Análisis de la comunicación MCP con Wireshark

– Jorge Luis Lopez 221038

Durante la práctica puse a capturar tráfico en Wireshark mientras realizaba distintas operaciones con mi implementación de MCP. Lo que observé confirma cómo funciona la comunicación entre el anfitrión y los servidores remotos a través de JSON-RPC encapsulado en TLS.

Primero, noté que cada conexión comienza con el handshake TCP de tres vías (SYN, SYN-ACK y ACK). Este proceso garantiza que exista una conexión confiable antes de intercambiar datos. Luego, se establece el handshake TLS, que en mi captura se ve en las líneas donde aparecen los mensajes Client Hello y Server Hello. A partir de ahí, todo el intercambio aparece como TLS Application Data, que corresponde a los mensajes reales de MCP, aunque cifrados.

En este flujo pude identificar tres tipos de mensajes:

- Los mensajes de sincronización, que son los primeros bloques Application Data después del handshake TLS. Aquí viajan los mensajes initialize e initialized, donde cliente y servidor confirman versión y capacidades.

- Las peticiones, que corresponden a los Application Data enviados desde mi máquina (cliente) hacia el servidor. Estas son las llamadas JSON-RPC con el método y parámetros que quiero ejecutar.

- Las respuestas, que son los Application Data que regresan desde el servidor hacia mi cliente, con los resultados o errores de esas peticiones.

En cuanto al análisis por capas:

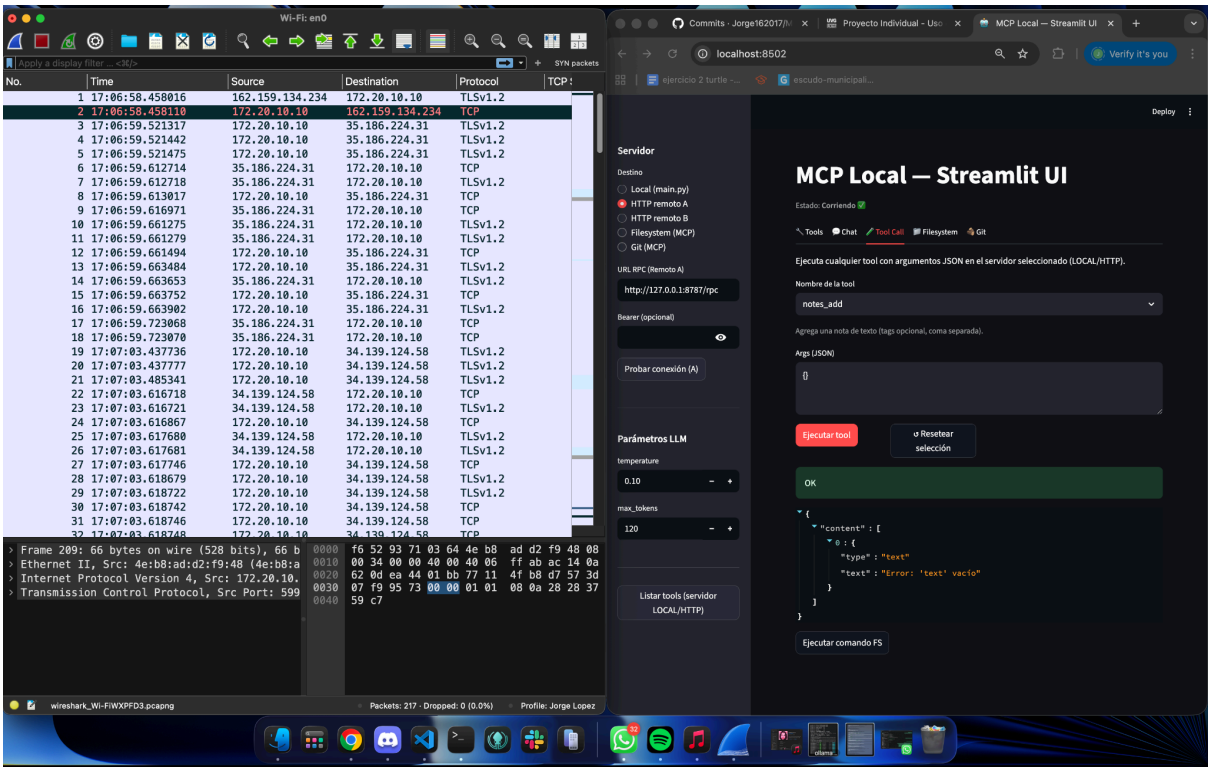
- En la capa de enlace, los paquetes viajan encapsulados en tramas Ethernet/Wi-Fi, utilizando direcciones MAC locales.

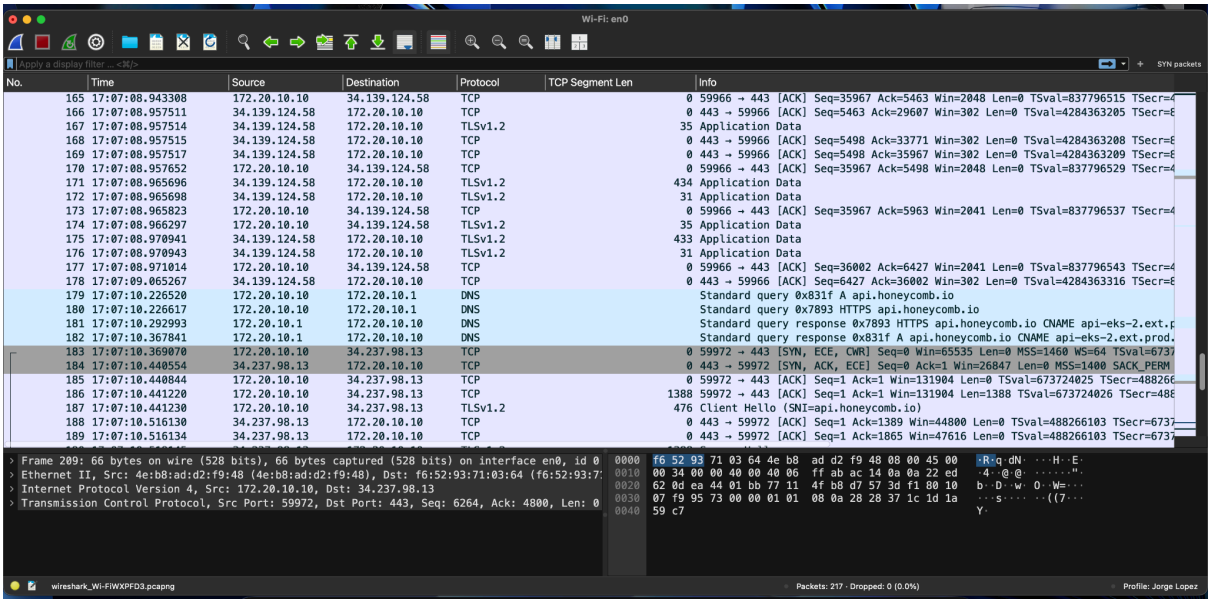
- En la capa de red, se observa el tráfico IP desde mi dirección privada (172.20.10.10) hacia varias direcciones públicas de los servidores MCP, como Google Cloud y Honeycomb.

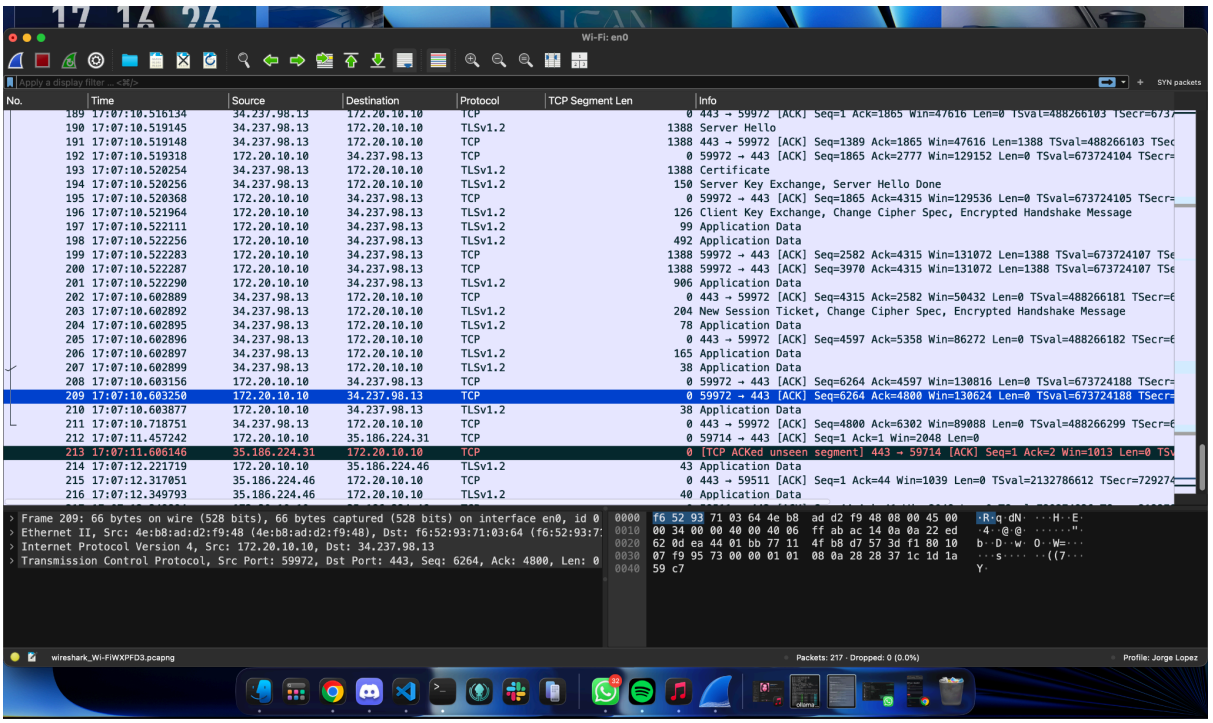
- En la capa de transporte, el protocolo utilizado es TCP sobre el puerto 443, lo cual asegura la entrega confiable y ordenada de los mensajes.

- Finalmente, en la capa de aplicación, se observa TLS (v1.2 y v1.3) que cifra todo el contenido. Dentro de estos mensajes cifrados viajan los mensajes JSON-RPC del MCP, que incluyen la sincronización, las solicitudes y las respuestas.

En conclusión, gracias a Wireshark pude confirmar que MCP utiliza JSON-RPC sobre TLS y que toda la comunicación entre anfitrión y servidor remoto se traduce en secuencias de Application Data cifradas. Aunque no es posible ver el contenido exacto de los mensajes debido al cifrado, la secuencia de solicitudes y respuestas queda claramente reflejada en la captura, validando que la interacción entre el chatbot y los servidores MCP se realizó correctamente.






Wi-Fi: en0
Apply a display filter ... <⌘/>
SYN packets
No. Time Source Destination Protocol TCP Segment Len Info
189 17:07:10.516134 34.237.98.13 172.20.10.10 TCP 0 443 → 59972 [ACK] Seq=1 Ack=1865 Win=47616 Len=0 TSval=488266103 TSecr=673
190 17:07:10.519145 34.237.98.13 172.20.10.10 TLSv1.2 1388 Server Hello
191 17:07:10.519148 34.237.98.13 172.20.10.10 TCP 1388 443 → 59972 [ACK] Seq=1389 Ack=1865 Win=47616 Len=1388 TSval=488266103 TSec
192 17:07:10.519318 172.20.10.10 34.237.98.13 TCP 0 59972 → 443 [ACK] Seq=1865 Ack=2777 Win=129152 Len=0 TSval=673724104 TSecr=
193 17:07:10.520254 34.237.98.13 172.20.10.10 TLSv1.2 1388 Certificate
194 17:07:10.520256 34.237.98.13 172.20.10.10 TLSv1.2 150 Server Key Exchange, Server Hello Done
195 17:07:10.520368 172.20.10.10 34.237.98.13 TCP 0 59972 → 443 [ACK] Seq=1865 Ack=4315 Win=129536 Len=0 TSval=673724105 TSecr=
196 17:07:10.521964 172.20.10.10 34.237.98.13 TLSv1.2 126 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
197 17:07:10.522111 172.20.10.10 34.237.98.13 TLSv1.2 99 Application Data
198 17:07:10.522256 172.20.10.10 34.237.98.13 TLSv1.2 492 Application Data
199 17:07:10.522283 172.20.10.10 34.237.98.13 TCP 1388 59972 → 443 [ACK] Seq=2582 Ack=4315 Win=131072 Len=1388 TSval=673724107 TSe
200 17:07:10.522287 172.20.10.10 34.237.98.13 TCP 1388 59972 → 443 [ACK] Seq=3970 Ack=4315 Win=131072 Len=1388 TSval=673724107 TSe
201 17:07:10.522290 172.20.10.10 34.237.98.13 TLSv1.2 906 Application Data
202 17:07:10.602889 34.237.98.13 172.20.10.10 TCP 0 443 → 59972 [ACK] Seq=4315 Ack=2582 Win=50432 Len=0 TSval=488266181 TSecr=6
203 17:07:10.602892 34.237.98.13 172.20.10.10 TLSv1.2 204 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
204 17:07:10.602895 34.237.98.13 172.20.10.10 TLSv1.2 78 Application Data
205 17:07:10.602896 34.237.98.13 172.20.10.10 TCP 0 443 → 59972 [ACK] Seq=4597 Ack=5358 Win=86272 Len=0 TSval=488266182 TSecr=6
206 17:07:10.602897 34.237.98.13 172.20.10.10 TLSv1.2 165 Application Data
207 17:07:10.602899 34.237.98.13 172.20.10.10 TLSv1.2 38 Application Data
208 17:07:10.603156 172.20.10.10 34.237.98.13 TCP 0 59972 → 443 [ACK] Seq=6264 Ack=4597 Win=130816 Len=0 TSval=673724188 TSecr=
209 17:07:10.603250 172.20.10.10 34.237.98.13 TCP 0 59972 → 443 [ACK] Seq=6264 Ack=4800 Win=130624 Len=0 TSval=673724188 TSecr=
210 17:07:10.603877 172.20.10.10 34.237.98.13 TLSv1.2 38 Application Data
211 17:07:10.718751 34.237.98.13 172.20.10.10 TCP 0 443 → 59972 [ACK] Seq=4800 Ack=6302 Win=89088 Len=0 TSval=488266299 TSecr=6
212 17:07:11.457242 172.20.10.10 35.186.224.31 TCP 0 59714 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0
213 17:07:11.606146 35.186.224.31 172.20.10.10 TCP 0 [TCP ACKed unseen segment] 443 → 59714 [ACK] Seq=1 Ack=2 Win=1013 Len=0 TSv
214 17:07:12.221719 172.20.10.10 35.186.224.46 TLSv1.2 43 Application Data
215 17:07:12.317051 35.186.224.46 172.20.10.10 TCP 0 443 → 59511 [ACK] Seq=1 Ack=44 Win=1039 Len=0 TSval=2132786612 TSecr=729274
216 17:07:12.349793 35.186.224.46 172.20.10.10 TLSv1.2 40 Application Data
Frame 209: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface en0, id 0
Ethernet II, Src: 4e:b8:ad:d2:f9:48 (4e:b8:ad:d2:f9:48), Dst: f6:52:93:71:03:64 (f6:52:93:7
Internet Protocol Version 4, Src: 172.20.10.10, Dst: 34.237.98.13
Transmission Control Protocol, Src Port: 59972, Dst Port: 443, Seq: 6264, Ack: 4800, Len: 0
0000 f6 52 93 71 03 64 4e b8 ad d2 f9 48 08 00 45 00 ·R·q·dN· ···H··E·
0010 00 34 00 00 40 00 40 06 ff ab ac 14 0a 0a 22 ed ·4··@·@· ·····"·
0020 62 0d ea 44 01 bb 77 11 4f b8 d7 57 3d f1 80 10 b··D··w· O··W=···
0030 07 f9 95 73 00 00 01 01 08 0a 28 28 37 1c 1d 1a ···s···· ··((7···
0040 59 c7 Y·
wireshark_Wi-FiWXPFD3.pcapng
Packets: 217 · Dropped: 0 (0.0%)
Profile: Jorge Lopez