**QUIC Wireshark**

The QUIC packets were captured via Wireshark when browsing the internet. Wireshark was activated upon opening a YouTube video and filtering the packets specifically for QUIC packets. A quick Wireshark test before opening YouTube was done to make sure my computer wasn’t using an existing connection that involved QUIC before doing the packet capture on YouTube.

* On Wireshark, the QUIC packets were found using quic in the Wireshark filter.
* A sudden burst of QUIC packets were also received upon loading the YouTube comments section and other recommended videos were loaded.
* When bringing up YouTube’s sidebar, an influx of QUIC packets were detected as well.
* The packet being read in this assignment will be the initial packet used to establish the first connection with YouTube.
* The packet capture was recording while the YouTube video was being played, so more packets were captured as the video kept playing and more sections of the video were buffered and loaded.

Table

Description automatically generated

|  |  |
| --- | --- |
| **Packet Header captured (in Hexadecimal)** | **The type and meaning** |
| 0xca 00 | Converts to 11001010 in binary.  (8-bit Header)  Marks this packet to have the QUIC Long Header, uses fixed bits, that this is the initial packet type, that there are no reserved packets, and that this will be packet number 0. |
| 0x08 | ID Length 8 (in Hexadecimal) |
| 0x35 65 04 a8 4a de 30 24 | Converts to 0011 0101 0110 0101 0000 0100 1010 1000 0100 1010 1101 1110 0011 0000 0010 0100 in binary (Connection ID header)  This is the 64-bit connection ID randomly chosen by the client. This is used to help create the initial packet being captured. |
| 0x00 00 00 01 | Converts to 0000 0000 000 0001 (16-bits) (Version header)  Lists version 1 as a supported version to be used during negotiation. Uncertain why the Version header is 16-bits instead of 32-bits. |
| 0xce | Converts to 1100 1110 in binary  (Packet Number header)  Identifies this packet as the first packet being used in the QUIC exchange.  It is also 8-bits instead of the expected 32-bits. |
| 0x00 4b 19 e5 c7 12 10 46 40 90 5a 2a f0 48 53 37 80 5f 20 7e 25 b7 7a c8 f1 90 5c eb f1 3e 76 3c ca 56 b4 01 29 eb 00 49 a1 28 ab c2 2d 66 03 ce ab 75 89 5c 4e aa a8 | Server Token being used to establish a 0-RTT connection |

A picture containing text

Description automatically generated Text

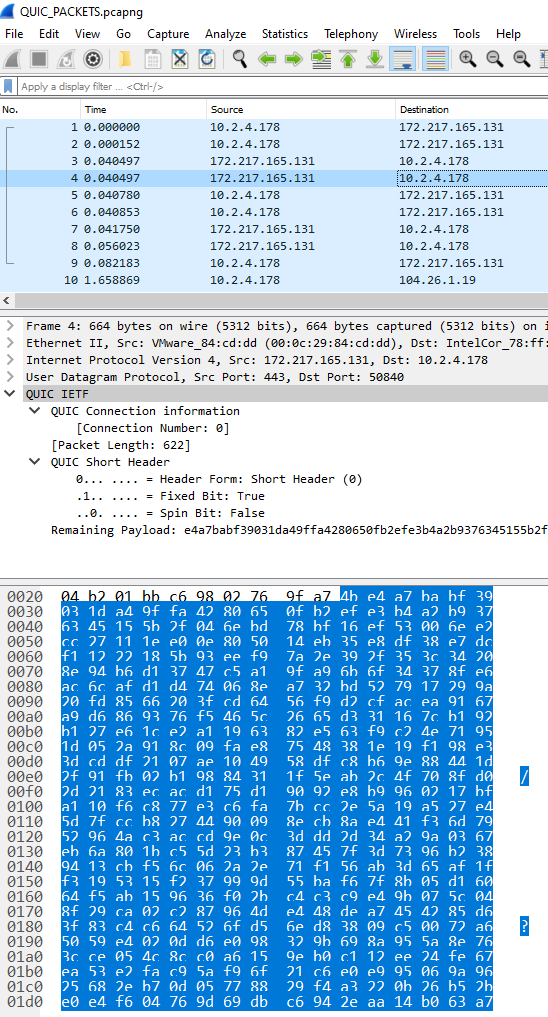
Description automatically generated with medium confidence

The rest is just the QUIC protected payload which was too large to include in the description table. It sends the payload through TLSv1.3 while sending a series of Frames. The Frames found in the initial exchange are PADDING and PING using 0x00 and 0x01 respectively.

Graphical user interface, text, table

Description automatically generated with medium confidence

The packet below was recorded while the YouTube video from before continued playing. After the initial QUIC packet was exchanged between my computer and the server, my computer transitioned to using QUIC short header packets for a more light-weight exchange. The packet shown below is a QUIC packet using the Short Header.



|  |  |
| --- | --- |
| **Data captured (Hexadecimal)** | **Header meaning** |
| 0x4b | 0100 1011 (8-bit Header)  Identifies that this packet will be using the short header since we’ve already established a connection.  Also labels this as packet number 4 |
| **No Connection ID Present** | |
| The rest of the data is just the protected payload contained in the short frames. Unlike the long header, there is no Connection ID or version number found in QUIC’s short header. | |