.356253	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2077184 Len=0
588	31.356253	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2081280 Len=0
589	31.356253	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2085376 Len=0
590	31.356253	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2089472 Len=0
591	31.356253	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2093568 Len=0
592	31.356253	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2097664 Len=0
593	31.356274	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2101760 Len=0
594	31.356563	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2043392 Len=0
595	31.356572	192.168.1.33	192.168.1.110	Gopher	35094	Response
596	31.356563	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2048000 Len=0
597	31.356563	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2052096 Len=0
598	31.356563	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2056192 Len=0
599	31.356563	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=101917 Win=2056192 Len=0

donde:

- **Wireshark disecciona el tráfico como Gopher, y el comportamiento observado (cliente envía un selector /lena.gif y servidor devuelve bytes que empiezan por GIF87a) encaja exactamente con Gopher.**

Volvemos a aplicar el filtro: `tcp.port == 7000`:

tcp.port == 7000						
No.	Time	Source	Destination	Protocol	Length	Info
550	23.659351	192.168.1.110	192.168.1.33	Gopher	56	Request: [Directory list]
551	23.659418	192.168.1.33	192.168.1.110	TCP	54	7000 → 62897 [ACK] Seq=1 Ack=3 Win=64256 Len=0
552	23.662193	192.168.1.33	192.168.1.110	Gopher	111	Response
553	23.662161	192.168.1.33	192.168.1.110	Gopher	220	Response
554	23.662373	192.168.1.110	192.168.1.33	TCP	54	62897 → 7000 [ACK] Seq=3 Ack=225 Win=2102016 Len=0
555	23.662554	192.168.1.110	192.168.1.33	TCP	54	62897 → 7000 [FIN, ACK] Seq=3 Ack=225 Win=2102016 Len=0
556	23.662565	192.168.1.33	192.168.1.110	TCP	54	7000 → 62897 [ACK] Seq=225 Ack=4 Win=64256 Len=0
557	25.852265	192.168.1.110	192.168.1.33	TCP	66	62900 → 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
558	25.852296	192.168.1.33	192.168.1.110	TCP	66	7000 → 62900 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
559	25.852649	192.168.1.110	192.168.1.33	TCP	54	62900 → 7000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
560	25.853468	192.168.1.110	192.168.1.33	Gopher	66	Request: /hello.txt
561	25.853479	192.168.1.33	192.168.1.110	TCP	54	7000 → 62900 [ACK] Seq=1 Ack=13 Win=64256 Len=0
562	25.858682	192.168.1.33	192.168.1.110	Gopher	67	Response
563	25.858754	192.168.1.33	192.168.1.110	TCP	54	7000 → 62900 [FIN, ACK] Seq=14 Ack=13 Win=64256 Len=0
564	25.859028	192.168.1.110	192.168.1.33	TCP	54	62900 → 7000 [ACK] Seq=13 Ack=15 Win=2102272 Len=0
565	25.859976	192.168.1.110	192.168.1.33	TCP	54	62900 → 7000 [FIN, ACK] Seq=13 Ack=15 Win=2102272 Len=0
566	25.859993	192.168.1.33	192.168.1.110	TCP	54	7000 → 62900 [ACK] Seq=15 Ack=14 Win=64256 Len=0
568	31.351551	192.168.1.110	192.168.1.33	TCP	66	62910 → 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
569	31.351573	192.168.1.33	192.168.1.110	TCP	66	7000 → 62910 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
570	31.351799	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=1 Ack=2 Win=2102272 Len=0
571	31.352463	192.168.1.110	192.168.1.33	Gopher	65	Request: /lena.gif
572	31.352469	192.168.1.33	192.168.1.110	TCP	54	7000 → 62910 [ACK] Seq=1 Ack=12 Win=64256 Len=0
573	31.355524	192.168.1.33	192.168.1.110	Gopher	4150	Response
574	31.355543	192.168.1.33	192.168.1.110	Gopher	7354	Response
575	31.355556	192.168.1.33	192.168.1.110	Gopher	2974	Response
576	31.355954	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2087936 Len=0
577	31.355963	192.168.1.33	192.168.1.110	Gopher	14654	Response
578	31.355967	192.168.1.33	192.168.1.110	Gopher	14654	Response
579	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2092032 Len=0
580	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2096128 Len=0
581	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2106224 Len=0
582	31.355954	192.168.1.110	192.168.1.33	TCP	54	[TCP Window Update] 62910 → 7000 [ACK] Seq=12 Ack=14317 Win=2102272 Len=0
583	31.356253	192.168.1.110	192.168.1.33	TCP	54	62910 → 7000 [ACK] Seq=12 Ack=43517 Win=2073088 Len=0
584	31.356259	192.168.1.33	192.168.1.110	Gopher	29254	Response
585	31.356263	192.168.1.33	192.168.1.110	Gopher	2974	Response
586	31.356265	192.168.1.33	192.168.1.110	Gopher	26334	Response

donde:

- Con el filtro `tcp.port == 7000` se observan varias conexiones donde Wireshark ya decodifica Gopher.
- Gopher Request: [Directory list] (listado de directorio/menú).
- Gopher Request: /hello.txt (petición de fichero de texto).
- Gopher Request: /lena.gif (petición de imagen GIF).
- Respuestas “Gopher Response” asociadas.
- **Conclusión: la aplicación usa el protocolo Gopher sobre TCP, pero en un puerto no estándar, el 7000 en vez de 70.**

2. Información intercambiada entre el cliente y el servidor

Vamos a realizar una captura de pantalla en la que se vea la información intercambiada entre el cliente y el servidor:

- Aplicamos filtro: `tcp.port == 7000`.
- Seleccionaremos dos paquetes
 - “Gopher Request: /hello.txt”
 - “Gopher Request: /lena.gif”.
- Analyze → Follow → TCP Stream → Analizamos la información que se intercambia.

Información que intercambia hello.txt:

donde:

- En la parte del cliente: la petición con el nombre del fichero y el lugar donde está: `/hello.txt`.
 - En la parte del servidor: la respuesta del servidor que envía el contenido de ese fichero: `Hello world!`.

Información que intercambia [lena.gif](#):

The screenshot shows a Wireshark capture of a Gopher request. The packet details are as follows:

No.	Time	Source	Destination	Protocol	Length	Info
568	31.351551	192.168.1.110	192.168.1.33	TCP	66	62910 -> 7000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
569	31.351573	192.168.1.33	192.168.1.110	TCP	66	7000 -> 62910 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
570	31.351799	192.168.1.110	192.168.1.33	TCP	54	62910 -> 7000 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
571	31.352463	192.168.1.110	192.168.1.33	Gopher	65	Request: /lena.gif
572	31.352469	192.168.1.33	192.168.1.110	TCP	54	7000 -> 62910 [ACK] Seq=1 Ack=12 Win=64256 Len=0
573	31.355524	192.168.1.33	192.168.1.110			
574	31.355543	192.168.1.33	192.168.1.110			
575	31.355556	192.168.1.33	192.168.1.110			
576	31.355954	192.168.1.110	192.168.1.33			
577	31.355963	192.168.1.33	192.168.1.110			
578	31.355967	192.168.1.33	192.168.1.110			
579	31.355954	192.168.1.110	192.168.1.33			
580	31.355954	192.168.1.110	192.168.1.33			
581	31.355954	192.168.1.110	192.168.1.33			
582	31.355954	192.168.1.110	192.168.1.33			
583	31.356253	192.168.1.110	192.168.1.33			
584	31.356259	192.168.1.33	192.168.1.110			
585	31.356263	192.168.1.33	192.168.1.110			
586	31.356265	192.168.1.33	192.168.1.110			
587	31.356253	192.168.1.110	192.168.1.33			
588	31.356253	192.168.1.110	192.168.1.33			
589	31.356253	192.168.1.110	192.168.1.33			

Details for the selected Gopher request (No. 571):

Field	Value
Request URL	/lena.gif
Protocol	Gopher
Method	Request
HTTP Version	1.1
Content Type	image/gif
Content Length	1000
Content	Binary data (1000 bytes)

Message details for the selected Gopher request (No. 571):

Field	Value
Request URL	/lena.gif
Protocol	Gopher
Method	Request
HTTP Version	1.1
Content Type	image/gif
Content Length	1000
Content	Binary data (1000 bytes)

donde:

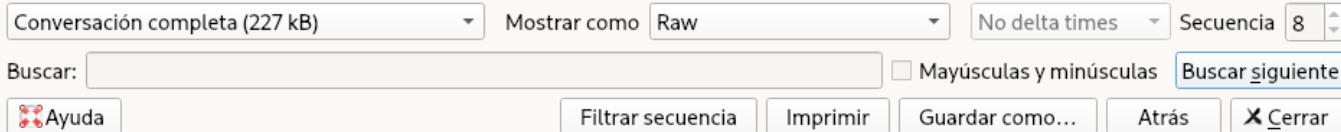
- En la parte del cliente: la petición `/lena.gif`.
 - En la parte del servidor: la respuesta:
`GIF87a.....a].LP.8D.]Y.]a.8H.YY.DL....Ua.0@.,@.0H.
<D.....$<.....YP.....P.4a.HP.4.....Y.@a.4.....qa.4....D].De..m..}.PU..
q..y.8a.La.uu.yi.yy.ye.DUq,a.mai,Y.@P.e].LP.ae.]].HP.8D.4D.]Y.U].DL.UY.@H.@P
.YY....<D....,@. <m L....YPY LH.8H.8m.8a.@D.4..y..m].. Que es el fichero en crudo.`
 - La respuesta del servidor empieza por GIF87a, que es la firma (“magic bytes”) de un fichero GIF. Eso indica que lo que viene a continuación son bytes binarios del propio fichero.

Veamos la foto en crudo: Estando en este paque 571, en la ventana: Analyze → Follow → TCP Stream → Cambiamos “Mostrar como”: Raw.

Wireshark · Seguir secuencia TCP (tcp.stream eq 8) · actividad2.pcap

```
2f6c656e612e6769660d0a
47494638376100020002e70000ca615db64c50993844c65d59ba5d61893848c65959b2444ceed2aeba5561953040952c40ae
4048aa3c44e2c2b6eac6aa91243ceab6aadab6aece5950d29d99d6b2aa500834611848500434f2cea1d29985591040611434
e28981e69571610c34d2898595445d894465e2896dc897dc65055d28571c68579853861aa4c61d67575de7969c67979da79
65b24455712c61d66d61692c59994050d2655dbe4c50c66165d25d5dbe4850a13844a13444ce5d59d2555dba444cce5559b6
4048aa4050ba5959e6ceb2b23c44eac2b6892c4085203c6d204cdebeaaad6595059204c480c384808386d1838611440440434
e29579ea996d5d103ce29179ea8d6deec295c68d85794475d2897d550c3491659df2be85e2ae9dca8575e6b68de27d65be7d
7dde7961c2796d813055c67171de6d61da6d5dda695db66975da655dda615dc2444cba5965aae404ce2cabaaa3c48aa
3846debeb67d28407128487d2440d2a59dc6a5a5501038791c3c6d1c44ca5950591c446914404c043465183c711434855079
59144499759d994c65551040f2ca91debea5590830ca8581d67d79daea1b64c5dda7969de716d916181ca756ddaa6969e2aa
81a14048ca6959953850612855ce595db6444ceecaaaae3c44ae34444c1440a58da14c0c406d203cded998dbe55505d18405d
1440400434ca958d59143c65103459103ca54c55551038550c38e68d6dde8579e6896da54055e28569e28169ca8179d67d71
a56979b65055ca7d79793865c67d75ceaaa1b67979d26d6da56565be796dbe756dd2504cd6695d99556da53844d26159b669
65d6595da53444c65961eed6b6953048a12c40aa444c812c4889243cdabe6daa595612050500c3c652040c65550ea9d7dea
957de28d89551844d2918d551444eac2a1591434591034be8d590c34d28d79ea8969c6505dda7d81d28979dab6a5c68181
de7d71be5055da7d6de2aa99713869eeb67dca7d71da756ddaa6d6dde795dbe7579d6716985558dae757da54850eaaa79b671
71c6696dd66169da6d59ca5050c6656db66571a1384cc64c4cda5d590000002c00000000000200020008fe0059b1ea20b09c
40820257083cc790d58815b0c09c63746e853b59e764c1f2b291d53929af58ad6ab54a8a146a1ea47898b2cac3c856acc8b4
08a7ab06b845ae7ef0934189520857fce6b942e52ae70f17a80eb8387000152a1750974255ea22808b79cae61d08c0150857
73606f1890066480b404d27a0c0030d60080b73a66001850a3c6007d93de0a7b0ba0c684b96fd7eab14b7786beb93374f09d
3143d80c1b8cdf46aed1582f8019936af0957c79c63d613018538eac6fcc5b7b9ad7da7d3b49185dcda833d7104699b60e1d
7b2337662c772fdc1aba82cb9d4b3770dfd96bd702304659b3b1e4750d14efbbf6790de97407104800640d38700480fe2458
440b152d5a0794b9daba0894b4cb3060c083d74216234626cebd1af14a9c89fa60b4d0022302ae700e18b2c8529f81e73804
8b825e48442023ee78c148802b80b1c22b2d80e14e0bee9cd3827e2252f40a43189d73510bb040a18e3a5a1422632119d468
63068568a1ce13eb10e09450f310751e2d4088b74891e181c25d784812105e026b10690010e6004196340418508339063849
005de1196080398b68f72573407ca9e597045479a4390998434b96f7746092498c0c7050417b26b450430f453451451765b4
51471f8534524927a5b4524b2fc534534d37e5b4534f3f053554513f1c95d4524d3d15155554598595565c05e05500609923
1659fe66a1a5165bd2b805975cafdd951700bef5f59772821146d9610324b6d86e900d375965bc5e96d9669775f65968338c
265769a7a5461d6baed505406c75d1d6580db7e526d76e91f5aa0370c205369d6ac801b65c73cb4177dd74ab59879d5ddb75
f7dd93e499879e7aecb9079f7cf4d9879f7efcf907a08004b66020820acac2a083104a386085174aac21871e822822890392
9811432bb6f8628c33de68638e3bf6f8a32b41ba326491e321098492691699a693e24509c494555e99e5965d3a09e6d06396
a9349a6ad6c0a69b54c639a70175de29459e7d1a349040ac28e411a010494828461a71e48547208944924928a9c4924badc0
```

1 client pkt(s), 13 server pkt(s), 1 turn(s).

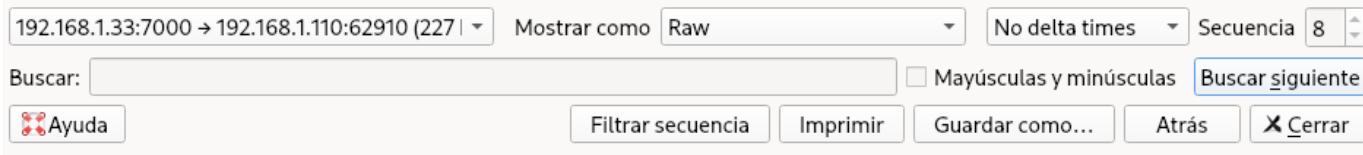


donde:

- Vemos la petición del cliente y del servidor en crudo.

Cambiamos unas opciones para poder guardar la foto:

0 client pkt(s), 13 server pkt(s), 0 turn(s).



donde:

- “Show data as / Mostrar datos como” debe estar en Raw.
- Tipo “Stream/Conversation” o “Direction: 192.168.1.33:7000 --> 192.168.1.110
- Pulsamos el botón guardar y damos un nombre al fichero.

Abrimos el fichero descargado:

