

# Report: Capturing and Analyzing DNS Traffic with Wireshark

## By Matthew Miller

This report details how I captured and analyzed DNS (Domain Name System) traffic using Wireshark on my Windows system, focusing on demonstrating my ability to use this tool for network traffic analysis.

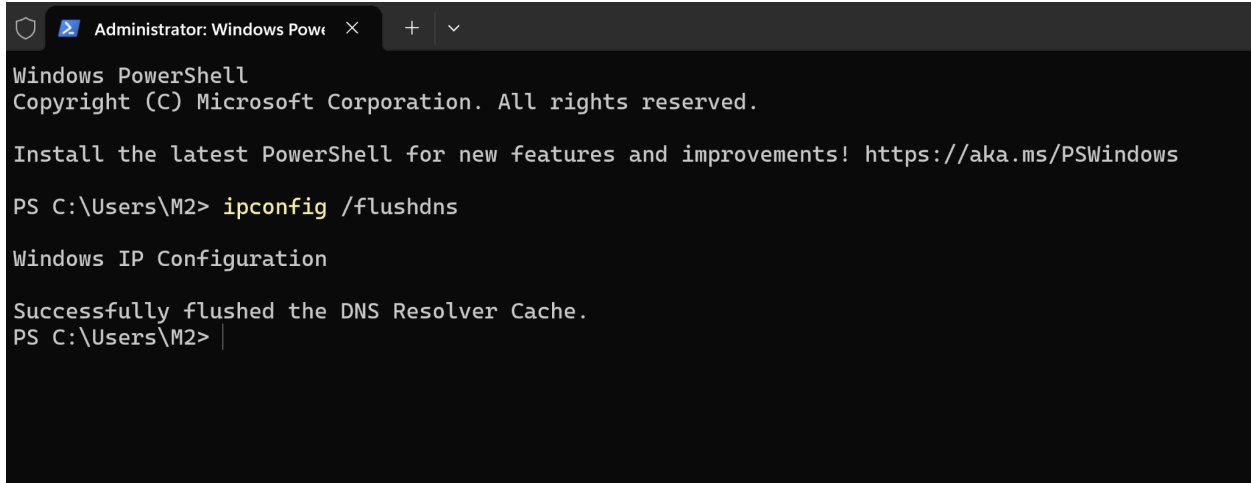
---

### Step 1: Set Up My Environment

#### 1.1 Cleared the DNS Cache

Clearing the DNS cache ensured that my system had to query the DNS server again, generating fresh DNS traffic. This step was important to capture relevant DNS packets.

- **What I Did:**
  1. I opened Command Prompt as an administrator.
  2. I typed `ipconfig /flushdns` and pressed Enter to clear the DNS cache. This ensured that any DNS queries I made would generate new traffic, which I could capture and analyze.

A screenshot of a Windows PowerShell terminal window titled "Administrator: Windows PowerShell". The window shows the command prompt at "PS C:\Users\M2>". The user has entered the command "ipconfig /flushdns". The output shows "Windows IP Configuration" followed by "Successfully flushed the DNS Resolver Cache." and the prompt "PS C:\Users\M2>".

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\M2> ipconfig /flushdns

Windows IP Configuration

Successfully flushed the DNS Resolver Cache.
PS C:\Users\M2>
```

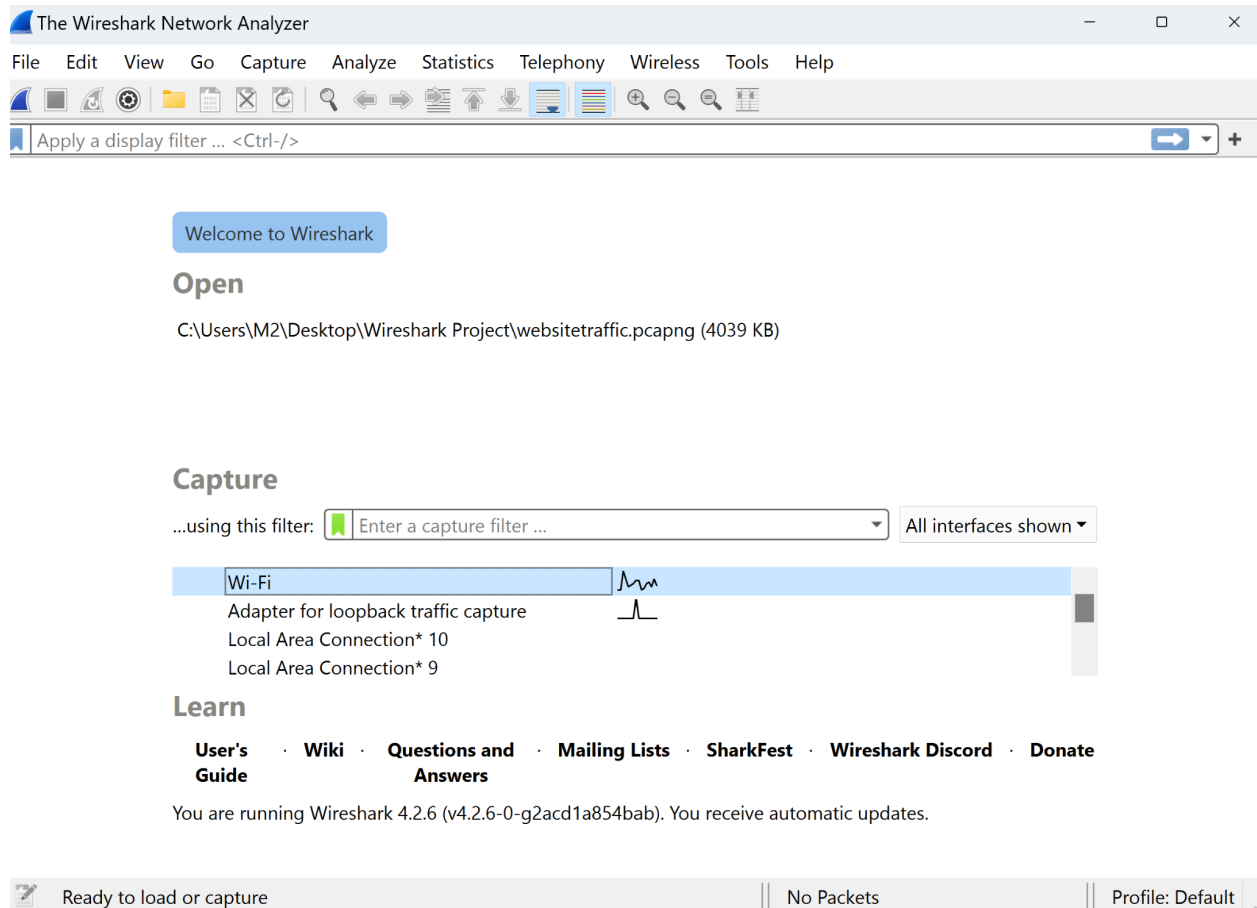
### Step 2: Start Wireshark and Set Up Capture Filters

#### 2.1 Opened Wireshark and Selected the Network Interface

I launched Wireshark to start capturing real-time network traffic. Selecting the correct network interface was crucial to ensure I captured traffic on the active connection.

- **What I Did:**

1. I opened Wireshark on my Windows machine.
2. From the list of available network interfaces, I selected the WiFi connection. I clicked on the selected interface to begin capturing traffic.

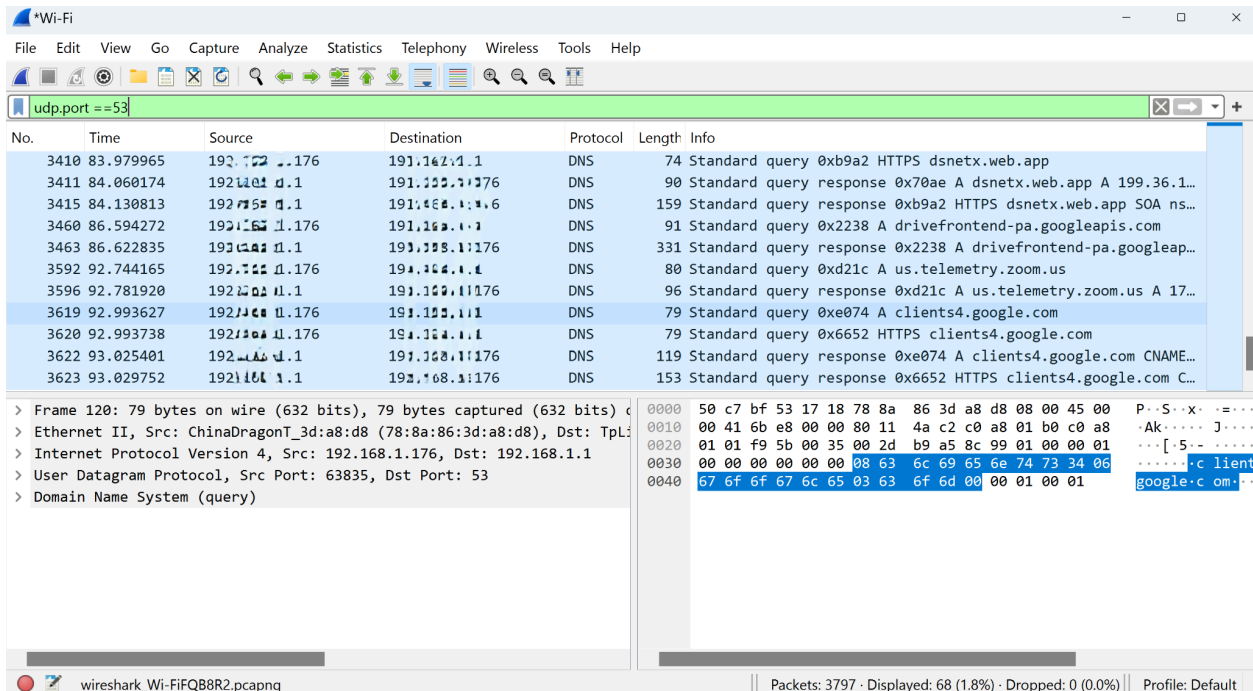


## 2.2 Set a Capture Filter for DNS Traffic

To focus specifically on DNS traffic, I set a capture filter. This allowed me to limit the traffic captured by Wireshark, making it easier to analyze later.

- **What I Did:**

1. In the capture filter bar, I typed `udp.port == 53` and pressed Enter.
2. This filter ensured that Wireshark captured only DNS traffic, which uses UDP on port 53.

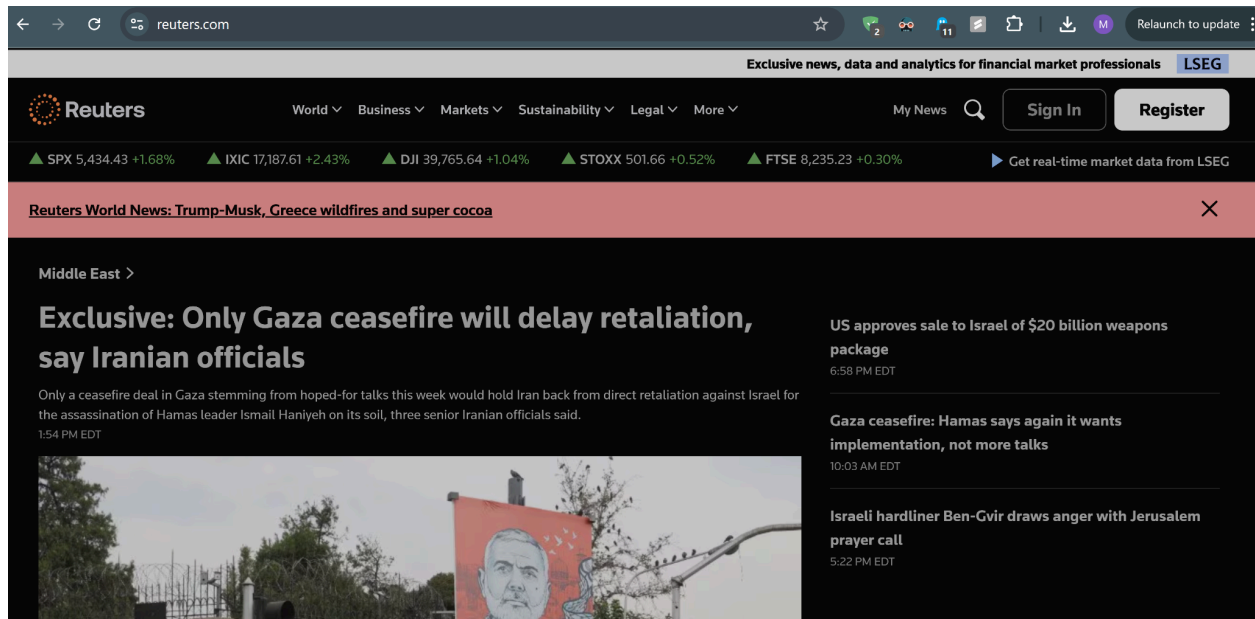


### Step 3: Initiate DNS Traffic

#### 3.1 Generated DNS Traffic by Visiting a Website

To generate DNS traffic, I visited a website using Chrome. This action prompted my computer to resolve the domain name into an IP address, generating DNS packets that I captured with Wireshark.

- What I Did:**
  - I opened Chrome on my Windows computer.
  - I typed a domain name (www.reuters.com) into the address bar and pressed Enter.
  - I allowed the page to load completely, ensuring that DNS queries were sent and responses were received.



### 3.2 Stopped the Capture in Wireshark

After the website had loaded and DNS traffic had been generated, I stopped the Wireshark capture to analyze the packets.

- **What I Did:**
  1. I returned to the Wireshark window.
  2. I clicked the red square button (Stop Capture) in the toolbar to halt the capture.

### Step 4: Analyze DNS Traffic in Wireshark

#### 4.1 Applied a Display Filter for DNS Traffic

To focus on DNS packets, I applied a display filter. This allowed me to filter out all other captured traffic, making it easier to analyze the DNS queries and responses.

- **What I Did:**
  1. In the display filter bar at the top of Wireshark, I typed `dns` and pressed Enter.
  2. Wireshark then displayed only the DNS traffic, which I could analyze in detail.

Wireshark capture of DNS traffic. The packet list shows several DNS queries and responses. Packet 120 is selected, showing details for Ethernet II, IP, and DNS. The DNS section is expanded, showing a Standard query for clients4.google.com.

No.	Time	Source	Destination	Protocol	Length	Info
120	4.906707	192.168.1.176	192.168.1.1	DNS	79	Standard query 0x8c99 A clients4.google.com
121	4.906813	192.168.1.176	192.168.1.1	DNS	79	Standard query 0xd2e7 HTTPS clients4.google.com
122	4.934572	192.168.1.1	192.168.1.176	DNS	153	Standard query response 0xd2e7 HTTPS clients4.google.com C...
123	4.938094	192.168.1.1	192.168.1.176	DNS	119	Standard query response 0x8c99 A clients4.google.com CNAME...
570	39.966317	192.168.1.176	192.168.1.1	DNS	81	Standard query 0xcfe7 A socks-mt1.pusher.com
571	39.966437	192.168.1.176	192.168.1.1	DNS	81	Standard query 0x5c0a HTTPS socks-mt1.pusher.com
573	39.993527	192.168.1.1	192.168.1.176	DNS	211	Standard query response 0xcfe7 A socks-mt1.pusher.com CNA...
574	40.004686	192.168.1.1	192.168.1.176	DNS	247	Standard query response 0x5c0a HTTPS socks-mt1.pusher.com...
655	45.462464	192.168.1.176	192.168.1.1	DNS	76	Standard query 0xf155 A www.linkedin.com
656	45.462598	192.168.1.176	192.168.1.1	DNS	76	Standard query 0xd6e HTTPS www.linkedin.com
665	45.488897	192.168.1.1	192.168.1.176	DNS	202	Standard query response 0xd6e HTTPS www.linkedin.com CNAM...

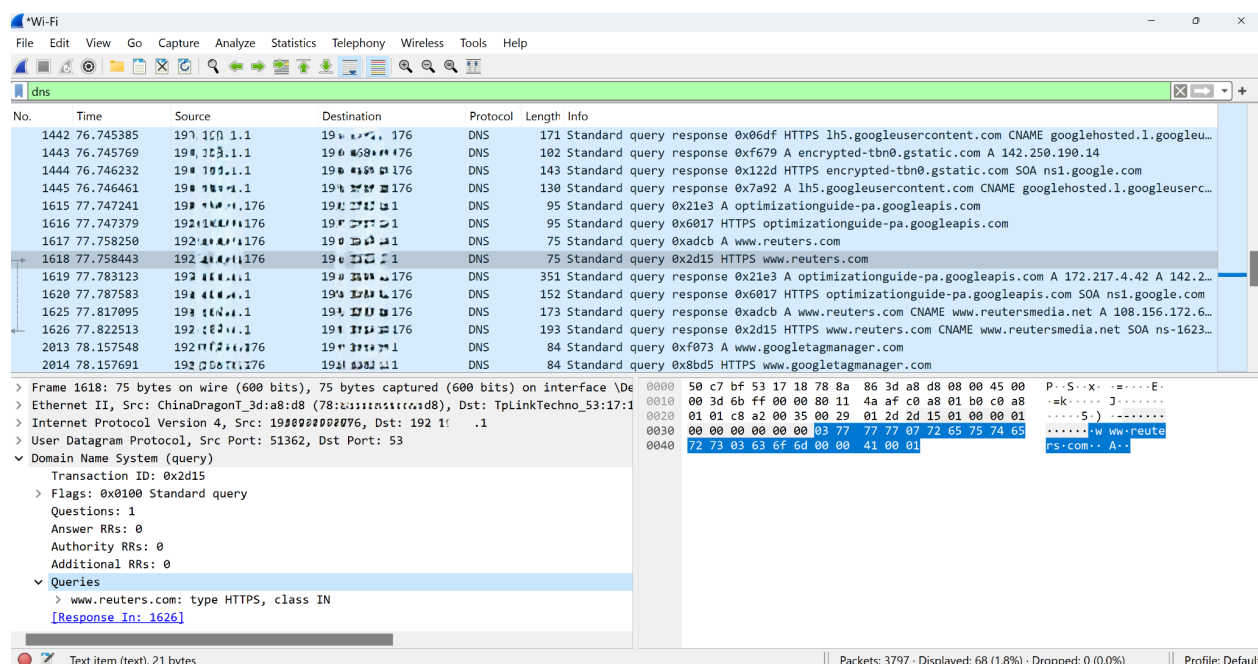
Frame 120: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface 0  
 Ethernet II, Src: ChinaDragonT\_3d:a8:d8 (78:8a:86:3d:a8:d8), Dst: TPLink (86:3d:a8:d8:00:00:00:00:00:00)  
 Internet Protocol Version 4, Src: 192.168.1.176, Dst: 192.168.1.1  
 User Datagram Protocol, Src Port: 63835, Dst Port: 53  
 Domain Name System (query)

Domain Name System: Protocol | Packets: 3797 · Displayed: 68 (1.8%) · Dropped: 0 (0.0%) | Profile: Default

## 4.2 Examined DNS Queries

I analyzed the DNS queries, which are requests my computer sent to the DNS server to resolve domain names into IP addresses. Understanding these queries is key to analyzing DNS traffic.

- **What I Did:**
  1. I looked through the filtered traffic for packets labeled "Standard query" in the "Info" column.
  2. I clicked on a "Standard query" packet to view its details in the lower pane.
  3. In the packet details pane, I expanded the "Domain Name System (query)" section to see the domain name that was queried (reuters.com)



**Importance:** Examining DNS queries is crucial because it allows me to understand which domain names my system is attempting to resolve. Every time I visit a website or access an online service, my computer needs to translate the human-readable domain name (www.reuters.com) into an IP address that it can use to route the request. By analyzing these queries, I can see exactly which domains my system is trying to resolve, which is fundamental for understanding the behavior of network traffic. This step also helps in identifying any unusual or unexpected DNS queries, which could be a sign of malicious activity or misconfiguration.

### 4.3 Analyzed DNS Responses

Next, I examined the DNS responses sent by the DNS server, which contained the IP address corresponding to the queried domain name. This helped me verify the DNS resolution process.

- **What I Did:**

1. I found the "Standard query response" packet that corresponded to the query I previously examined.
2. I clicked on the response packet to see its details in the lower pane, in this case, you can see in the screenshot that it says "Response In: 1626, which means the response is in frame 1626 in the packet list or I can click on that to open it.
3. I expanded the "Domain Name System (response)" section to see the "Answers" section, which contained the resolved IP address.

No.	Time	Source	Destination	Protocol	Length	Info
1616	77.747379	192.168.1.176	192.168.1.1	DNS	95	Standard query 0x6017 HTTPS optimizationguide-pa.googleapis.com
1617	77.758250	192.168.1.176	192.168.1.1	DNS	75	Standard query 0xadcb A www.reuters.com
1618	77.758443	192.168.1.176	192.168.1.1	DNS	75	Standard query 0x2d15 HTTPS www.reuters.com
1619	77.783123	192.168.1.176	192.168.1.176	DNS	351	Standard query response 0x21e3 A optimizationguide-pa.googleapis.com A 172.217.4.42 A 142.2...
1620	77.787583	192.168.1.176	192.168.1.176	DNS	152	Standard query response 0x6017 HTTPS optimizationguide-pa.googleapis.com SOA ns1.google.com
1625	77.817095	192.168.1.176	192.168.1.176	DNS	173	Standard query response 0xadcb A www.reuters.com CNAME www.reutersmedia.net A 108.156.172.6...
1626	77.822513	192.168.1.176	192.168.1.176	DNS	193	Standard query response 0x2d15 HTTPS www.reuters.com CNAME www.reutersmedia.net SOA ns-1623...
2013	78.157548	192.168.1.176	192.168.1.1	DNS	84	Standard query 0xf073 A www.googletagmanager.com
2014	78.157691	192.168.1.176	192.168.1.1	DNS	84	Standard query 0x8bd5 HTTPS www.googletagmanager.com
2041	78.190892	192.168.1.176	192.168.1.176	DNS	141	Standard query response 0x8bd5 HTTPS www.googletagmanager.com SOA ns1.google.com
2066	78.194439	192.168.1.176	192.168.1.176	DNS	100	Standard query response 0xf073 A www.googletagmanager.com A 142.250.191.104
2903	81.107798	192.168.1.176	192.168.1.1	DNS	105	Standard query 0xf7e7 A cloudfront-us-east-2.images.arcpublishing.com
2904	81.107901	192.168.1.176	192.168.1.1	DNS	105	Standard query 0x0210 HTTPS cloudfront-us-east-2.images.arcpublishing.com
2905	81.162647	192.168.1.176	192.168.1.176	DNS	183	Standard query response 0x0210 HTTPS cloudfront-us-east-2.images.arcpublishing.com SOA ns-1...

Domain Name System (response)

Transaction ID: 0x2d15

Flags: 0x8180 Standard query response, No error

Questions: 1

Answer RRs: 1

Authority RRs: 1

Additional RRs: 0

Queries

- www.reuters.com: type HTTPS, class IN

Answers

- www.reuters.com: type CNAME, class IN, cname www.reutersmedia.net

Authoritative nameservers

[Request In: 1618]

[Time: 0.064070000 seconds]

0000 78 8a 86 3d a8 d8 50 c7 bf 53 17 18 00 00 45 00 ... P . S . . . E .

0010 00 b3 9a e6 40 00 37 11 24 52 c0 a8 01 01 c0 a8 ... @ 7 \$ R . . . .

0020 01 b0 00 35 c8 a2 00 9f 23 90 2d 15 81 80 00 01 ... 5 . . . # . . . .

0030 00 01 00 01 00 00 03 77 77 07 72 65 75 74 65 ... . . . . w w . r e u t e

0040 72 73 03 63 6f 6d 00 00 41 00 01 c0 0c 00 05 00 ... s : c o m . A . . . . .

0050 01 00 00 01 07 00 16 03 77 77 77 0c 72 65 75 74 ... . . . . w w w . r e u t

0060 65 72 73 6d 65 64 69 61 03 6e 65 74 00 c0 31 00 ... e r s m e d i a . n e t . 1

0070 06 00 01 00 00 00 1e 00 48 07 6e 73 2d 31 36 32 ... . . . . . H . n s - 1 6 2

0080 33 09 61 77 73 64 6e 73 2d 31 30 02 63 6f 02 75 ... 3 . a w s d n s - 1 0 . c o u

0090 6b 00 11 61 77 73 64 6e 73 2d 68 6f 73 74 6d 61 ... k . a w s d n s - h o s t m a

00a0 73 74 65 72 06 61 6d 61 7a 6f 6e c0 18 00 00 00 ... s t e r . a m a z o n . . . .

00b0 01 00 00 1c 20 00 00 03 84 00 12 75 00 00 01 51 ... . . . . . u . . . . . Q

00c0 80

**Importance:** Analyzing DNS responses is just as important as examining the queries because it shows how the DNS server responds to my system’s requests. This step verifies whether the DNS server successfully resolved the domain name to the correct IP address. If the response contains incorrect or unexpected IP addresses, it could indicate DNS spoofing or other forms of attacks, where a malicious actor redirects traffic to a fraudulent server. Additionally, by comparing the DNS queries with their corresponding responses, I can ensure that the DNS resolution process is functioning correctly and efficiently, which is critical for maintaining network security and performance.

## Conclusion

Through this project, I have demonstrated a clear understanding of how DNS traffic operates within a network. I effectively utilized Wireshark to capture and analyze DNS queries and responses, gaining insights into the communication between my system and DNS servers. This process not only validated the correct functioning of the DNS resolution process but also highlighted the importance of monitoring DNS traffic for potential security threats.

By completing this analysis, I have reinforced my skills in using Wireshark as a powerful tool for network traffic analysis, specifically in the context of DNS operations. This knowledge is vital for identifying and addressing network issues, ensuring secure and reliable communication across the internet.