Network Configuration Record

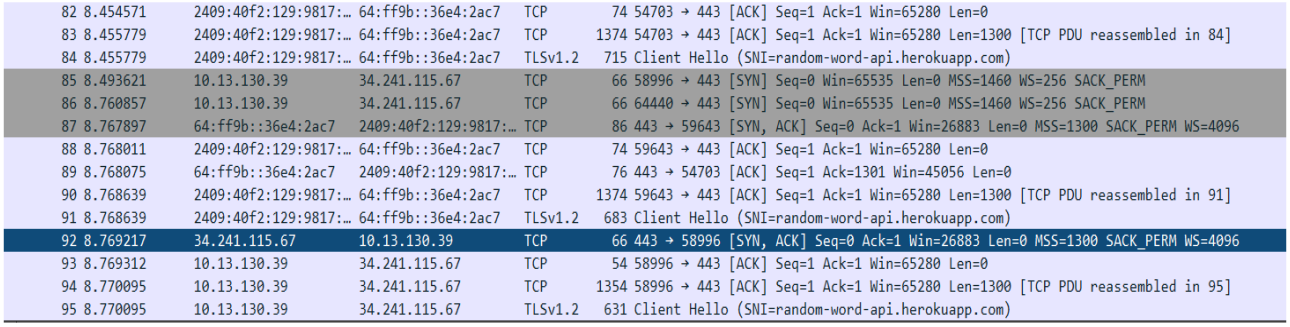
Client IP: 2409:40f2:129:9817:a160:6cf6:d140:1919

Server IP: 2408:6800:4002:811::200a

Port: 443 (HTTPS)

Protocol: TCP + TLS

Find the TCP 3-way handshake



82	8.454571	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TCP	74 54703 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
83	8.455779	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TCP	1374 54703 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=1300 [TCP PDU reassembled in 84]
84	8.455779	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TLSv1.2	715 Client Hello (SNI=random-word-api.herokuapp.com)
85	8.493621	10.13.130.39	34.241.115.67	TCP	66 58996 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
86	8.760857	10.13.130.39	34.241.115.67	TCP	66 64440 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
87	8.767897	64:ff9b::36e4:2ac7	2409:40f2:129:9817:…	TCP	86 443 → 59643 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1300 SACK_PERM WS=4096
88	8.768011	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TCP	74 59643 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
89	8.768075	64:ff9b::36e4:2ac7	2409:40f2:129:9817:…	TCP	76 443 → 54703 [ACK] Seq=1 Ack=1301 Win=45056 Len=0
90	8.768639	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TCP	1374 59643 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=1300 [TCP PDU reassembled in 91]
91	8.768639	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TLSv1.2	683 Client Hello (SNI=random-word-api.herokuapp.com)
92	8.769217	34.241.115.67	10.13.130.39	TCP	66 443 → 58996 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1300 SACK_PERM WS=4096
93	8.769312	10.13.130.39	34.241.115.67	TCP	54 58996 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
94	8.770095	10.13.130.39	34.241.115.67	TCP	1354 58996 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=1300 [TCP PDU reassembled in 95]
95	8.770095	10.13.130.39	34.241.115.67	TLSv1.2	631 Client Hello (SNI=random-word-api.herokuapp.com)

TCP Handshake (Port 443)

SYN → SYN-ACK → ACK

Client IP: 10.13.130.39

TLS Handshake observed:

1. Client Hello

2. Server Hello

3. Certificate

4. Change Cipher Spec

5. Encrypted Handshake Message

74	8.148057	2603:1063:2000::12	2409:40f2:129:9817:…	TLSv1.2	105	Application Data
84	8.455779	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TLSv1.2	715	Client Hello (SNI=random-word-api.herokuapp.com)
91	8.768639	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TLSv1.2	683	Client Hello (SNI=random-word-api.herokuapp.com)
95	8.770095	10.13.130.39	34.241.115.67	TLSv1.2	631	Client Hello (SNI=random-word-api.herokuapp.com)
96	9.112404	2404:6800:4007:804:…	2409:40f2:129:9817:…	TLSv1.2	163	Application Data
97	9.120974	2409:40f2:129:9817:…	2404:6800:4007:804:…	TLSv1.2	109	Application Data
98	9.121123	2409:40f2:129:9817:…	2404:6800:4007:804:…	TLSv1.2	109	Application Data
101	9.154313	64:ff9b::36e4:2ac7	2409:40f2:129:9817:…	TLSv1.2	170	Server Hello
102	9.155552	64:ff9b::36e4:2ac7	2409:40f2:129:9817:…	TLSv1.2	125	Change Cipher Spec, Encrypted Handshake Message
104	9.155924	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TLSv1.2	125	Change Cipher Spec, Encrypted Handshake Message
105	9.156240	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TLSv1.2	860	Application Data
113	9.691502	34.241.115.67	10.13.130.39	TLSv1.2	150	Server Hello
114	9.692288	34.241.115.67	10.13.130.39	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
116	9.692578	10.13.130.39	34.241.115.67	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
119	9.767192	64:ff9b::36e4:2ac7	2409:40f2:129:9817:…	TLSv1.2	170	Server Hello
120	9.768927	64:ff9b::36e4:2ac7	2409:40f2:129:9817:…	TLSv1.2	125	Change Cipher Spec, Encrypted Handshake Message
122	9.769361	2409:40f2:129:9817:…	64:ff9b::36e4:2ac7	TLSv1.2	125	Change Cipher Spec, Encrypted Handshake Message
130	10.238896	64:ff9b::36e4:2ac7	2409:40f2:129:9817:…	TLSv1.2	275	Application Data

DNS Resolution

dns

	Time	Source	Destination	Protocol	Length	Info
41	7.205080	10.13.130.39	10.13.130.119	DNS	89	Standard query 0xab4b HTTPS random-word-api.herokuapp.com
42	7.205563	10.13.130.39	10.13.130.119	DNS	89	Standard query 0x49c1 AAAA random-word-api.herokuapp.com
43	7.206008	10.13.130.39	10.13.130.119	DNS	89	Standard query 0xa68a A random-word-api.herokuapp.com
44	7.277212	10.13.130.39	10.13.130.119	DNS	89	Standard query 0xc54f HTTPS word-edit.officeapps.live.com
45	7.277626	10.13.130.39	10.13.130.119	DNS	89	Standard query 0x02ee AAAA word-edit.officeapps.live.com
46	7.277945	10.13.130.39	10.13.130.119	DNS	89	Standard query 0xd9c7 A word-edit.officeapps.live.com
55	7.369400	10.13.130.119	10.13.130.39	DNS	304	Standard query response 0xc54f HTTPS word-edit.officeapps.live.com CNAME word-edit-geo.wac.trafficmanager.net CNAME word-edit.wac.tra…
56	7.383301	10.13.130.119	10.13.130.39	DNS	279	Standard query response 0xd9c7 A word-edit.officeapps.live.com CNAME word-edit-geo.wac.trafficmanager.net CNAME word-edit.wac.traffic…
57	7.392918	10.13.130.119	10.13.130.39	DNS	303	Standard query response 0x02ee AAAA word-edit.officeapps.live.com CNAME word-edit-geo.wac.trafficmanager.net CNAME word-edit.wac.traf…
60	7.780507	10.13.130.119	10.13.130.39	DNS	173	Standard query response 0x49c1 AAAA random-word-api.herokuapp.com AAAA 64:ff9b::36e4:2ac7 AAAA 64:ff9b::364e:866f AAAA 64:ff9b::22f1:…
63	7.809760	10.13.130.119	10.13.130.39	DNS	154	Standard query response 0xab4b HTTPS random-word-api.herokuapp.com SOA dns1.p03.nsone.net
76	8.179188	10.13.130.119	10.13.130.39	DNS	137	Standard query response 0xa68a A random-word-api.herokuapp.com A 34.241.115.67 A 54.78.134.111 A 54.228.42.199
132	10.256772	10.13.130.39	10.13.130.119	DNS	74	Standard query 0x4e0a HTTPS www.google.com
133	10.257138	10.13.130.39	10.13.130.119	DNS	74	Standard query 0xc2ea AAAA www.google.com

299 26.297000 10.13.130.39 10.13.130.119 DNS 86 Standard query 0x1e0a A waa-pa.clients6.google.com

302 26.376025 10.13.130.119 10.13.130.39 DNS 279 Standard query response 0xbd77 HTTPS common.online.office.com CNAME common-geo.wac.trafficmanager.net CNAME common.wac.trafficmanager.net.wac-0003.wac-dc-msedge.net.wac-0003.wac-msedge.net SOA ns1.wac-msedge.net

DNS Resolution Time

$26.376625 - 26.274435 = 0.102190$ seconds

Comparing API Call Patterns (TLS/HTTP Request Analysis)

When the API or web service was accessed multiple times, the Wireshark capture showed several **TLS handshakes** with packets labeled *Client Hello*, *Server Hello*, and *Change Cipher Spec*, followed by **Application Data** packets.
In the first access, a full handshake occurred indicating the establishment of a new secure session.
However, for subsequent accesses, fewer handshake packets were seen, and mostly **Application Data** was exchanged, suggesting **connection reuse or caching**.

The captured data (TLSv1.2 and TLSv1.3) on port **443** shows that after the initial connection, the browser or client reused the existing secure session to reduce latency.
This confirms that the API used **HTTPS persistent connections and caching** to improve efficiency during repeated API calls.

Web in Action: Building and Tracing an Interactive API-Based Webpage

Wireshark analysis over Wi-Fi network

1. Introduction

This project titled 'BeeSmart - The Spelling Bee Game' is an interactive web-based application built using HTML, CSS, and JavaScript. The project demonstrates how front-end interactivity communicates with APIs and how these interactions can be analyzed using Wireshark to understand the underlying network protocols.

2. APIs Used and Integration

Two public APIs were integrated into the BeeSmart game:

1. Random Word API (https://random-word-api.herokuapp.com/word) – This API provides random English words that the player needs to spell correctly during the game.

2. Dictionary API (https://api.dictionaryapi.dev/api/v2/entries/en/) – After a player attempts a spelling, this API fetches the word's meaning and synonyms, which are then displayed dynamically on the webpage.

The Fetch API was used to make asynchronous GET requests to these APIs. The responses were handled using JavaScript's async/await to update the DOM in real-time, displaying new words and meanings without page reloads.

3. Wireshark Network Captures and Explanation

3.1 DNS Query and Response

The DNS packets show how the browser resolved the Netlify-hosted domain 'dainty-vacherin-c64991.netlify.app' into its IP address (13.215.239.219). The DNS response (as shown in your screenshots) confirms successful resolution before any HTTPS connection could start.

3.2 TCP Three-Way Handshake

Once the DNS resolution was complete, the browser initiated a TCP connection to the server. This was observed through packets containing the flags SYN, SYN-ACK, and ACK. These three packets establish a reliable connection between the client (192.168.0.104) and the Netlify server. This ensures that both ends are ready for data transfer.

3.3 TLS Handshake (HTTPS Encryption)

After the TCP connection, the TLS (Transport Layer Security) handshake took place. This can be seen in the packets labeled 'Client Hello' and 'Server Hello' in your Wireshark screenshots. This process negotiates encryption methods and establishes a secure HTTPS session for all further API and website communication.

3.4 Website and API Requests

Once the TLS handshake was successful, the webpage sent multiple HTTPS requests to the APIs. In Wireshark, this appears as packets sent to api.dictionaryapi.dev and random-word-api.herokuapp.com. Each request returns JSON data, which the JavaScript code processes to update the game display dynamically.

4. Reflections and Analysis

4.1 TCP vs TLS Handshake

The TCP handshake establishes a connection using the SYN, SYN-ACK, and ACK sequence. It ensures reliable communication but does not encrypt the data. The TLS handshake, on the other hand, occurs after TCP and adds an extra security layer, ensuring that all further communication is encrypted and safe. In the BeeSmart project, both were visible in Wireshark, confirming a secure HTTPS connection to the Netlify server.

4.2 DNS Resolution Time

DNS resolution time measures how long it takes for a domain name to be translated into an IP address. In our Wireshark capture, this process took only a few milliseconds, showing efficient DNS lookup. Once resolved, subsequent requests reused the cached IP, reducing latency for future requests.

4.3 Caching and Query Parameters

When the same API was called multiple times, some responses were retrieved faster due to browser caching. However, changing query parameters in the API request forces a fresh fetch from the server, bypassing the cache. This demonstrates how varying parameters can affect network load and overall performance.

4.4 Latency in Wi-Fi vs Mobile Hotspot

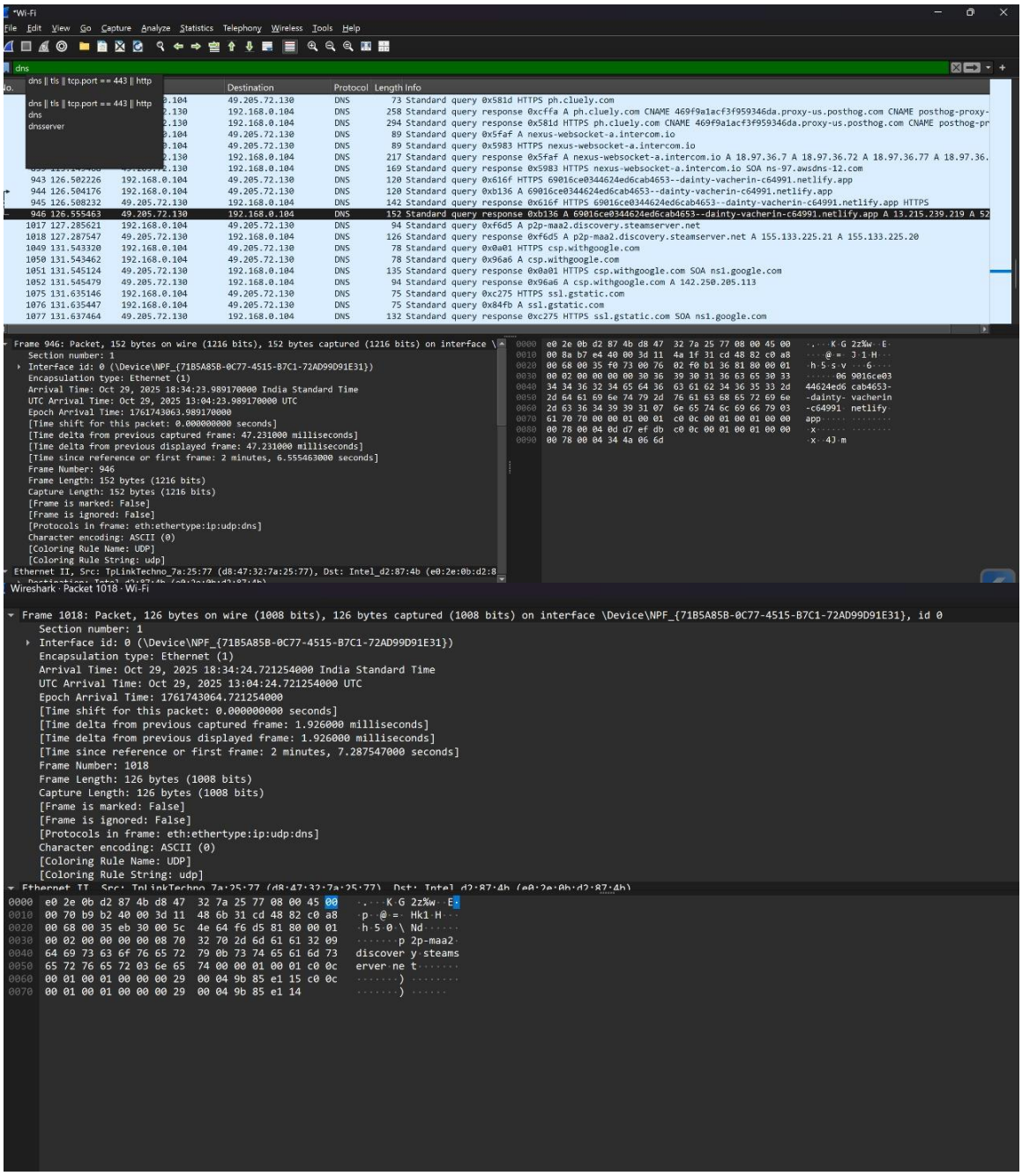
Latency represents the delay between sending a request and receiving a response. On Wi-Fi, latency was minimal (~47ms) because of stable bandwidth and low interference. In contrast, using a mobile hotspot would increase latency due to variable signal strength, network congestion, and cellular routing overhead. Thus, Wi-Fi provides a more stable and faster browsing experience for API-heavy web applications.

5. Conclusion

Through the BeeSmart project, we learned how web pages interact with APIs and how this activity translates into network-level communication captured in Wireshark. The combined use of HTML, CSS, and JavaScript with APIs enabled a dynamic and interactive user experience. The Wireshark analysis reinforced concepts such as DNS resolution, TCP and TLS handshakes, latency, and caching — helping us connect web development with real-world network behavior.

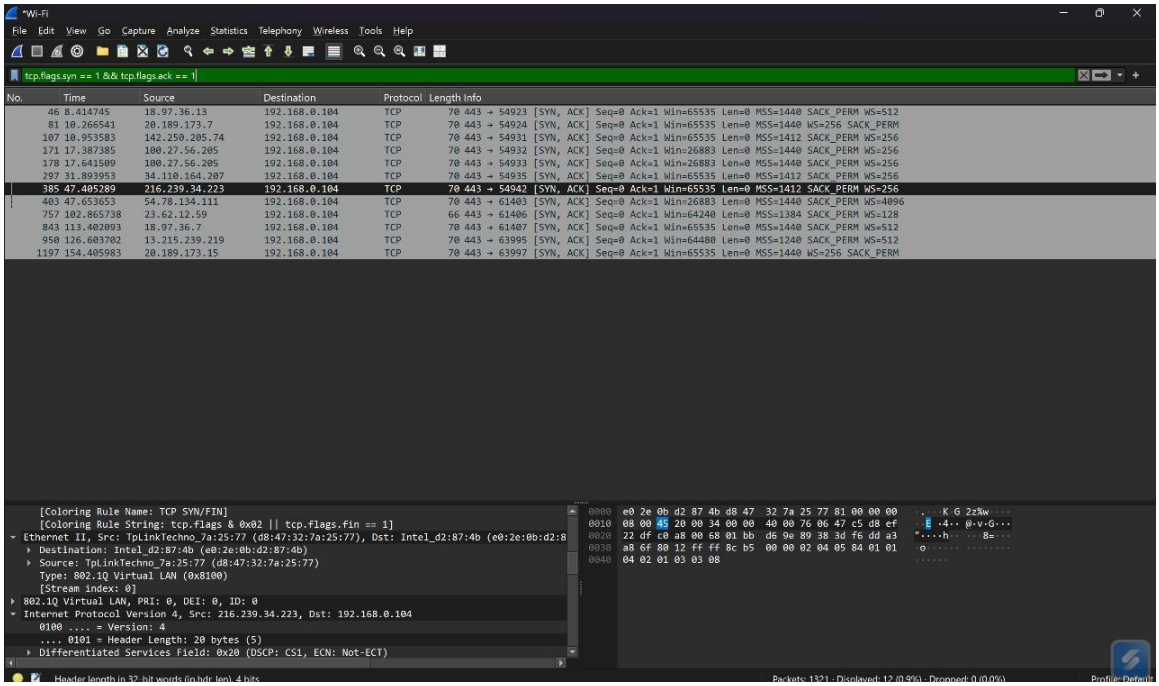
6. Screenshots Explanation

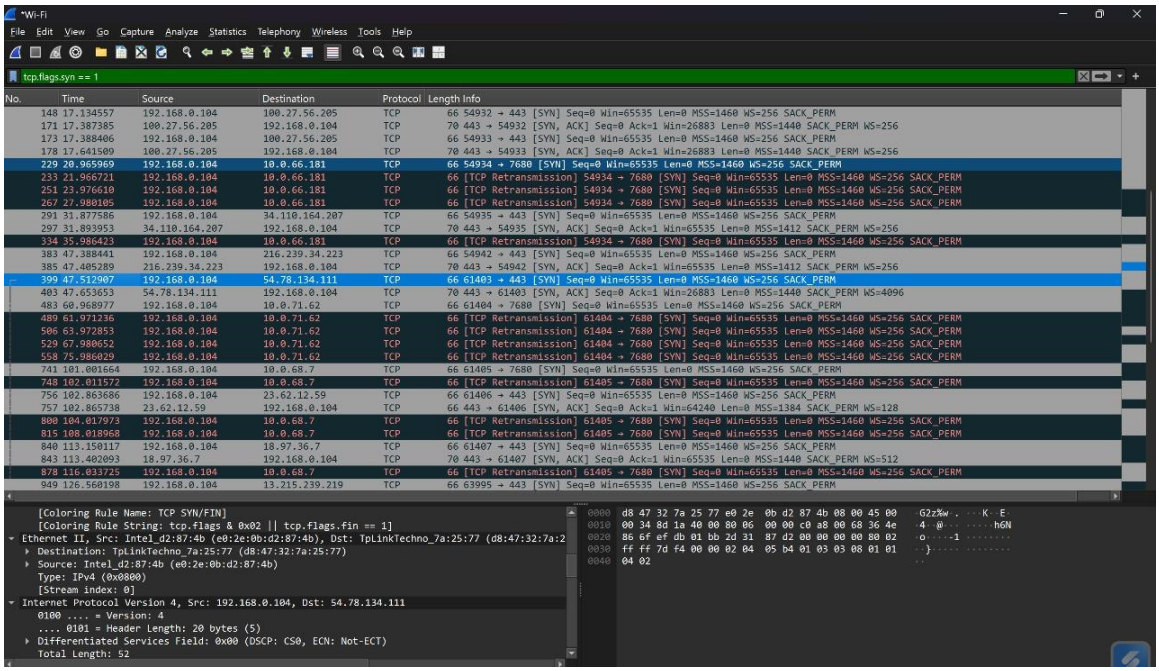
• Screenshot 1 – DNS Response showing the IP address (13.215.239.219) for the Netlify domain.



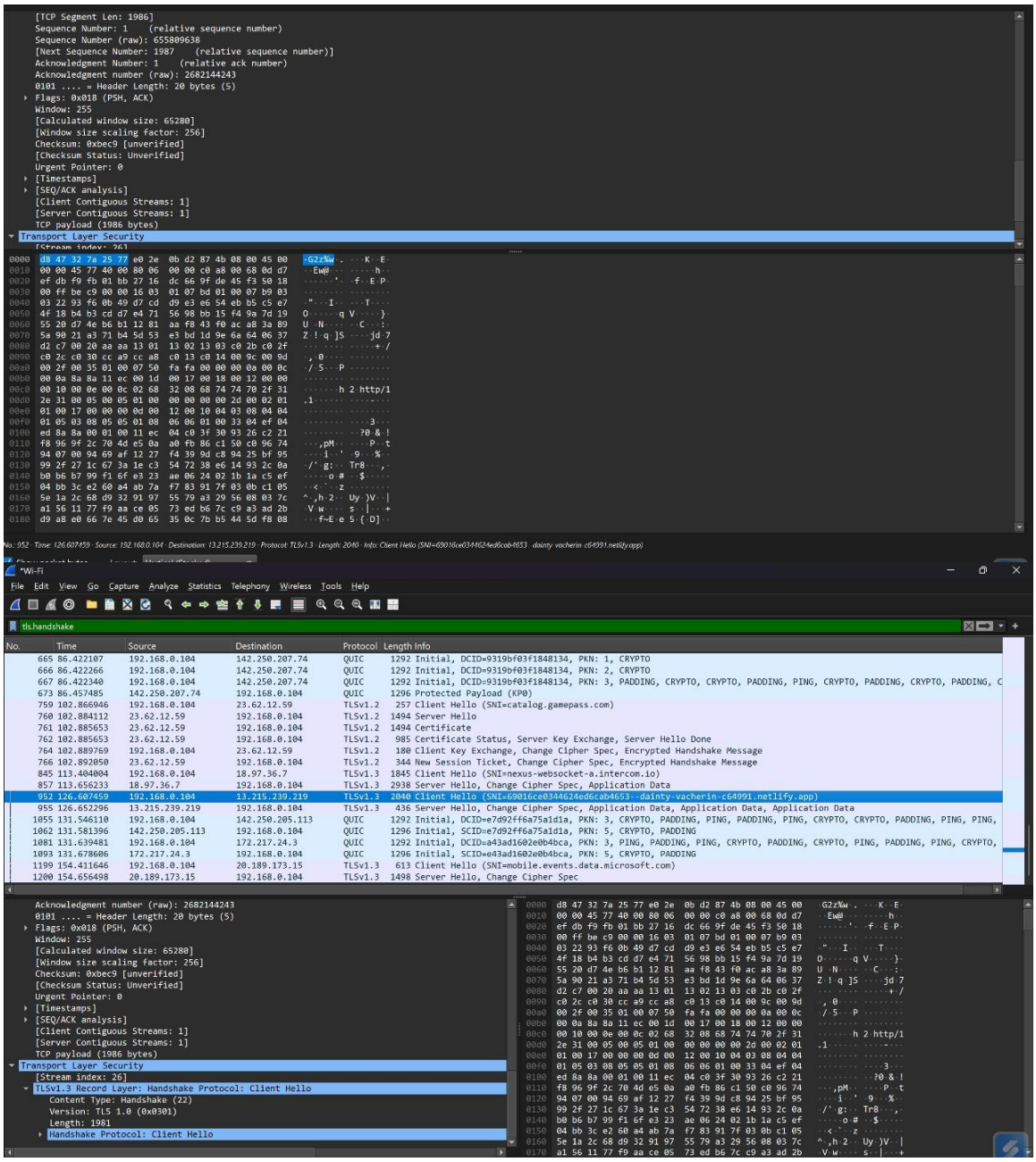
• Screenshot 2 – TCP Handshake showing SYN, SYN-ACK, and ACK packets establishing a

connection.

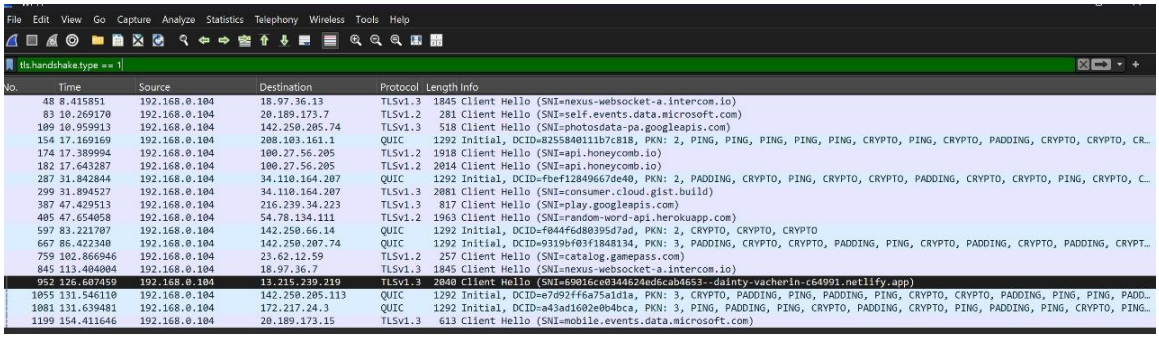




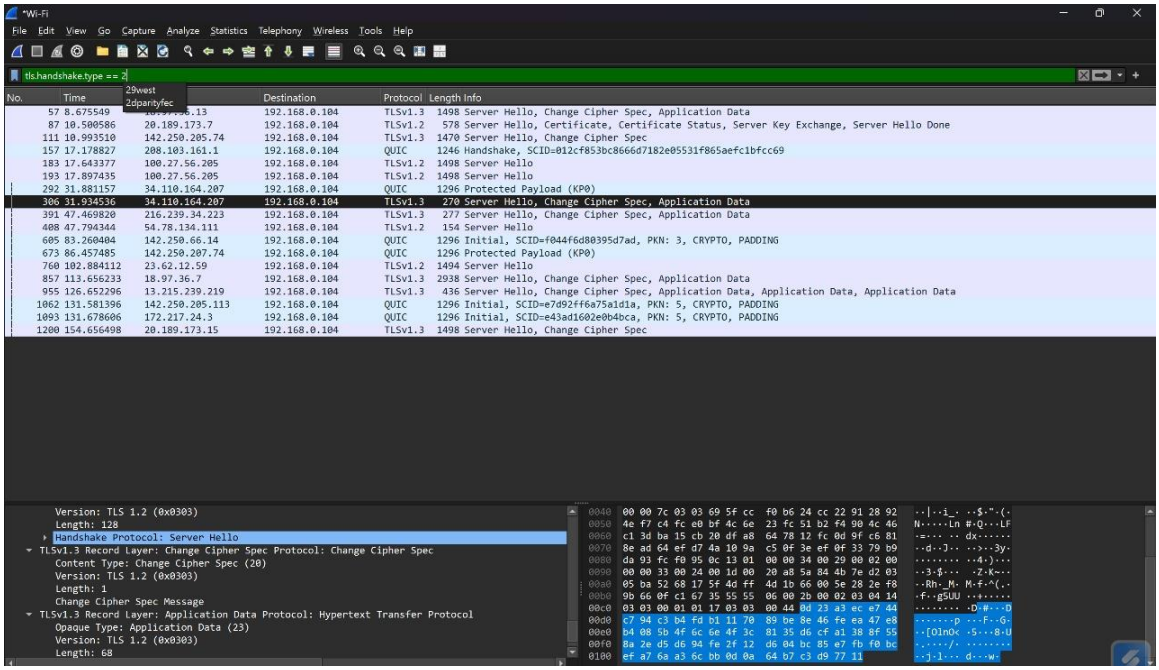
• Screenshot 3 – TLSv1.3 Client Hello and Server Hello packets, confirming secure HTTPS setup.



• Screenshot 4 – TLS Change Cipher Spec and Application Data exchange, showing encrypted communication.



• Screenshot 5 – TCP ACK packets confirming successful data delivery between client and server.



These captures validate that the BeeSmart game successfully initiates secure API calls over HTTPS, demonstrating end-to-end communication between the client browser and the remote API servers.