

Project : Packet Analysis with Wireshark – Analyzing google.com

Objective:

To capture and analyze the network packets generated while accessing **https://google.com**, with a focus on DNS resolution, TCP handshakes, and encrypted HTTPS traffic.

Tools Used:

- **Wireshark** (Network protocol analyzer)
- **Web Browser:** Google Chrome
- **Website Visited:** <https://google.com>
- **Operating System:** Windows

Steps Followed:

1. **Installed Wireshark** from <https://www.wireshark.org> and launched the tool.
2. **Selected Network Interface** (Wi-Fi) to capture live traffic.
3. **Started Packet Capture** and visited <https://google.com> in the browser.
4. **Applied Filters:**
 - dns → To capture the DNS request resolving google.com.
 - tcp.port == 443 → To view HTTPS traffic.
 - tls → To view the TLS handshake packets.

Analyzed Captured Packets:

- **DNS Lookup:**
 - Observed DNS query and response for resolving google.com.
- **TCP Handshake:**
 - Captured 3-way TCP handshake between my system and Google's server.
- **TLS Handshake:**
 - Observed the start of secure communication.
 - Captured "Client Hello", "Server Hello", and key exchange mechanisms.

Results:

1. **Filter: dns**

No.	Time	Source	Destination	Protocol	Length	Info
15	5.860464			DNS	109	Standard query 0x43bd AAAA clientservices.googleapis.com
16	5.860774			DNS	109	Standard query 0x4cd8 A clientservices.googleapis.com
17	5.860997			DNS	109	Standard query 0xcaa9 HTTPS clientservices.googleapis.com
18	5.861580			DNS	99	Standard query 0xc751 AAAA accounts.google.com
19	5.861879			DNS	99	Standard query 0x6358 A accounts.google.com
20	5.863456			DNS	99	Standard query 0x299c HTTPS accounts.google.com
21	5.912527			DNS	166	Standard query response HTTPS clientservic
22	5.912527			DNS	149	Standard query response HTTPS accounts
23	5.912527			DNS	115	Standard query response \ account
24	5.912527			DNS	125	Standard query response \ clientser
25	5.926321			DNS	127	Standard query response account
27	6.037614			DNS	94	Standard query www.google.com
28	6.037987			DNS	94	Standard query www.google.com
29	6.038269			DNS	94	Standard query HTTPS www.google.com
34	6.086272			DNS	122	Standard query response
35	6.092907			DNS	110	Standard query response
36	6.092907			DNS	119	Standard query response HTTPS www.google.com HTTPS
156	6.674736			DNS	94	Standard query lh3.google.com

Frame 15: 109 bytes on wire (872 bits), 109 bytes captured (872 bits) on interface \Device\NPF... id 0

Ethernet II, Src: Intel..., Dst: NokiaSolution...

Internet Protocol Version 6, Src: ..., Dst: ...

User Datagram Protocol, Src Port: 60211, Dst Port: 53

Domain Name System (query)

This filter displays all DNS traffic captured during browsing. It includes DNS queries and responses, showing how the domain name google.com was resolved to its IP address.

◆ 2. Filter: tcp.port == 443

No.	Time	Source	Destination	Protocol	Length	Info
11	5.191082			TCP	55	61474 → [ACK] Seq= Ack= Win= Len=
12	5.219645			TCP	54	443 → [ACK] Seq= Ack= Win= Len=
13	5.219689			TCP	54	[TCP Dup ACK 1181] [TCP ACKed unseen segment] [ACK] Seq= Ack= Win= Len=
14	5.244345			TCP	66	[TCP Previous segment not captured] 443 → [ACK] Seq= Ack=2 Win= Len= SLE=1 SRE=0
26	5.937189			TCP	86	5832 → 443 [SYN] Seq=0 Win=64800 Len=0 MSS=1440 WS=256 SACK_PERM
30	6.060784			TCP	86	443 → 5832 [SYN, ACK] Seq=0 Ack=1 Win= Len=0 MSS=1230 SACK_PERM
31	6.060876			TCP	74	5832 → 443 [ACK] Seq= in= out=
32	6.061911			TCP	1304	5832 → 443 [ACK] Seq= in= out=
33	6.061911			TLSv1.3	573	Client Hello (SNI=acco le.com)
37	6.093727			TCP	86	5833 → 443 [SYN] Seq=0 Len=0 MSS=
38	6.139935			TCP	86	443 → 5833 [SYN, ACK] Seq=0 Ack=1 Win= Len=0
39	6.140016			TCP	74	5833 → 443 [ACK] Seq=1 Win= Len=0
40	6.140635			TCP	1304	5833 → 443 [ACK] Seq=1 Win= Len= [TCP PDU reassembled in 41]
41	6.140635			TLSv1.3	664	Client Hello (google.com)
42	6.176440			TCP	86	[TCP Dup ACK 3081] 443 →
43	6.176440			TCP	74	443 → 5832 [ACK] Seq=1
44	6.181644			TLSv1.3	1294	Server Hello
45	6.181644			TLSv1.3	1294	Change Cipher

Frame 14: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF... id 0

Ethernet II, Src: Nokia..., Dst: Intel...

Internet Protocol Version 4, Src: ..., Dst: ...

Transmission Control Protocol, Src Port: 443, Dst Port: 61474, Seq: 1, Ack: 2, Len: 0

This filter highlights TCP packets using port 443, indicating encrypted HTTPS traffic. It shows the secure communication channel established between the browser and google.com.

◆ 3. Filter: tls

The image shows a Wireshark packet capture with the filter 'tls' applied. The packet list on the left shows packets 2431 through 2546. The packet details pane on the right shows the selected packet (2431) with the following structure:

- Frame 33: 573 bytes on wire (4584 bits), 573 bytes captured (4584 bits) on interface \Device\NPF_{...}, id 0
- Ethernet II, Src: Intel_..., Dst: NokiaSolution...
- Internet Protocol Version 4, Src: ..., Dst: 303:c04::54
- Transmission Control Protocol, Src Port: 5..., Dst Port: 443, Seq: ..., Ack: 1, Len: 499
- [2 Reassembled TCP Segments]
- Transport Layer Security

The packet bytes pane on the right shows the raw data of the selected packet, with a hex dump and ASCII representation.

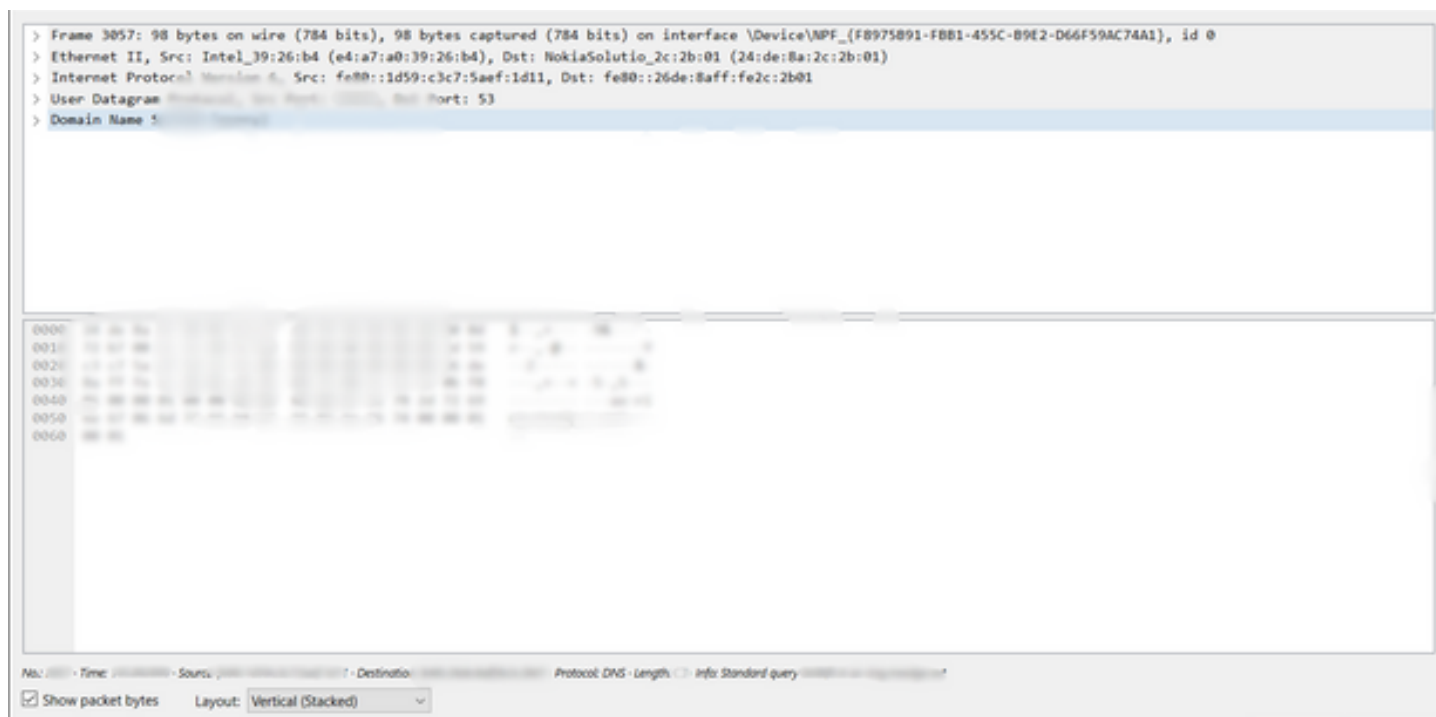
This filter displays packets related to the TLS handshake process. It includes details like Client Hello and Server Certificate, confirming secure encrypted communication with google.com.

-DNS Packet Details for google.com:

The image shows a Wireshark packet capture with the filter 'dns' applied. The packet list on the left shows packets 3054 through 3071. The packet details pane on the right shows the selected packet (3057) with the following structure:

- Frame 33: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface \Device\NPF_{...}, id 0
- Ethernet II, Src: Intel_..., Dst: NokiaSolution...
- Internet Protocol Version 4, Src: ..., Dst: 303:c04::54
- User Datagram Protocol, Src Port: ..., Dst Port: 53
- Domain Name System (query)

The packet bytes pane on the right shows the raw data of the selected packet, with a hex dump and ASCII representation.



This screenshot shows the breakdown of a DNS packet captured during access to google.com.

Screenshots Included:

- DNS query and response for google.com
- TCP 3-way handshake packets
- TLS handshake sequence
- Encrypted HTTPS traffic packets

Key Observations:

- **DNS resolution** is always the first step in accessing a domain, regardless of the protocol.
- **TLS handshake** ensures secure encryption between client and server.
- **All application data was encrypted**, confirming Google's HTTPS implementation is secure.

Conclusion:

This project demonstrated how HTTPS traffic is handled on a network level. Even though actual data isn't visible due to encryption, the initial DNS and TLS handshake packets still provide meaningful insight. This exercise strengthened my understanding of:

- DNS resolution process
- TCP and TLS handshake protocols
- The importance of HTTPS for secure communication

By analyzing google.com, I learned how modern websites ensure secure data transmission and how tools like Wireshark help in observing the underlying protocols.