Network Traffic Analysis with Wireshark

By: Michael Emil Santos

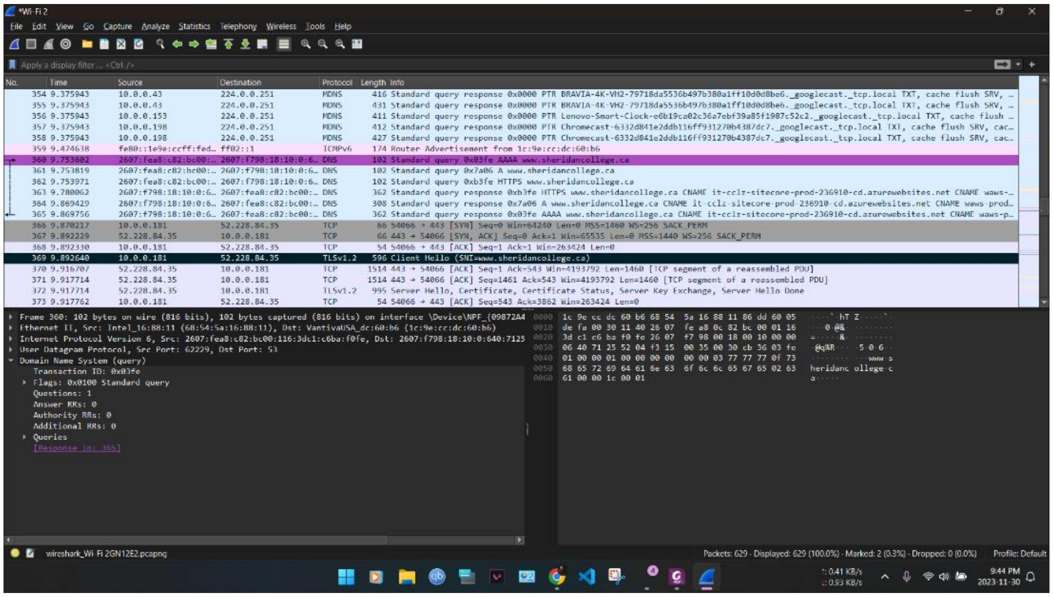
Introduction

This project demonstrates the use of Wireshark, an industry-standard network protocol analyzer, to capture and interpret network traffic. By analyzing data packets generated from common web requests, I have identified communication protocols and their role in network interactions. The objective was to distinguish normal traffic from potential anomalies, showcasing skills in network analysis and cybersecurity.

Project Scope and Objectives

1. Packet Capture and Analysis:
   Objective: To capture packets generated from accessing different websites and online applications.

   Approach: Using Wireshark, I recorded packet flows, focusing on key details such as IP addresses, protocol types (e.g., DNS, TCP, HTTPS), and packet structures. This data was essential for understanding the traffic patterns and network behavior.



*Wi-Fi 2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

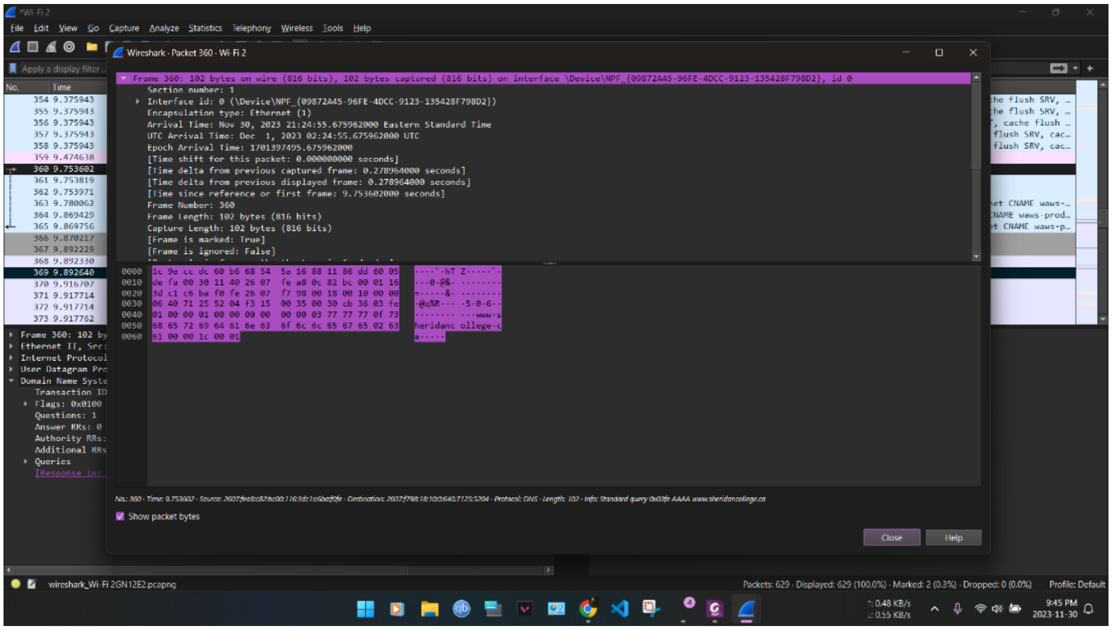
```
No.  Time      Source                Destination           Protocol Length Info
 354 9.375943  10.0.0.43             224.0.0.251           MDNS      416 Standard query response 0x0000 PTR BRAVIA-4K-VH2-79718da5536b497b380a1ff10d0d8be6._googlecast._tcp.local TXT, cache flush SRV, …
 355 9.375943  10.0.0.43             224.0.0.251           MDNS      431 Standard query response 0x0000 PTR BRAVIA-4K-VH2-79718da5536b497b380a1ff10d0d8be6._googlecast._tcp.local TXT, cache flush SRV, …
 356 9.375943  10.0.0.153            224.0.0.251           MDNS      411 Standard query response 0x0000 PTR Lenovo-Smart-Clock-e6b19ca02c36a7ebf39a85f1987c52c2._googlecast._tcp.local TXT, cache flush …
 357 9.375943  10.0.0.198            224.0.0.251           MDNS      412 Standard query response 0x0000 PTR Chromecast-6332d841e2ddb116ff931270b4387dc7._googlecast._tcp.local TXT, cache flush SRV, cac…
 358 9.375943  10.0.0.198            224.0.0.251           MDNS      427 Standard query response 0x0000 PTR Chromecast-6332d841e2ddb116ff931270b4387dc7._googlecast._tcp.local TXT, cache flush SRV, cac…
 359 9.474638  fe80::1e9e:ccff:fed…  ff02::1               ICMPv6    174 Router Advertisement from 1c:9e:cc:dc:60:b6
 360 9.753602  2607:fea8:c82:bc00:…  2607:f798:18:10:0:6…  DNS       102 Standard query 0x03fe AAAA www.sheridancollege.ca
 361 9.753819  2607:fea8:c82:bc00:…  2607:f798:18:10:0:6…  DNS       102 Standard query 0x7a06 A www.sheridancollege.ca
 362 9.753971  2607:fea8:c82:bc00:…  2607:f798:18:10:0:6…  DNS       102 Standard query 0xb3fe HTTPS www.sheridancollege.ca
 363 9.780062  2607:f798:18:10:0:6…  2607:fea8:c82:bc00:…  DNS       362 Standard query response 0xb3fe HTTPS www.sheridancollege.ca CNAME it-cclz-sitecore-prod-236910-cd.azurewebsites.net CNAME waws-…
 364 9.869429  2607:f798:18:10:0:6…  2607:fea8:c82:bc00:…  DNS       308 Standard query response 0x7a06 A www.sheridancollege.ca CNAME it-cclz-sitecore-prod-236910-cd.azurewebsites.net CNAME waws-prod…
 365 9.869756  2607:f798:18:10:0:6…  2607:fea8:c82:bc00:…  DNS       362 Standard query response 0x03fe AAAA www.sheridancollege.ca CNAME it-cclz-sitecore-prod-236910-cd.azurewebsites.net CNAME waws-p…
 366 9.870217  10.0.0.181            52.228.84.35          TCP        66 54066 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
 367 9.892229  52.228.84.35          10.0.0.181            TCP        66 443 → 54066 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
 368 9.892330  10.0.0.181            52.228.84.35          TCP        54 54066 → 443 [ACK] Seq=1 Ack=1 Win=263424 Len=0
 369 9.892640  10.0.0.181            52.228.84.35          TLSv1.2   596 Client Hello (SNI=www.sheridancollege.ca)
 370 9.916707  52.228.84.35          10.0.0.181            TCP      1514 443 → 54066 [ACK] Seq=1 Ack=543 Win=4193792 Len=1460 [TCP segment of a reassembled PDU]
 371 9.917714  52.228.84.35          10.0.0.181            TCP      1514 443 → 54066 [ACK] Seq=1461 Ack=543 Win=4193792 Len=1460 [TCP segment of a reassembled PDU]
 372 9.917714  52.228.84.35          10.0.0.181            TLSv1.2   995 Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
 373 9.917762  10.0.0.181            52.228.84.35          TCP        54 54066 → 443 [ACK] Seq=543 Ack=3862 Win=263424 Len=0
```

```
> Frame 360: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface \Device\NPF_{09872A4
> Ethernet II, Src: Intel_16:88:11 (68:54:5a:16:88:11), Dst: VantivaUSA_dc:60:b6 (1c:9e:cc:dc:60:b6)
> Internet Protocol Version 6, Src: 2607:fea8:c82:bc00:116:3dc1:c6ba:f0fe, Dst: 2607:f798:18:10:0:640:7125
> User Datagram Protocol, Src Port: 62229, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x03fe
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  > Queries
    [Response In: 365]
```

```
0000  1c 9e cc dc 60 b6 68 54  5a 16 88 11 86 dd 60 05   ····`·hT Z·····`·
0010  de fa 00 30 11 40 26 07  fe a8 0c 82 bc 00 01 16   ···0·@&· ········
0020  3d c1 c6 ba f0 fe 26 07  f7 98 00 18 00 10 00 00   =·····&· ········
0030  06 40 71 25 52 04 f3 15  00 35 00 30 cb 36 03 fe   ·@q%R··· ·5·0·6··
0040  01 00 00 01 00 00 00 00  00 00 03 77 77 77 0f 73   ········ ···www·s
0050  68 65 72 69 64 61 6e 63  6f 6c 6c 65 67 65 02 63   heridanc ollege·c
0060  61 00 00 1c 00 01                                  a·····
```

wireshark_Wi-Fi 2GN12E2.pcapng

Packets: 629 · Displayed: 629 (100.0%) · Marked: 2 (0.3%) · Dropped: 0 (0.0%)   Profile: Default



```
*Wi-Fi 2
Wireshark · Packet 360 · Wi-Fi 2

No.  Time
354 9.375943
355 9.375943
356 9.375943
357 9.375943
358 9.375943
359 9.474638
360 9.753602
361 9.753819
362 9.753971
363 9.780062
364 9.869429
365 9.869756
366 9.870217
367 9.892229
368 9.892330
369 9.892640
370 9.916707
371 9.917714
372 9.917714
373 9.917762

▾ Frame 360: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface \Device\NPF_{09872A45-96FE-4DCC-9123-135428F798D2}, id 0
    Section number: 1
  ▸ Interface id: 0 (\Device\NPF_{09872A45-96FE-4DCC-9123-135428F798D2})
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 30, 2023 21:24:55.675962000 Eastern Standard Time
    UTC Arrival Time: Dec  1, 2023 02:24:55.675962000 UTC
    Epoch Arrival Time: 1701397495.675962000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.278964000 seconds]
    [Time delta from previous displayed frame: 0.278964000 seconds]
    [Time since reference or first frame: 9.753602000 seconds]
    Frame Number: 360
    Frame Length: 102 bytes (816 bits)
    Capture Length: 102 bytes (816 bits)
    [Frame is marked: True]
    [Frame is ignored: False]

0000  1c 9e cc dc 60 b6 68 54  5a 16 88 11 86 dd 60 05   ····`·hT Z·····`·
0010  de fa 00 30 11 40 26 07  fe a8 0c 82 bc 00 01 16   ···0·@&· ········
0020  3d c1 c6 ba f0 fe 26 07  f7 98 00 18 00 10 00 00   =·····&· ········
0030  06 40 71 25 52 04 f3 15  00 35 00 30 cb 36 03 fe   ·@q%R··· ·5·0·6··
0040  01 00 00 01 00 00 00 00  00 00 03 77 77 77 0f 73   ········ ···www·s
0050  68 65 72 69 64 61 6e 63  6f 6c 6c 65 67 65 02 63   heridanc ollege·c
0060  61 00 00 1c 00 01                                  a·····

▸ Frame 360: 102 by
▸ Ethernet II, Src:
▸ Internet Protocol
▸ User Datagram Pro
▾ Domain Name Syste
    Transaction ID
  ▸ Flags: 0x0100
    Questions: 1
    Answer RRs: 0
    Authority RRs:
    Additional RRs
  ▸ Queries
    [Response In:

No.: 360 · Time: 9.753602 · Source: 2607:fea8:c82:bc00:116:3dc1:c6ba:f0fe · Destination: 2607:f798:18:10:0:640:7125:5204 · Protocol: DNS · Length: 102 · Info: Standard query 0x03fe AAAA www.sheridancollege.ca
☑ Show packet bytes

wireshark_Wi-Fi 2GN12E2.pcapng
Packets: 629 · Displayed: 629 (100.0%) · Marked: 2 (0.3%) · Dropped: 0 (0.0%)    Profile: Default
```

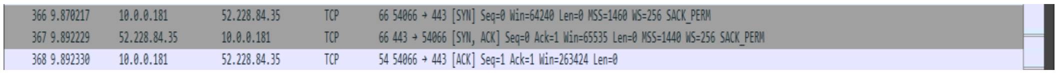
The first packet of information when a web request was sent can be viewed and interpreted as follows:

In this session 629 packets were captured and the packet number for the request is 360.

- Time of capture: 9.753602 seconds (from when the capture starts in Wireshark.
- Source IP Address: 2607:fea8:c82:bc00:116:3dc1:c6ba:f0fe
- Destination IP Address: 2607:f798:18:10:0:640:7125:5204
- Protocol used: DNS.
- Packet Length: 102 bytes.
- Query: AAAA (IPv6).
- Transaction ID: 0x03fe
- QR field is set to 0 that shows it is a query and not a response.

This packet represents a request to resolve the IPv6 address for the given domain. This request is specifically asking for the IPv6 address of the domain that the source is trying to establish a connection with the domain.

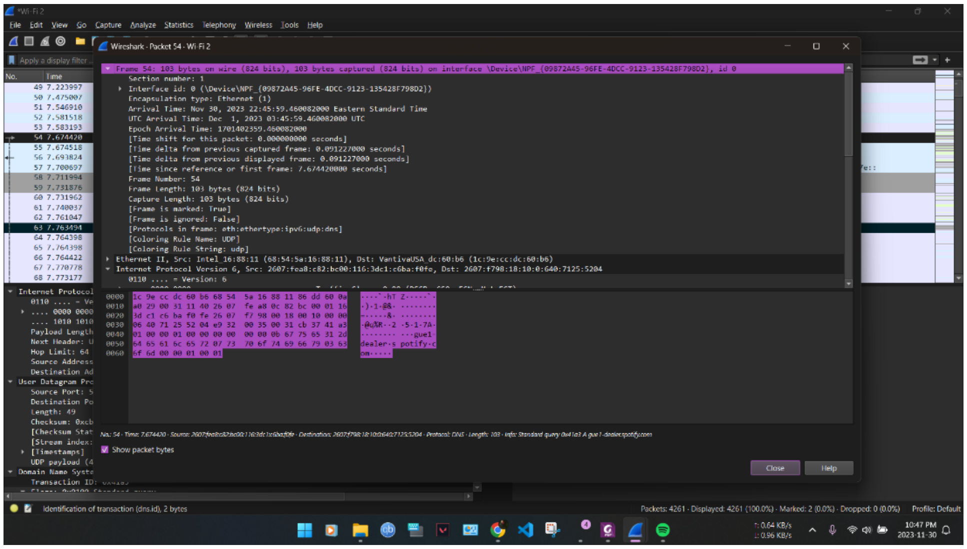
There are 3 packets that are not related to the request:



```
366 9.870217   10.0.0.181    52.228.84.35   TCP   66 54066 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
367 9.892229   52.228.84.35  10.0.0.181     TCP   66 443 → 54066 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
368 9.892330   10.0.0.181    52.228.84.35   TCP   54 54066 → 443 [ACK] Seq=1 Ack=1 Win=263424 Len=0
```

These 3 packets are part of the 3-way TCP handshake where the first packet is the initial step in establishing a TCP connection, the second packet is the response to the first packet which acknowledges the receipt of the initial packet, and the third packet indicates that both ends are synchronized and ready to exchange data over the established TCP connection.

Request to app (Spotify):



Wireshark · Packet 54 · Wi-Fi 2

```
▾ Frame 54: 103 bytes on wire (824 bits), 103 bytes captured (824 bits) on interface \Device\NPF_{09872A45-96FE-4DCC-9123-135428F798D2}, id 0
    Section number: 1
  ▸ Interface id: 0 (\Device\NPF_{09872A45-96FE-4DCC-9123-135428F798D2})
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 30, 2023 22:45:59.460082000 Eastern Standard Time
    UTC Arrival Time: Dec  1, 2023 03:45:59.460082000 UTC
    Epoch Arrival Time: 1701402359.460082000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.091227000 seconds]
    [Time delta from previous displayed frame: 0.091227000 seconds]
    [Time since reference or first frame: 7.674420000 seconds]
    Frame Number: 54
    Frame Length: 103 bytes (824 bits)
    Capture Length: 103 bytes (824 bits)
    [Frame is marked: True]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ipv6:udp:dns]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
▸ Ethernet II, Src: Intel_16:88:11 (68:54:5a:16:88:11), Dst: VantivaUSA_dc:60:b6 (1c:9e:cc:dc:60:b6)
▾ Internet Protocol Version 6, Src: 2607:fea8:c82:bc00:116:3dc1:c6ba:f0fe, Dst: 2607:f798:18:10:0:640:7125:5204
    0110 .... = Version: 6
```

```
0000  1c 9e cc dc 60 b6 68 54  5a 16 88 11 86 dd 60 0a   ····`·hT Z·····`·
0010  a0 29 00 31 11 40 26 07  fe a8 0c 82 bc 00 01 16   ·)·1·@&· ········
0020  3d c1 c6 ba f0 fe 26 07  f7 98 00 18 00 10 00 00   =·····&· ········
0030  06 40 71 25 52 04 e9 32  00 35 00 31 cb 37 41 a3   ·@q%R··2 ·5·1·7A·
0040  01 00 00 01 00 00 00 00  00 00 0b 67 75 65 31 2d   ········ ···gue1-
0050  64 65 61 6c 65 72 07 73  70 6f 74 69 66 79 03 63   dealer·s potify·c
0060  6f 6d 00 00 01 00 01                               om·····
```

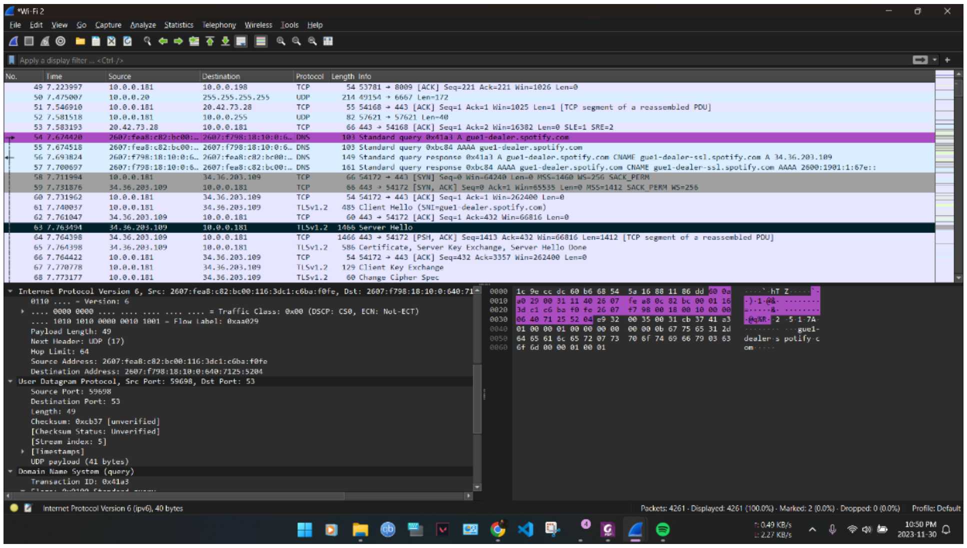
No.: 54 · Time: 7.674420 · Source: 2607:fea8:c82:bc00:116:3dc1:c6ba:f0fe · Destination: 2607:f798:18:10:0:640:7125:5204 · Protocol: DNS · Length: 103 · Info: Standard query 0x41a3 A gue1-dealer.spotify.com

☑ Show packet bytes

Identification of transaction (dns.id), 2 bytes

Packets: 4261 · Displayed: 4261 (100.0%) · Marked: 2 (0.0%) · Dropped: 0 (0.0%)   Profile: Default

Close
Help
0.64 KB/s
0.96 KB/s
10:47 PM
2023-11-30



*Wi-Fi 2

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time      Source               Destination          Protocol Length Info
  49 7.223997  10.0.0.181           10.0.0.198           TCP        54 53781 → 8009 [ACK] Seq=221 Ack=221 Win=1026 Len=0
  50 7.475007  10.0.0.20            255.255.255.255      UDP       214 49154 → 6667 Len=172
  51 7.546910  10.0.0.181           20.42.73.28          TCP        55 54168 → 443 [ACK] Seq=1 Ack=1 Win=1025 Len=1 [TCP segment of a reassembled PDU]
  52 7.581518  10.0.0.181           10.0.0.255           UDP        82 57621 → 57621 Len=40
  53 7.583193  20.42.73.28          10.0.0.181           TCP        66 443 → 54168 [ACK] Seq=1 Ack=2 Win=16382 Len=0 SLE=1 SRE=2
  54 7.674420  2607:fea8:c82:bc00:… 2607:f798:18:10:0:6… DNS       103 Standard query 0x41a3 A gue1-dealer.spotify.com
  55 7.674518  2607:fea8:c82:bc00:… 2607:f798:18:10:0:6… DNS       103 Standard query 0xbc84 AAAA gue1-dealer.spotify.com
  56 7.693824  2607:f798:18:10:0:6… 2607:fea8:c82:bc00:… DNS       149 Standard query response 0x41a3 A gue1-dealer.spotify.com CNAME gue1-dealer-ssl.spotify.com A 34.36.203.109
  57 7.700697  2607:f798:18:10:0:6… 2607:fea8:c82:bc00:… DNS       161 Standard query response 0xbc84 AAAA gue1-dealer.spotify.com CNAME gue1-dealer-ssl.spotify.com AAAA 2600:1901:1:67e::
  58 7.711994  10.0.0.181           34.36.203.109        TCP        66 54172 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
  59 7.731876  34.36.203.109        10.0.0.181           TCP        66 443 → 54172 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
  60 7.731962  10.0.0.181           34.36.203.109        TCP        54 54172 → 443 [ACK] Seq=1 Ack=1 Win=262400 Len=0
  61 7.740037  10.0.0.181           34.36.203.109        TLSv1.2   485 Client Hello (SNI=gue1-dealer.spotify.com)
  62 7.761047  34.36.203.109        10.0.0.181           TCP        60 443 → 54172 [ACK] Seq=1 Ack=432 Win=66816 Len=0
  63 7.763494  34.36.203.109        10.0.0.181           TLSv1.2  1466 Server Hello
  64 7.764398  34.36.203.109        10.0.0.181           TCP      1466 443 → 54172 [PSH, ACK] Seq=1413 Ack=432 Win=66816 Len=1412 [TCP segment of a reassembled PDU]
  65 7.764398  34.36.203.109        10.0.0.181           TLSv1.2   586 Certificate, Server Key Exchange, Server Hello Done
  66 7.764422  10.0.0.181           34.36.203.109        TCP        54 54172 → 443 [ACK] Seq=432 Ack=3357 Win=262400 Len=0
  67 7.770778  10.0.0.181           34.36.203.109        TLSv1.2   129 Client Key Exchange
  68 7.773177  10.0.0.181           34.36.203.109        TLSv1.2    60 Change Cipher Spec
```

```
▾ Internet Protocol Version 6, Src: 2607:fea8:c82:bc00:116:3dc1:c6ba:f0fe, Dst: 2607:f798:18:10:0:640:71
    0110 .... = Version: 6
  ▸ .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 1010 1010 0000 0010 1001 = Flow Label: 0xaa029
    Payload Length: 49
    Next Header: UDP (17)
    Hop Limit: 64
    Source Address: 2607:fea8:c82:bc00:116:3dc1:c6ba:f0fe
    Destination Address: 2607:f798:18:10:0:640:7125:5204
▾ User Datagram Protocol, Src Port: 59698, Dst Port: 53
    Source Port: 59698
    Destination Port: 53
    Length: 49
    Checksum: 0xcb37 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 5]
  ▸ [Timestamps]
    UDP payload (41 bytes)
▾ Domain Name System (query)
    Transaction ID: 0x41a3
```

```
0000  1c 9e cc dc 60 b6 68 54  5a 16 88 11 86 dd 60 0a   ····`·hT Z·····`·
0010  a0 29 00 31 11 40 26 07  fe a8 0c 82 bc 00 01 16   ·)·1·@&· ········
0020  3d c1 c6 ba f0 fe 26 07  f7 98 00 18 00 10 00 00   =·····&· ········
0030  06 40 71 25 52 04 e9 32  00 35 00 31 cb 37 41 a3   ·@q%R··2 ·5·1·7A·
0040  01 00 00 01 00 00 00 00  00 00 0b 67 75 65 31 2d   ········ ···gue1-
0050  64 65 61 6c 65 72 07 73  70 6f 74 69 66 79 03 63   dealer·s potify·c
0060  6f 6d 00 00 01 00 01                               om·····
```

Internet Protocol Version 6 (ipv6), 40 bytes

Packets: 4261 · Displayed: 4261 (100.0%) · Marked: 2 (0.0%) · Dropped: 0 (0.0%)   Profile: Default

The second packet of information when Spotify was opened can be viewed and interpreted as follows:

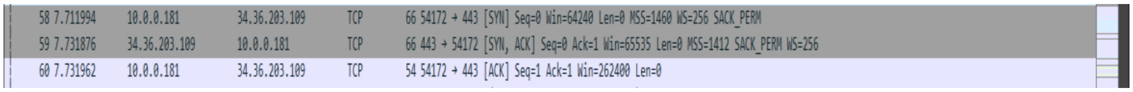
In this session 4261 packets were captured and the packet number for the request is 54.

- Time of capture: 7.674420 seconds (from when the capture starts in Wireshark.
- Source IP Address: 2607:fea8:c82:bc00:116:3dc1:c6ba:f0fe
- Destination IP Address: 2607:f798:18:10:0:640:7125:5204
- Protocol used: DNS.
- Packet Length: 103 bytes.
- Query: A (IPv4).
- Transaction ID: 0x41a3
- Domain: gue1-dealer.spotify.com

- QR field is set to 0 that shows it is a query and not a response.

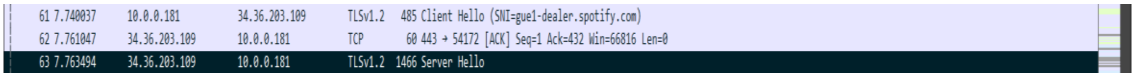
This packet represents a request to resolve the IPv4 address for the domain "gue1-dealer.spotify.com" through the DNS. The source is querying the DNS server for the IPv4 address associated with the domain name.

There are multiple packets that are not related to the request:



```
58 7.711994   10.0.0.181     34.36.203.109  TCP   66 54172 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
59 7.731876   34.36.203.109  10.0.0.181     TCP   66 443 → 54172 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
60 7.731962   10.0.0.181     34.36.203.109  TCP   54 54172 → 443 [ACK] Seq=1 Ack=1 Win=262400 Len=0
```

These 3 packets are part of the 3-way TCP Handshake.

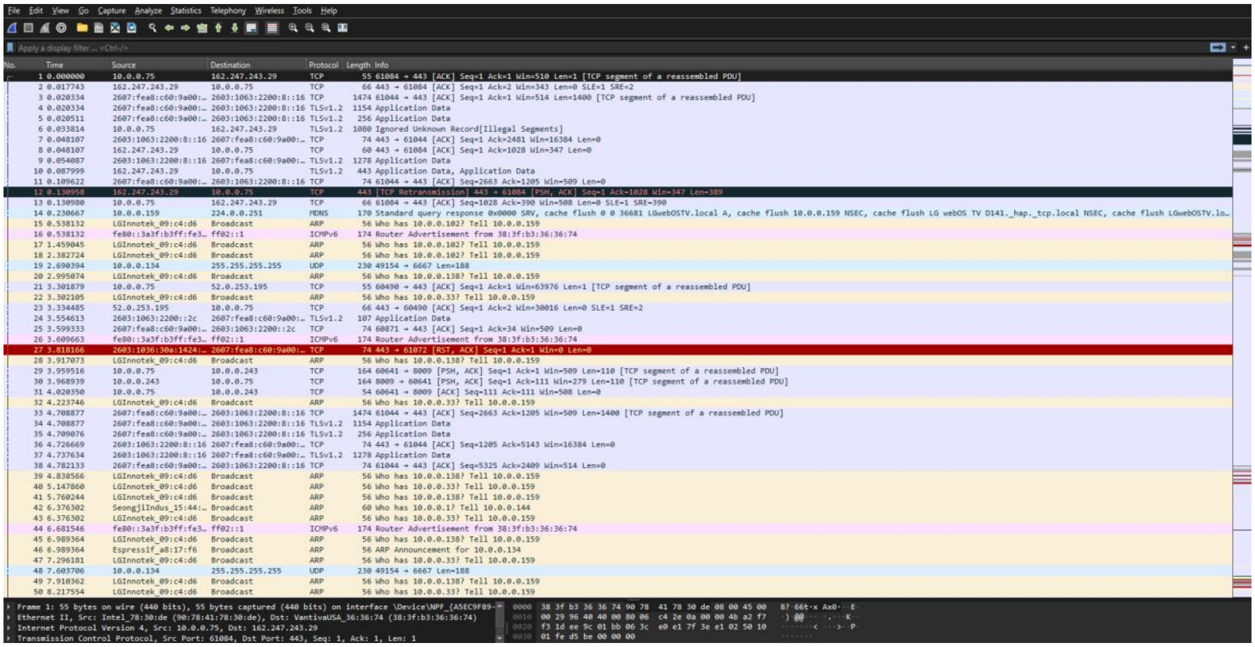


```
61 7.740037   10.0.0.181     34.36.203.109  TLSv1.2  485 Client Hello (SNI=gue1-dealer.spotify.com)
62 7.761047   34.36.203.109  10.0.0.181     TCP       60 443 → 54172 [ACK] Seq=1 Ack=432 Win=66816 Len=0
63 7.763494   34.36.203.109  10.0.0.181     TLSv1.2 1466 Server Hello
```

There are 2 TLS-related packets of which the first one, the "Client Hello" message is the initial message sent by the client to initiate the TLS handshake with the server "gue1-dealer.spotify.com".

The second packet "Server Hello" is the response from the server to the client's "Hello" message during the TLS handshake. It is used to acknowledge and secure communication.

The TLS Version is TLSv1.2.

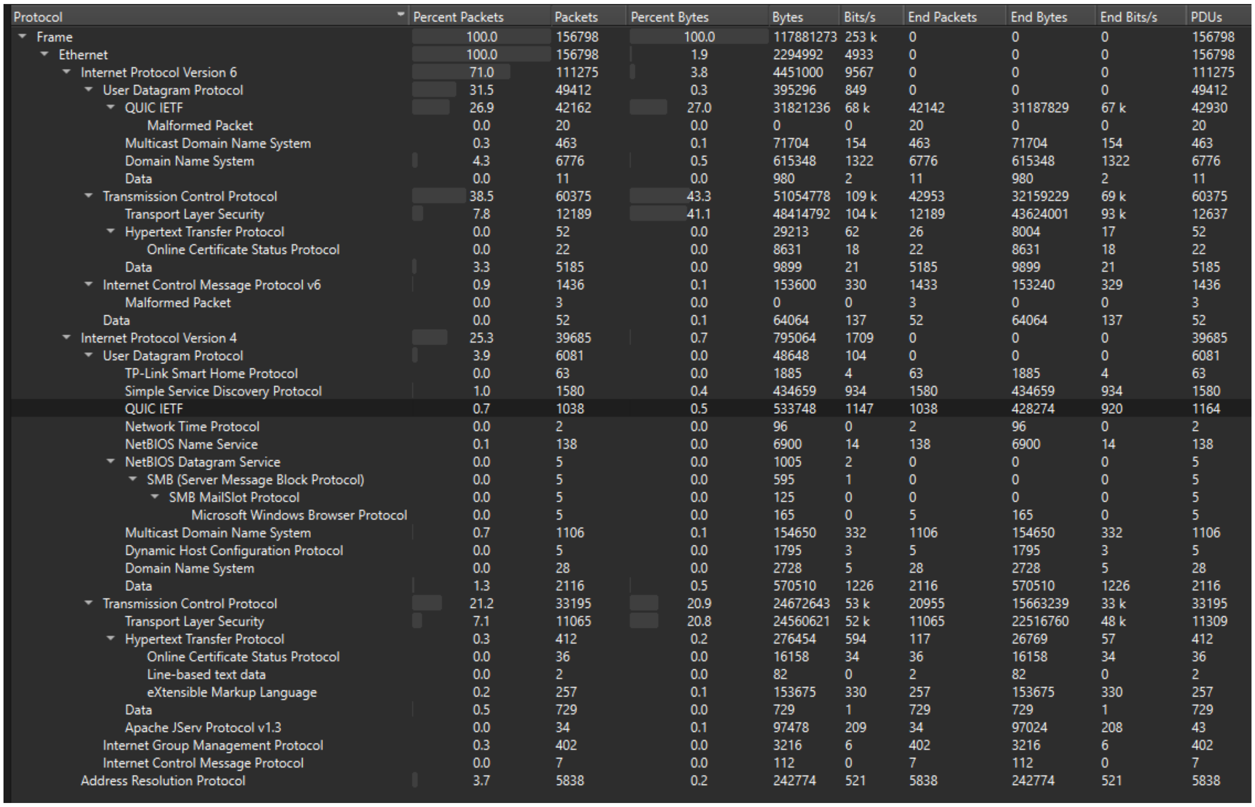


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No. Time       Source                Destination           Protocol Length Info
 1 0.000000   10.0.0.75             162.247.243.29        TCP        55 61084 → 443 [ACK] Seq=1 Ack=1 Win=510 Len=1 [TCP segment of a reassembled PDU]
 2 0.017743   162.247.243.29        10.0.0.75             TCP        66 443 → 61084 [ACK] Seq=1 Ack=2 Win=343 Len=0 SLE=1 SRE=2
 3 0.020334   2607:fea8:c60:9a00:…  2603:1063:2200:8::16  TCP      1474 61044 → 443 [ACK] Seq=1 Ack=1 Win=514 Len=1400 [TCP segment of a reassembled PDU]
 4 0.020334   2607:fea8:c60:9a00:…  2603:1063:2200:8::16  TLSv1.2  1154 Application Data
 5 0.020511   2607:fea8:c60:9a00:…  2603:1063:2200:8::16  TLSv1.2   256 Application Data
 6 0.033814   10.0.0.75             162.247.243.29        TLSv1.2  1080 Ignored Unknown Record[Illegal Segments]
 7 0.048107   2603:1063:2200:8::16  2607:fea8:c60:9a00:…  TCP        74 443 → 61044 [ACK] Seq=1 Ack=2481 Win=16384 Len=0
 8 0.048107   162.247.243.29        10.0.0.75             TCP        60 443 → 61084 [ACK] Seq=1 Ack=1028 Win=347 Len=0
 9 0.054087   2603:1063:2200:8::16  2607:fea8:c60:9a00:…  TLSv1.2  1278 Application Data
10 0.087999   162.247.243.29        10.0.0.75             TLSv1.2   443 Application Data, Application Data
11 0.109622   2607:fea8:c60:9a00:…  2603:1063:2200:8::16  TCP        74 61044 → 443 [ACK] Seq=2663 Ack=1205 Win=509 Len=0
12 0.130958   162.247.243.29        10.0.0.75             TCP       443 [TCP Retransmission] 443 → 61084 [PSH, ACK] Seq=1 Ack=1028 Win=347 Len=389
13 0.130980   10.0.0.75             162.247.243.29        TCP        66 61084 → 443 [ACK] Seq=1028 Ack=390 Win=508 Len=0 SLE=1 SRE=390
14 0.230667   10.0.0.159            224.0.0.251           MDNS      170 Standard query response 0x0000 SRV, cache flush 0 0 36681 LGwebOSTV.local A, cache flush 10.0.0.159 NSEC, cache flush LG webOS TV D141._hap._tcp.local NSEC, cache flush LGwebOSTV.lo…
15 0.538132   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.102? Tell 10.0.0.159
16 0.538132   fe80::3a3f:b3ff:fe3…  ff02::1               ICMPv6    174 Router Advertisement from 38:3f:b3:36:36:74
17 1.459045   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.102? Tell 10.0.0.159
18 2.382724   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.102? Tell 10.0.0.159
19 2.690394   10.0.0.134            255.255.255.255       UDP       230 49154 → 6667 Len=188
20 2.995074   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.138? Tell 10.0.0.159
21 3.301879   10.0.0.75             52.0.253.195          TCP        55 60490 → 443 [ACK] Seq=1 Ack=1 Win=63976 Len=1 [TCP segment of a reassembled PDU]
22 3.302105   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.33? Tell 10.0.0.159
23 3.334485   52.0.253.195          10.0.0.75             TCP        66 443 → 60490 [ACK] Seq=1 Ack=2 Win=30016 Len=0 SLE=1 SRE=2
24 3.554613   2603:1063:2200::2c    2607:fea8:c60:9a00:…  TLSv1.2   107 Application Data
25 3.599333   2607:fea8:c60:9a00:…  2603:1063:2200::2c    TCP        74 60871 → 443 [ACK] Seq=1 Ack=34 Win=509 Len=0
26 3.609663   fe80::3a3f:b3ff:fe3…  ff02::1               ICMPv6    174 Router Advertisement from 38:3f:b3:36:36:74
27 3.818166   2603:1036:30a:1424:…  2607:fea8:c60:9a00:…  TCP        74 443 → 61072 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
28 3.917073   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.138? Tell 10.0.0.159
29 3.959516   10.0.0.75             10.0.0.243            TCP       164 60641 → 8009 [PSH, ACK] Seq=1 Ack=1 Win=509 Len=110 [TCP segment of a reassembled PDU]
30 3.968939   10.0.0.243            10.0.0.75             TCP       164 8009 → 60641 [PSH, ACK] Seq=1 Ack=111 Win=279 Len=110 [TCP segment of a reassembled PDU]
31 4.020350   10.0.0.75             10.0.0.243            TCP        54 60641 → 8009 [ACK] Seq=111 Ack=111 Win=508 Len=0
32 4.223746   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.33? Tell 10.0.0.159
33 4.708877   2607:fea8:c60:9a00:…  2603:1063:2200:8::16  TCP      1474 61044 → 443 [ACK] Seq=2663 Ack=1205 Win=509 Len=1400 [TCP segment of a reassembled PDU]
34 4.708877   2607:fea8:c60:9a00:…  2603:1063:2200:8::16  TLSv1.2  1154 Application Data
35 4.709076   2607:fea8:c60:9a00:…  2603:1063:2200:8::16  TLSv1.2   256 Application Data
36 4.726669   2603:1063:2200:8::16  2607:fea8:c60:9a00:…  TCP        74 443 → 61044 [ACK] Seq=1205 Ack=5143 Win=16384 Len=0
37 4.737634   2603:1063:2200:8::16  2607:fea8:c60:9a00:…  TLSv1.2  1278 Application Data
38 4.782133   2607:fea8:c60:9a00:…  2603:1063:2200:8::16  TCP        74 61044 → 443 [ACK] Seq=5325 Ack=2409 Win=514 Len=0
39 4.838566   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.138? Tell 10.0.0.159
40 5.147860   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.33? Tell 10.0.0.159
41 5.760244   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.138? Tell 10.0.0.159
42 6.376302   SeongjiIndus_15:44:…  Broadcast             ARP        60 Who has 10.0.0.1? Tell 10.0.0.144
43 6.376302   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.33? Tell 10.0.0.159
44 6.681546   fe80::3a3f:b3ff:fe3…  ff02::1               ICMPv6    174 Router Advertisement from 38:3f:b3:36:36:74
45 6.989364   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.138? Tell 10.0.0.159
46 6.989364   Espressif_a8:17:f6    Broadcast             ARP        56 ARP Announcement for 10.0.0.134
47 7.296181   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.33? Tell 10.0.0.159
48 7.603706   10.0.0.134            255.255.255.255       UDP       230 49154 → 6667 Len=188
49 7.910362   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.138? Tell 10.0.0.159
50 8.217554   LGInnotek_09:c4:d6    Broadcast             ARP        56 Who has 10.0.0.33? Tell 10.0.0.159
```

```
Frame 1: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface \Device\NPF_{A5EC9F89-
Ethernet II, Src: Intel_78:30:de (90:78:41:78:30:de), Dst: VantivaUSA_36:36:74 (38:3f:b3:36:36:74)
Internet Protocol Version 4, Src: 10.0.0.75, Dst: 162.247.243.29
Transmission Control Protocol, Src Port: 61084, Dst Port: 443, Seq: 1, Ack: 1, Len: 1
```

```
0000  38 3f b3 36 36 74 90 78  41 78 30 de 08 00 45 00   8?·66t·x Ax0···E·
0010  00 29 96 40 40 00 80 06  c4 2e 0a 00 00 4b a2 f7   ·)·@@·····.···K··
0020  f3 1d ee 9c 01 bb 06 3c  e0 e1 7f 3e e1 02 50 10   ·······<···>··P·
0030  01 fe d5 be 00 00 00       ·······
```



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	156798	100.0	117881273	253 k	0	0	0	156798
Ethernet	100.0	156798	1.9	2294992	4933	0	0	0	156798
Internet Protocol Version 6	71.0	111275	3.8	4451000	9567	0	0	0	111275
User Datagram Protocol	31.5	49412	0.3	395296	849	0	0	0	49412
QUIC IETF	26.9	42162	27.0	31821236	68 k	42142	31187829	67 k	42930
Malformed Packet	0.0	20	0.0	0	0	20	0	0	20
Multicast Domain Name System	0.3	463	0.1	71704	154	463	71704	154	463
Domain Name System	4.3	6776	0.5	615348	1322	6776	615348	1322	6776
Data	0.0	11	0.0	980	2	11	980	2	11
Transmission Control Protocol	38.5	60375	43.3	51054778	109 k	42953	32159229	69 k	60375
Transport Layer Security	7.8	12189	41.1	48414792	104 k	12189	43624001	93 k	12637
Hypertext Transfer Protocol	0.0	52	0.0	29213	62	26	8004	17	52
Online Certificate Status Protocol	0.0	22	0.0	8631	18	22	8631	18	22
Data	3.3	5185	0.0	9899	21	5185	9899	21	5185
Internet Control Message Protocol v6	0.9	1436	0.1	153600	330	1433	153240	329	1436
Malformed Packet	0.0	3	0.0	0	0	3	0	0	3
Data	0.0	52	0.1	64064	137	52	64064	137	52
Internet Protocol Version 4	25.3	39685	0.7	795064	1709	0	0	0	39685
User Datagram Protocol	3.9	6081	0.0	48648	104	0	0	0	6081
TP-Link Smart Home Protocol	0.0	63	0.0	1885	4	63	1885	4	63
Simple Service Discovery Protocol	1.0	1580	0.4	434659	934	1580	434659	934	1580
QUIC IETF	0.7	1038	0.5	533748	1147	1038	428274	920	1164
Network Time Protocol	0.0	2	0.0	96	0	2	96	0	2
NetBIOS Name Service	0.1	138	0.0	6900	14	138	6900	14	138
NetBIOS Datagram Service	0.0	5	0.0	1005	2	0	0	0	5
SMB (Server Message Block Protocol)	0.0	5	0.0	595	1	0	0	0	5
SMB MailSlot Protocol	0.0	5	0.0	125	0	0	0	0	5
Microsoft Windows Browser Protocol	0.0	5	0.0	165	0	5	165	0	5
Multicast Domain Name System	0.7	1106	0.1	154650	332	1106	154650	332	1106
Dynamic Host Configuration Protocol	0.0	5	0.0	1795	3	5	1795	3	5
Domain Name System	0.0	28	0.0	2728	5	28	2728	5	28
Data	1.3	2116	0.5	570510	1226	2116	570510	1226	2116
Transmission Control Protocol	21.2	33195	20.9	24672643	53 k	20955	15663239	33 k	33195
Transport Layer Security	7.1	11065	20.8	24560621	52 k	11065	22516760	48 k	11309
Hypertext Transfer Protocol	0.3	412	0.2	276454	594	117	26769	57	412
Online Certificate Status Protocol	0.0	36	0.0	16158	34	36	16158	34	36
Line-based text data	0.0	2	0.0	82	0	2	82	0	2
eXtensible Markup Language	0.2	257	0.1	153675	330	257	153675	330	257
Data	0.5	729	0.0	729	1	729	729	1	729
Apache JServ Protocol v1.3	0.0	34	0.1	97478	209	34	97024	208	43
Internet Group Management Protocol	0.3	402	0.0	3216	6	402	3216	6	402
Internet Control Message Protocol	0.0	7	0.0	112	0	7	112	0	7
Address Resolution Protocol	3.7	5838	0.2	242774	521	5838	242774	521	5838

2. Protocols Identified and Analyzed - The following protocols were observed in the network capture:

- **TCP (Transmission Control Protocol)**: This is the main protocol used for most of the traffic, indicating reliable, connection-oriented transmissions. (Eddy, 2022)
- **UDP (User Datagram Protocol)**: This is less prevalent than TCP, used for connectionless communications, typically for streaming, gaming, and real-time services. (Postel, 1980)
- **HTTPS (Over TCP Port 443)**: This represents encrypted web traffic, indicating secure web browsing. (Touch & Eliot Lear, 2023)
- **DNS (Over UDP Port 53)**: Domain Name System are critical services and essential for resolving domain names to IP addresses. (Touch & Eliot Lear, 2023)
- **SSDP (Simple Service Discovery Protocol over UDP Port 1900)**: Used for discovering and locating UPnP devices on a local network. (Goland, 1999)
- **mDNS (Multicast DNS over UDP Port 5353)**: Used for resolving hostnames to IP addresses within small networks without central DNS services. (Cheshire, 2013)

A. Types of Services and Apps Using These Protocols

Based on the protocols identified, the following services and applications are typically associated:

- **Web Browsing (HTTPS/TCP Port 443)**: This includes all secure web traffic, including services such as online banking, shopping, social media platforms, streaming services, and cloud-based applications.

- **Streaming and Real-time Communications (UDP)**: Encompasses applications like video conferencing apps, VoIP services, live game streaming platforms, and other multimedia streaming services.
- **Domain Name Resolution (DNS/UDP Port 53)**: This is a foundational aspect of internet usage and includes any service that requires domain name resolution, which is a fundamental part of internet usage, including all above-mentioned services.
- **Network Device Discovery (SSDP/UDP Port 1900 and mDNS/UDP Port 5353)**: These services are typically used by network devices for configuration and discovery within a local area network.

B. Exact services and applications accessed based on the capture information are as follows:

- **IPv6 Addresses**:
  - 2a03:2880::/32 This range indicates the activities related to Facebook and its associated platforms.
  - 2607:f8b0::/32 This range points to usage of Google's suite, including Search, YouTube, Gmail, etc.
  - 2620:1ec::/32 This range Suggests access to Wikipedia or related resources.
- **IPv4 Addresses**:
  - 54.231.232.49, 52.109.92.58, and 3.98.112.127 are within the **Amazon Web Services (AWS)** range, potentially indicating use of AWS-hosted services or websites.
  - Addresses in the 23.*.*.* range are typically associated with **Akamai Technologies**, a content delivery network provider, which could indicate access to various content-rich services such as media streaming, software download sites, or large online platforms.

Based on this data, the report concludes that the network traffic consisted of a variety of encrypted web browsing activities, including social media interaction, information searching, video streaming, and the use of cloud-based services. Additionally, local network activities included device discovery and configuration.

Also, the identification of Facebook services, Google services, and Akamai Technologies from the IP addresses in the network capture is based on known IP address allocations and the services these companies provide. The reason these addresses are associated with these services and what roles they play in terms of privacy, security, hosting, and other network services:

Facebook Services (IPv6 addresses in the 2a03:2880::/32 range)

Why It Shows Up:

- Facebook owns a block of IP addresses, and traffic to or from these addresses can be indicative of interactions with Facebook's various services, such as the main social networking site, Instagram, WhatsApp, or other affiliated platforms.

Purpose in Terms of Privacy and Security:

- Facebook uses encryption (like HTTPS over TCP port 443) to protect user data and communications.
- Implements privacy controls, allowing users to manage who sees their information.

Hosting and Network Services:

- Operates its own data centers and uses a content delivery network (CDN) to distribute content efficiently across the globe.
- Provides APIs and services for third-party developers and businesses. (Engineering at Meta: Networking & Traffic, n.d.)

Google Services (IPv6 addresses in the 2607:f8b0::/32 range)

Why It Shows Up:

- Google owns a vast range of IP addresses for its services, including search, email (Gmail), video hosting (YouTube), and cloud services (Google Cloud Platform).

Purpose in Terms of Privacy and Security:

- Google also employs encryption to protect data in transit.
- Offers various security features like two-factor authentication and secure browsing tools.
- It has robust privacy policies with options for users to control their data.

Hosting and Network Services:

- Provides extensive hosting services via Google Cloud, including computing, storage, and networking capabilities.
- Operates one of the largest CDNs, speeding up access to services and reducing latency. (Google Cloud documentation, n.d.)

Akamai Technologies (IPv4 addresses in the 23.*.*.* range)

Why It Shows Up:

- Akamai is one of the largest CDN providers and their IP ranges are used to deliver content for clients across different industries. This can include anything from media streaming to software downloads.

Purpose in Terms of Privacy and Security:

- Akamai's CDN enhances security by defending against DDoS attacks and improving website performance, which is crucial for maintaining availability and user experience.
- Offers secure content delivery with advanced encryption and authentication features.

Hosting and Network Services:

- Reduces latency by caching content on servers closer to the end-users.
- Provides cloud security solutions, web and mobile performance services, and cloud service monitoring. (Resource Library, n.d.)

The presence of these services in network traffic is common given the widespread use of these platforms for both personal and business purposes. They implement layers of security and privacy measures to protect user data and ensure the integrity of their services. In terms of network services, they contribute to a huge portion of internet traffic because they host a variety of services, from social media to search engines, video streaming, and beyond.

3. Network Troubleshooting with Wireshark

There are three ways administrators can use Wireshark to troubleshoot network problems.

A. Use packet captures to identify traffic types:

To monitor viruses, potentially spreading to other hosts or malfunctioning NICs and the kind of traffic flowing through the network. Network administrators want to be able to verify that traffic is genuine and valid if something seems off on the network**.**

Consider the two following instances of traffic identification:

- **Investigate NIC communications in a cluster:** In private cloud failover clusters or other deployments, Intel Advanced Network Services (ANS) probes detect network team members and ensure they are operational.

We could investigate communication between cluster team members or investigate why there are so many ANS probes, indicating a problem with one of the NICs. The NIC may be incorrectly configured, using the incorrect driver, or failing. Because Wireshark already has an ANS traffic filter, this capture is simple.

- **Detect virus propagation based on Address Resolution Protocol (ARP) queries or known malware sites:** A high number of ARP queries coming from a single machine is one suspicious traffic pattern. A virus may have infected the device and is now trying to spread to other systems. A protocol analyzer can assist us in detecting this type of traffic and its source, allowing us to thoroughly inspect the machine.

Perhaps we've heard of malware that spreads over the network by using recognizable URLs or other data. Use the documentation for Wireshark to help create a filter using these details to determine whether the malware was on the network.

B. Use packet capture to monitor the network performance:

Wireshark includes a statistical analysis tool to aid in the management of network performance issues. It provides numerous options for analyzing network efficiency and identifying areas for improvement.

In Wireshark, go to the Statistics menu and select the following performance filters:

- **Capture File Properties**: Provides basic details about the capture file, such as its size, the duration of its capture, its interface, its filters, etc. When archiving baseline captures for future comparisons, this information is extremely helpful.

- **Protocol Hierarchy:** Identifies and ranks every protocol in the capture, allowing us to determine the proportion of traffic that each protocol accounts for on the network.

For example, we can quickly find out how much of the traffic was on the network is IPv6 versus IPv4 or compare DNS and Dynamic Host Configuration Protocol traffic.

- **Conversations:** Shows all the communications, along with the start time, traffic type, duration, and size, between the designated endpoints. Apply what we've learned to clarify the data that is transferred between two nodes**.**
- **IO Graphs:** Allows us to filter content and view performance data in an interactive graph format that shows network traffic information. For instance, instead of using the numerical output from the Protocol Hierarchy tool, we might decide to use a graph to see the quantity of errors compared to legitimate traffic.
- **TCP Stream Graphs:** Shows details about TCP traffic, such as roundtrip, throughput, and window scaling. To enable this menu, select a packet that is TCP-based.
- **IPv4 Statistics > Destination and Ports:** Identifies the destination of network traffic, such as IP addresses and port numbers, and filters it accordingly. Remember that certain port numbers, like port 80 for HTTP and port 443 for HTTPS, are standardized**.**
- **IPv4 Statistics > Source and Destination Addresses:** Displays network traffic arranged according to IPv4 source and destination addresses, thereby reducing the amount of traffic that needs to be investigated. Understanding the relationship between two nodes—such as routers that share routing tables or proxies and the clients they serve—needs to have access to this data.

IPv6 network traffic can also be handled with Wireshark. Comprehensive documentation is available on the official Wireshark website.

C. Use packet captures to discover network devices:

Protocol analyzers can also be used to identify and record the devices connected to the network and exchange data. Network experts can classify involvement in networks in several ways, such as the following:

- Network capture makes it evident which hosts are connected to the network by displaying the source and destination MAC and IP addresses.
- In addition to the protocols identified, knowing what applications exchange network information is useful for ensuring that security system rules and packet filtering configurations are correct. We can also use these criteria to detect peer-to-peer BitTorrent or other network abuses.
- Client requests to the web, FTP, or file servers are displayed, assisting in the identification of queries or processes that consume server resources.
- Recognize information sent to client devices to figure out what the clients do. Ansible or other configuration management automation messages may be included in this data.

D. Network Security Insights

In addition, network analysts and cybersecurity experts can investigate security flaws throughout a network in several ways with Wireshark. They can utilize Wireshark, for instance, to:

- **Determine network threats:** Malware, viruses, and other malicious traffic can all be determined using Wireshark. Cybersecurity experts can identify suspicious activity patterns and take the necessary steps to reduce the threat by examining network traffic.

- **Track network performance:** We can use Wireshark to track network performance and find any bottlenecks or other problems that might be interfering with it. Cybersecurity experts can find the source of performance problems and take the necessary steps to fix them by examining network traffic.
- **Identify network intrusions:** By examining network traffic for indications of unauthorized access or other questionable activity, Wireshark can be used to identify network intrusions. Cybersecurity experts can identify and stop network intrusions before they cause serious harm by continuously monitoring network traffic.
- **Troubleshoot network issues:** By examining network traffic and determining the source of the problem, Wireshark can be used to troubleshoot network issues. Cybersecurity experts can examine network traffic in real-time or from a saved capture file to find configuration errors, network congestion, malfunctioning devices, and other problems that might be affecting network performance.

Conclusion:

This project highlights Wireshark's value as a powerful tool for network analysis, troubleshooting, and security monitoring. Through this hands-on experience, I developed skills in protocol analysis, traffic management, and security diagnostics. This project illustrates my ability to apply technical knowledge to real-world network scenarios, enhancing network security and operational efficiency.