Intern Name: S. Sai Amrutha
Mail: 240160226211.sunkavalli@gdgu.org

TASK – 5 CAPTURE AND ANALYSE NETWORK TRAFFIC USING WIRESHARK

Tools used: Wireshark, Google, ChatGPT.

INTRODUCTION:

Network traffic analysis is a crucial activity in cybersecurity that involves examining data packets transmitted between devices on a network. Understanding how these packets move, what data they carry, and which protocols they use enables security professionals to monitor for abnormal behavior, detect potential threats, and optimize network performance.

Wireshark is a widely used, open-source network protocol analyzer that captures and displays network traffic in real-time. It provides detailed insight into each packet, including its source and destination, the protocols involved, and the actual data being transmitted. By using Wireshark for packet capture and analysis, analysts can trace communication patterns, identify unauthorized access, and investigate security incidents with precision.

In this task, live network traffic is captured and analyzed using Wireshark to identify and study the protocols in use on a typical network. This process not only helps in understanding regular network operations but also strengthens skills required for investigating anomalies and securing digital infrastructure.

TCP/IP:

TCP/IP (Transmission Control Protocol/Internet Protocol) is the fundamental communication protocol suite that serves as the backbone of the internet and most modern networks. It is a layered model consisting of four layers:

- Application Layer: The topmost layer where applications like web browsers, email clients, and file transfer programs operate using protocols such as HTTP, FTP, SMTP, and DNS.

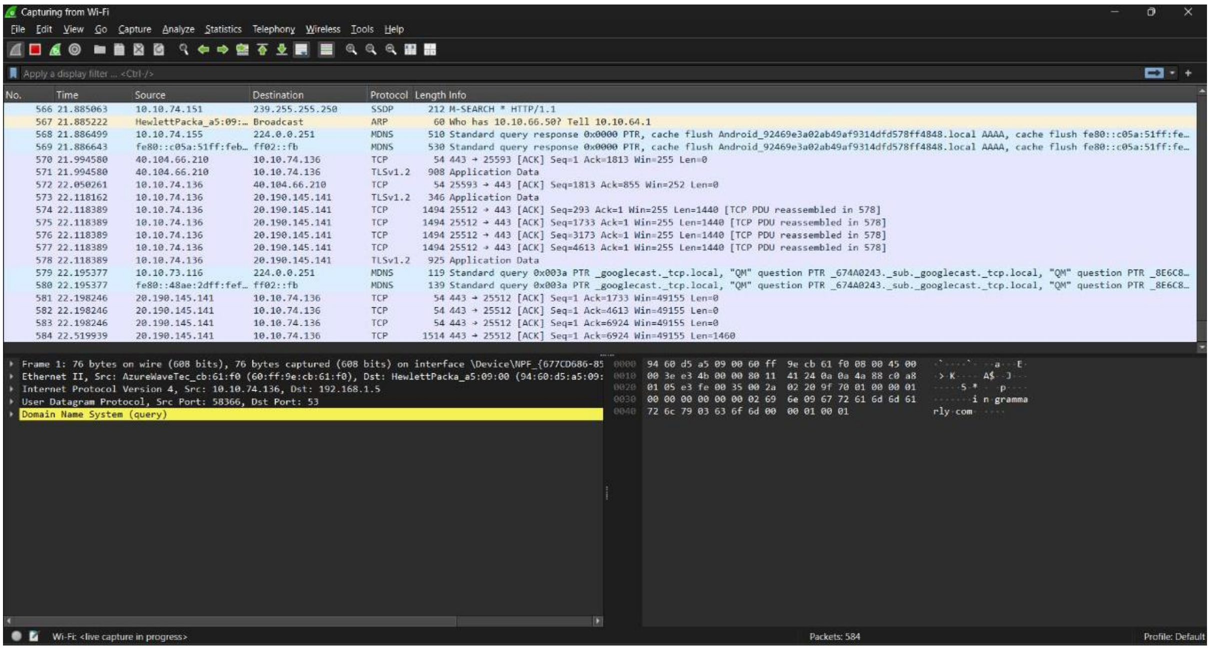
- Transport Layer: Responsible for reliable data transmission between devices. Key protocols here are TCP, which ensures reliable and ordered delivery, and UDP, used for faster but less reliable communication.

- Internet Layer: Handles logical addressing and routing of data packets. The Internet Protocol (IP) operates here, directing packets across network boundaries to their destinations.
- Network Access Layer: The lowest layer, managing physical connections and data transfer over hardware such as Ethernet or Wi-Fi.

TCP/IP breaks data into small packets, addresses them, routes them, and then reassembles them at the receiving device. This protocol suite enables different types of devices and networks to communicate seamlessly, regardless of their underlying hardware or software.

PACKET CAPTURE:

Packet capture is a technique used to intercept and record network data packets as they travel across computer networks. It enables network and cybersecurity professionals to save all communication occurring on a network interface so that each packet can be analyzed in detail. The collected packets are often stored in standard file formats such as .pcap or .pcapng. Tools like Wireshark use these files to inspect the data for troubleshooting, security analysis, and forensic investigations.


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Wi-Fi: <live capture in progress>
Packets: 584
Profile: Default

```
No.  Time       Source                 Destination      Protocol Length Info
 566 21.885063  10.10.74.151           239.255.255.250  SSDP      212 M-SEARCH * HTTP/1.1
 567 21.885222  HewlettPacka_a5:09:…   Broadcast        ARP        60 Who has 10.10.66.50? Tell 10.10.64.1
 568 21.886499  10.10.74.155           224.0.0.251      MDNS      510 Standard query response 0x0000 PTR, cache flush Android_92469e3a02ab49af9314dfd578ff4848.local AAAA, cache flush fe80::c05a:51ff:fe…
 569 21.886643  fe80::c05a:51ff:feb…   ff02::fb         MDNS      530 Standard query response 0x0000 PTR, cache flush Android_92469e3a02ab49af9314dfd578ff4848.local AAAA, cache flush fe80::c05a:51ff:fe…
 570 21.994580  40.104.66.210          10.10.74.136     TCP        54 443 → 25593 [ACK] Seq=1 Ack=1813 Win=255 Len=0
 571 21.994580  40.104.66.210          10.10.74.136     TLSv1.2   908 Application Data
 572 22.050261  10.10.74.136           40.104.66.210    TCP        54 25593 → 443 [ACK] Seq=1813 Ack=855 Win=252 Len=0
 573 22.118162  10.10.74.136           20.190.145.141   TLSv1.2   346 Application Data
 574 22.118389  10.10.74.136           20.190.145.141   TCP      1494 25512 → 443 [ACK] Seq=293 Ack=1 Win=255 Len=1440 [TCP PDU reassembled in 578]
 575 22.118389  10.10.74.136           20.190.145.141   TCP      1494 25512 → 443 [ACK] Seq=1733 Ack=1 Win=255 Len=1440 [TCP PDU reassembled in 578]
 576 22.118389  10.10.74.136           20.190.145.141   TCP      1494 25512 → 443 [ACK] Seq=3173 Ack=1 Win=255 Len=1440 [TCP PDU reassembled in 578]
 577 22.118389  10.10.74.136           20.190.145.141   TCP      1494 25512 → 443 [ACK] Seq=4613 Ack=1 Win=255 Len=1440 [TCP PDU reassembled in 578]
 578 22.118389  10.10.74.136           20.190.145.141   TLSv1.2   925 Application Data
 579 22.195377  10.10.73.116           224.0.0.251      MDNS      119 Standard query 0x003a PTR _googlecast._tcp.local, "QM" question PTR _674A0243._sub._googlecast._tcp.local, "QM" question PTR _8E6C8…
 580 22.195377  fe80::48ae:2dff:fef…   ff02::fb         MDNS      139 Standard query 0x003a PTR _googlecast._tcp.local, "QM" question PTR _674A0243._sub._googlecast._tcp.local, "QM" question PTR _8E6C8…
 581 22.198246  20.190.145.141         10.10.74.136     TCP        54 443 → 25512 [ACK] Seq=1 Ack=1733 Win=49155 Len=0
 582 22.198246  20.190.145.141         10.10.74.136     TCP        54 443 → 25512 [ACK] Seq=1 Ack=4613 Win=49155 Len=0
 583 22.198246  20.190.145.141         10.10.74.136     TCP        54 443 → 25512 [ACK] Seq=1 Ack=6924 Win=49155 Len=0
 584 22.519939  20.190.145.141         10.10.74.136     TCP      1514 443 → 25512 [ACK] Seq=1 Ack=6924 Win=49155 Len=1460
```

```
▸ Frame 1: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface \Device\NPF_{677CD686-85
▸ Ethernet II, Src: AzureWaveTec_cb:61:f0 (60:ff:9e:cb:61:f0), Dst: HewlettPacka_a5:09:00 (94:60:d5:a5:09:
▸ Internet Protocol Version 4, Src: 10.10.74.136, Dst: 192.168.1.5
▸ User Datagram Protocol, Src Port: 58366, Dst Port: 53
▸ Domain Name System (query)
```

```
0000  94 60 d5 a5 09 00 60 ff  9e cb 61 f0 08 00 45 00   ·`····`···a···E·
0010  00 3e e3 4b 00 00 80 11  41 24 0a 0a 4a 88 c0 a8   ·>·K····A$··J···
0020  01 05 e3 fe 00 35 00 2a  02 20 9f 70 01 00 00 01   ·····5·*· ·p····
0030  00 00 00 00 00 00 02 69  6e 09 67 72 61 6d 6d 61   ·······in·gramma
0040  72 6c 79 03 63 6f 6d 00  00 01 00 01               rly·com·····
```

Packet capture allows analysts to examine both the headers, which contain vital information like source and destination addresses, protocol types, and timestamps, as well as the payloads, which hold the actual data being transmitted. This deep visibility supports the detection of security threats, diagnosis of network problems, performance optimization, and the investigation of suspicious or malicious activities.

PROTOCOL ANALYSIS:

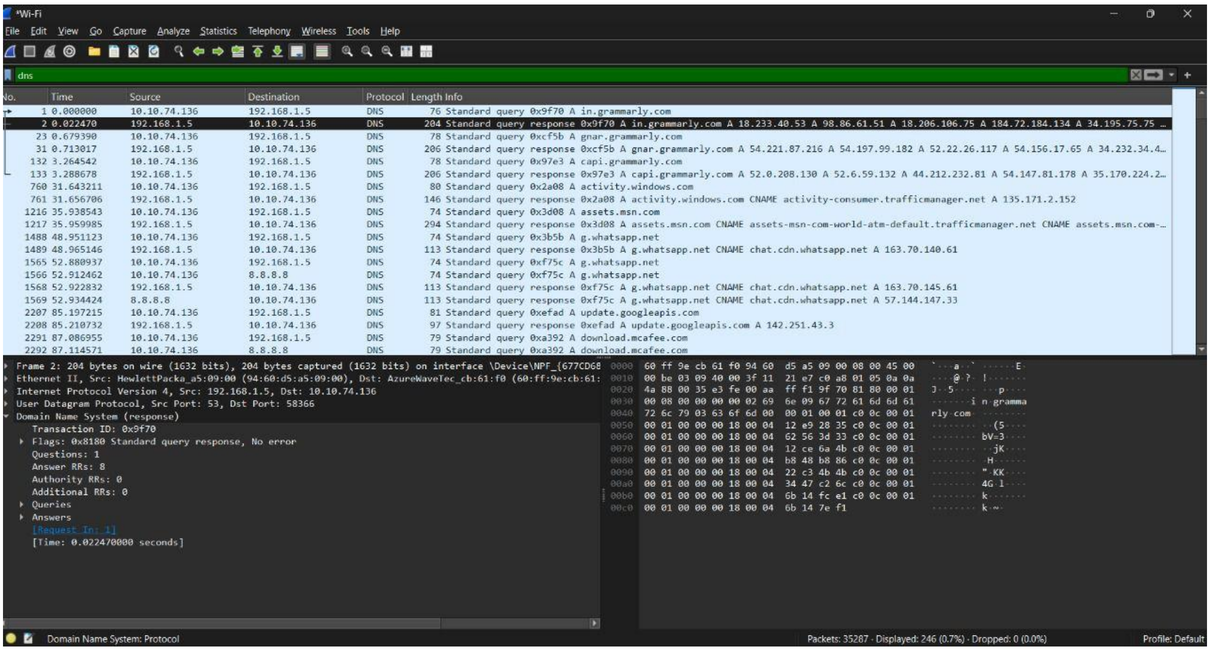
Protocol analysis is the systematic examination of network communication protocols to understand how data is exchanged between devices on a network.

It involves analyzing message structures, sequences, and meanings to ensure communication adheres to protocol standards. Protocol analysis helps identify normal and abnormal behaviors, troubleshoot network issues, detect security threats, and optimize network performance. This process can be applied to both well-known protocols like TCP, DNS, and HTTP, as well as unknown protocols through reverse engineering.

DNS TRAFFIC CAPTURE AND PROTOCOL ANALYSIS:

The displayed screenshot features Domain Name System (DNS) packets filtered with the dns filter in Wireshark. It shows DNS standard queries and responses that resolve domain names to their IP addresses.

Packet details include query names such as google.com, types of DNS records queried (A, AAAA, CNAME), and corresponding responses from DNS servers.



*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns

```
No.   Time      Source        Destination   Protocol Length Info
    1 0.000000  10.10.74.136  192.168.1.5   DNS       76 Standard query 0x9f70 A in.grammarly.com
    2 0.022470  192.168.1.5   10.10.74.136  DNS      204 Standard query response 0x9f70 A in.grammarly.com A 18.233.40.53 A 98.86.61.51 A 18.206.106.75 A 184.72.184.134 A 34.195.75.75 …
   23 0.679390  10.10.74.136  192.168.1.5   DNS       78 Standard query 0xcf5b A gnar.grammarly.com
   31 0.713017  192.168.1.5   10.10.74.136  DNS      206 Standard query response 0xcf5b A gnar.grammarly.com A 54.221.87.216 A 54.197.99.182 A 52.22.26.117 A 54.156.17.65 A 34.232.34.4…
  132 3.264542  10.10.74.136  192.168.1.5   DNS       78 Standard query 0x97e3 A capi.grammarly.com
  133 3.288678  192.168.1.5   10.10.74.136  DNS      206 Standard query response 0x97e3 A capi.grammarly.com A 52.0.208.130 A 52.6.59.132 A 44.212.232.81 A 54.147.81.178 A 35.170.224.2…
  760 31.643211 10.10.74.136  192.168.1.5   DNS       80 Standard query 0x2a08 A activity.windows.com
  761 31.656706 192.168.1.5   10.10.74.136  DNS      146 Standard query response 0x2a08 A activity.windows.com CNAME activity-consumer.trafficmanager.net A 135.171.2.152
 1216 35.938543 10.10.74.136  192.168.1.5   DNS       74 Standard query 0x3d08 A assets.msn.com
 1217 35.959985 192.168.1.5   10.10.74.136  DNS      294 Standard query response 0x3d08 A assets.msn.com CNAME assets-msn-com-world-atm-default.trafficmanager.net CNAME assets.msn.com-…
 1488 48.951123 10.10.74.136  192.168.1.5   DNS       74 Standard query 0x3b5b A g.whatsapp.net
 1489 48.965146 192.168.1.5   10.10.74.136  DNS      113 Standard query response 0x3b5b A g.whatsapp.net CNAME chat.cdn.whatsapp.net A 163.70.140.61
 1565 52.880937 10.10.74.136  192.168.1.5   DNS       74 Standard query 0xf75c A g.whatsapp.net
 1566 52.912462 10.10.74.136  8.8.8.8       DNS       74 Standard query 0xf75c A g.whatsapp.net
 1568 52.922832 192.168.1.5   10.10.74.136  DNS      113 Standard query response 0xf75c A g.whatsapp.net CNAME chat.cdn.whatsapp.net A 163.70.145.61
 1569 52.934424 8.8.8.8       10.10.74.136  DNS      113 Standard query response 0xf75c A g.whatsapp.net CNAME chat.cdn.whatsapp.net A 57.144.147.33
 2207 85.197215 10.10.74.136  192.168.1.5   DNS       81 Standard query 0xefad A update.googleapis.com
 2208 85.210732 192.168.1.5   10.10.74.136  DNS       97 Standard query response 0xefad A update.googleapis.com A 142.251.43.3
 2291 87.086955 10.10.74.136  192.168.1.5   DNS       79 Standard query 0xa392 A download.mcafee.com
 2292 87.114571 10.10.74.136  8.8.8.8       DNS       79 Standard query 0xa392 A download.mcafee.com
```

```
> Frame 2: 204 bytes on wire (1632 bits), 204 bytes captured (1632 bits) on interface \Device\NPF_{677CD68
> Ethernet II, Src: HewlettPacka_a5:09:00 (94:60:d5:a5:09:00), Dst: AzureWaveTec_cb:61:f0 (60:ff:9e:cb:61:
> Internet Protocol Version 4, Src: 192.168.1.5, Dst: 10.10.74.136
> User Datagram Protocol, Src Port: 53, Dst Port: 58366
v Domain Name System (response)
    Transaction ID: 0x9f70
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 8
    Authority RRs: 0
    Additional RRs: 0
  > Queries
  > Answers
    [Request In: 1]
    [Time: 0.022470000 seconds]
```

```
0000  60 ff 9e cb 61 f0 94 60  d5 a5 09 00 08 00 45 00   `···a··` ······E·
0010  00 be 03 09 40 00 3f 11  21 e7 c0 a8 01 05 0a 0a   ····@·?· !·······
0020  4a 88 00 35 e3 fe 00 aa  ff f1 9f 70 81 80 00 01   J··5···· ···p····
0030  00 08 00 00 00 00 02 69  6e 09 67 72 61 6d 6d 61   ·······i n·gramma
0040  72 6c 79 03 63 6f 6d 00  00 01 00 01 c0 0c 00 01   rly·com· ········
0050  00 01 00 00 00 18 00 04  12 e9 28 35 c0 0c 00 01   ········ ··(5····
0060  00 01 00 00 00 18 00 04  62 56 3d 33 c0 0c 00 01   ········ bV=3····
0070  00 01 00 00 00 18 00 04  12 ce 6a 4b c0 0c 00 01   ········ ··jK····
0080  00 01 00 00 00 18 00 04  b8 48 b8 86 c0 0c 00 01   ········ ·H······
0090  00 01 00 00 00 18 00 04  22 c3 4b 4b c0 0c 00 01   ········ "·KK····
00a0  00 01 00 00 00 18 00 04  34 47 c2 6c c0 0c 00 01   ········ 4G·l····
00b0  00 01 00 00 00 18 00 04  6b 14 fc e1 c0 0c 00 01   ········ k·······
00c0  00 01 00 00 00 18 00 04  6b 14 7e f1               ········ k·~·
```

Domain Name System: Protocol
Packets: 35287 · Displayed: 246 (0.7%) · Dropped: 0 (0.0%)
Profile: Default

A total of 584 DNS packets were captured during this session. DNS is fundamental to Internet functionality by allowing users to use human-friendly domain names.

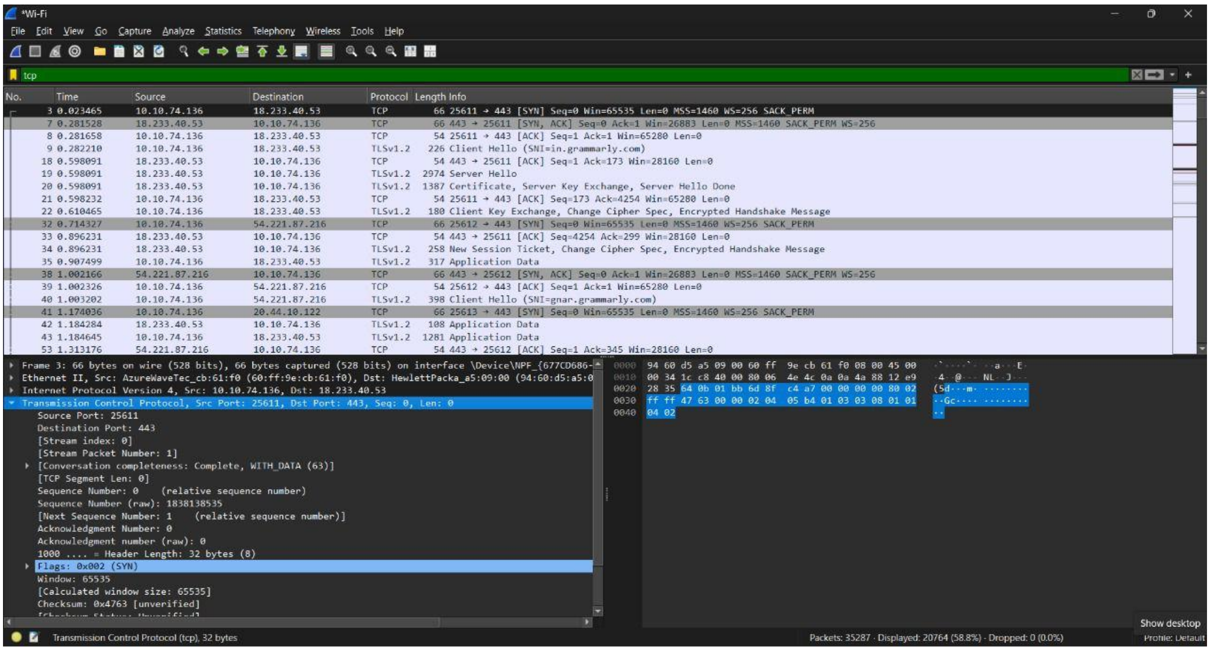
Analyzing DNS traffic helps detect fraud, phishing, or malware domains, and ensures that name resolution is functioning properly in the network.

Filtering DNS packets in Wireshark isolates domain resolution activity, making it easier to monitor DNS requests and responses for security and performance insights.

TCP TRAFFIC CAPTURE AND PROTOCOL ANALYSIS:

This screenshot highlights the Transmission Control Protocol (TCP) traffic filtered by the tcp filter in Wireshark. It captures the TCP packets showing connection establishment and ongoing data transmission between devices.

The packet details display critical TCP header information such as source and destination ports, flags (SYN, ACK), sequence numbers, and window sizes. The three-way handshake process (SYN, SYN-ACK, ACK) used to establish connections can be observed here.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

```
No.   Time      Source          Destination     Protocol Length Info
    3 0.023465  10.10.74.136    18.233.40.53    TCP        66 25611 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    7 0.281528  18.233.40.53    10.10.74.136    TCP        66 443 → 25611 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM WS=256
    8 0.281658  10.10.74.136    18.233.40.53    TCP        54 25611 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
    9 0.282210  10.10.74.136    18.233.40.53    TLSv1.2   226 Client Hello (SNI=in.grammarly.com)
   18 0.598091  18.233.40.53    10.10.74.136    TCP        54 443 → 25611 [ACK] Seq=1 Ack=173 Win=28160 Len=0
   19 0.598091  18.233.40.53    10.10.74.136    TLSv1.2  2974 Server Hello
   20 0.598091  18.233.40.53    10.10.74.136    TLSv1.2  1387 Certificate, Server Key Exchange, Server Hello Done
   21 0.598232  10.10.74.136    18.233.40.53    TCP        54 25611 → 443 [ACK] Seq=173 Ack=4254 Win=65280 Len=0
   22 0.610465  10.10.74.136    18.233.40.53    TLSv1.2   180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
   32 0.714327  10.10.74.136    54.221.87.216   TCP        66 25612 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
   33 0.896231  18.233.40.53    10.10.74.136    TCP        54 443 → 25611 [ACK] Seq=4254 Ack=299 Win=28160 Len=0
   34 0.896231  18.233.40.53    10.10.74.136    TLSv1.2   258 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
   35 0.907499  10.10.74.136    18.233.40.53    TLSv1.2   317 Application Data
   38 1.002166  54.221.87.216   10.10.74.136    TCP        66 443 → 25612 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM WS=256
   39 1.002326  10.10.74.136    54.221.87.216   TCP        54 25612 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
   40 1.003202  10.10.74.136    54.221.87.216   TLSv1.2   398 Client Hello (SNI=gnar.grammarly.com)
   41 1.174036  10.10.74.136    20.44.10.122    TCP        66 25613 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
   42 1.184284  18.233.40.53    10.10.74.136    TLSv1.2   108 Application Data
   43 1.184645  10.10.74.136    18.233.40.53    TLSv1.2  1281 Application Data
   53 1.313176  54.221.87.216   10.10.74.136    TCP        54 443 → 25612 [ACK] Seq=1 Ack=345 Win=28160 Len=0
```

```
▸ Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{677CD686-
▸ Ethernet II, Src: AzureWaveTec_cb:61:f0 (60:ff:9e:cb:61:f0), Dst: HewlettPacka_a5:09:00 (94:60:d5:a5:0
▸ Internet Protocol Version 4, Src: 10.10.74.136, Dst: 18.233.40.53
▾ Transmission Control Protocol, Src Port: 25611, Dst Port: 443, Seq: 0, Len: 0
    Source Port: 25611
    Destination Port: 443
    [Stream index: 0]
    [Stream Packet Number: 1]
  ▸ [Conversation completeness: Complete, WITH_DATA (63)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1838138535
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x002 (SYN)
    Window: 65535
    [Calculated window size: 65535]
    Checksum: 0x4763 [unverified]
```

```
0000  94 60 d5 a5 09 00 60 ff  9e cb 61 f0 08 00 45 00   ·`····`·  ··a···E·
0010  00 34 1c c8 40 00 80 06  4e 4c 0a 0a 4a 88 12 e9   ·4··@···  NL··J···
0020  28 35 64 0b 01 bb 6d 8f  c4 a7 00 00 00 00 80 02   (5d···m·  ········
0030  ff ff 47 63 00 00 02 04  05 b4 01 03 03 08 01 01   ··Gc····  ········
0040  04 02                                              ··
```

Transmission Control Protocol (tcp), 32 bytes

Packets: 35287 · Displayed: 20764 (58.8%) · Dropped: 0 (0.0%)

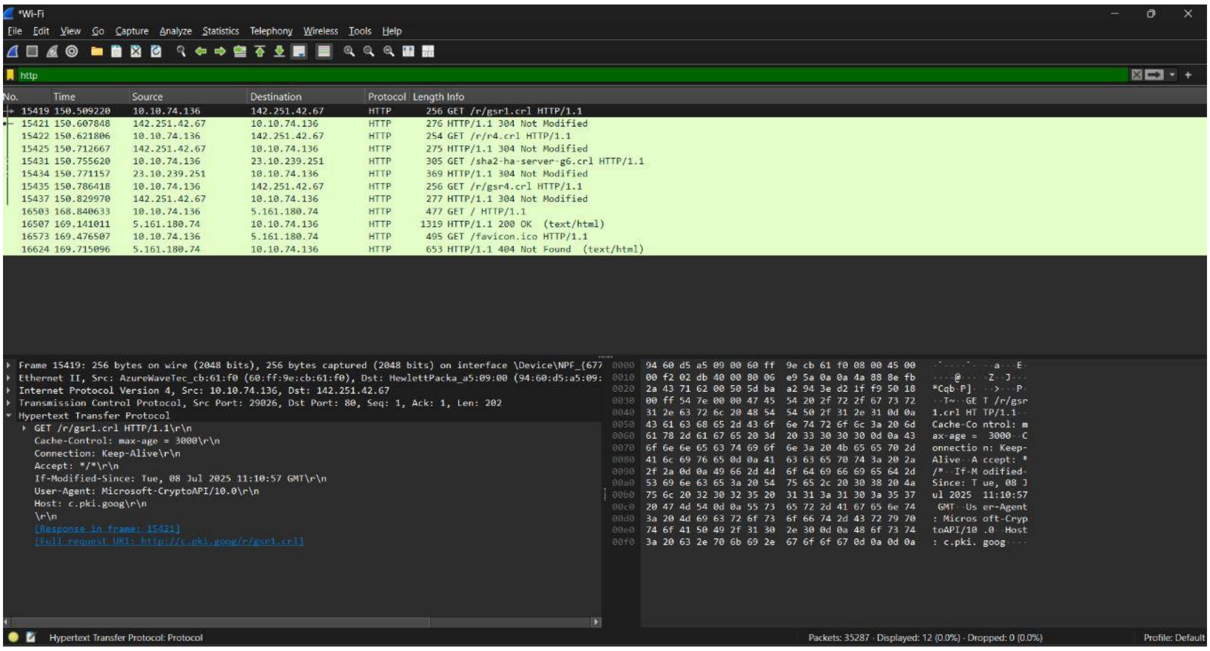
TCP is a reliable, connection-oriented protocol underlying many internet services like HTTP and HTTPS. Analyzing TCP packets reveals connection behaviors, performance issues, and potential network anomalies.

Using the TCP filter in Wireshark helps focus on the transport layer traffic, allowing deep dive into connection setups, data flow, and teardown processes necessary for network troubleshooting.

HTTP TRAFFIC CAPTURE AND PROTOCOL ANALYSIS:

The screenshot below shows HTTP traffic captured and filtered in Wireshark using the http filter. It displays HTTP GET requests and responses between the client and web servers. Packet details reveal request URLs, HTTP methods, response status codes such as 200 (OK) and 304 (Not Modified), and headers like User-Agent.

A total of 35,287 packets were captured during this session, which included a significant number of HTTP packets. HTTP is the foundation of data communication on the World Wide Web, allowing browsers to request web pages and servers to respond with content.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Profile: Default

http

```
No.    Time         Source          Destination     Protocol Length Info
15419 150.509220   10.10.74.136    142.251.42.67   HTTP      256 GET /r/gsr1.crl HTTP/1.1
15421 150.607848   142.251.42.67   10.10.74.136    HTTP      276 HTTP/1.1 304 Not Modified
15422 150.621806   10.10.74.136    142.251.42.67   HTTP      254 GET /r/r4.crl HTTP/1.1
15425 150.712667   142.251.42.67   10.10.74.136    HTTP      275 HTTP/1.1 304 Not Modified
15431 150.755620   10.10.74.136    23.10.239.251   HTTP      305 GET /sha2-ha-server-g6.crl HTTP/1.1
15434 150.771157   23.10.239.251   10.10.74.136    HTTP      369 HTTP/1.1 304 Not Modified
15435 150.786418   10.10.74.136    142.251.42.67   HTTP      256 GET /r/gsr4.crl HTTP/1.1
15437 150.829970   142.251.42.67   10.10.74.136    HTTP      277 HTTP/1.1 304 Not Modified
16503 168.840633   10.10.74.136    5.161.180.74    HTTP      477 GET / HTTP/1.1
16507 169.141011   5.161.180.74    10.10.74.136    HTTP     1319 HTTP/1.1 200 OK  (text/html)
16573 169.476507   10.10.74.136    5.161.180.74    HTTP      495 GET /favicon.ico HTTP/1.1
16624 169.715096   5.161.180.74    10.10.74.136    HTTP      653 HTTP/1.1 404 Not Found  (text/html)
```

```
> Frame 15419: 256 bytes on wire (2048 bits), 256 bytes captured (2048 bits) on interface \Device\NPF_{677
> Ethernet II, Src: AzureWaveTec_cb:61:f0 (60:ff:9e:cb:61:f0), Dst: HewlettPacka_a5:09:00 (94:60:d5:a5:09:
> Internet Protocol Version 4, Src: 10.10.74.136, Dst: 142.251.42.67
> Transmission Control Protocol, Src Port: 29026, Dst Port: 80, Seq: 1, Ack: 1, Len: 202
v Hypertext Transfer Protocol
  > GET /r/gsr1.crl HTTP/1.1\r\n
    Cache-Control: max-age = 3000\r\n
    Connection: Keep-Alive\r\n
    Accept: */*\r\n
    If-Modified-Since: Tue, 08 Jul 2025 11:10:57 GMT\r\n
    User-Agent: Microsoft-CryptoAPI/10.0\r\n
    Host: c.pki.goog\r\n
    \r\n
    [Response in frame: 15421]
    [Full request URI: http://c.pki.goog/r/gsr1.crl]
```

```
0000  94 60 d5 a5 09 00 60 ff  9e cb 61 f0 08 00 45 00   ·`····`· ··a···E·
0010  00 f2 02 db 40 00 80 06  e9 5a 0a 0a 4a 88 8e fb   ····@··· ·Z··J···
0020  2a 43 71 62 00 50 5d ba  a2 94 3e d2 1f f9 50 18   *Cqb·P]· ··>···P·
0030  00 ff 54 7e 00 00 47 45  54 20 2f 72 2f 67 73 72   ··T~··GE T /r/gsr
0040  31 2e 63 72 6c 20 48 54  54 50 2f 31 2e 31 0d 0a   1.crl HT TP/1.1··
0050  43 61 63 68 65 2d 43 6f  6e 74 72 6f 6c 3a 20 6d   Cache-Co ntrol: m
0060  61 78 2d 61 67 65 20 3d  20 33 30 30 30 0d 0a 43   ax-age =  3000··C
0070  6f 6e 6e 65 63 74 69 6f  6e 3a 20 4b 65 65 70 2d   onnectio n: Keep-
0080  41 6c 69 76 65 0d 0a 41  63 63 65 70 74 3a 20 2a   Alive··A ccept: *
0090  2f 2a 0d 0a 49 66 2d 4d  6f 64 69 66 69 65 64 2d   /*··If-M odified-
00a0  53 69 6e 63 65 3a 20 54  75 65 2c 20 30 38 20 4a   Since: T ue, 08 J
00b0  75 6c 20 32 30 32 35 20  31 31 3a 31 30 3a 35 37   ul 2025  11:10:57
00c0  20 47 4d 54 0d 0a 55 73  65 72 2d 41 67 65 6e 74    GMT··Us er-Agent
00d0  3a 20 4d 69 63 72 6f 73  6f 66 74 2d 43 72 79 70   : Micros oft-Cryp
00e0  74 6f 41 50 49 2f 31 30  2e 30 0d 0a 48 6f 73 74   toAPI/10 .0··Host
00f0  3a 20 63 2e 70 6b 69 2e  67 6f 6f 67 0d 0a 0d 0a   : c.pki. goog····
```

Hypertext Transfer Protocol: Protocol

Packets: 35287 · Displayed: 12 (0.0%) · Dropped: 0 (0.0%)

Protocol analysis of HTTP traffic helps verify legitimate web browsing activity, detect unusual or malicious HTTP requests, and understand how resources are loaded on a network.

Filtering specifically by HTTP protocol in Wireshark aids in isolating web traffic from other network data, making it easier to analyze web communication patterns in detail.

FILTERING:

Filtering in network analysis refers to the process of selecting and displaying only specific packets from a large set of captured network traffic based on defined criteria. Using filters makes it easier to focus on relevant data by excluding all other traffic that does not match the criteria.

In tools like Wireshark, filters can be applied to narrow down packets by protocol type (e.g., HTTP, DNS, TCP), IP address, port number, packet size, or other header and payload fields. This allows analysts to isolate particular communications or protocol behaviors for detailed examination.

Filtering is essential for efficient protocol analysis because it reduces noise from unrelated traffic and helps quickly identify patterns, problems, or suspicious activity within the network traffic. Filters in Wireshark are applied through expressions typed in the filter bar, enabling real-time refined views of captured data.

NETWORK TROUBLESHOOTING:

Network troubleshooting is the systematic process of diagnosing and resolving network problems that affect the performance, availability, or security of network communications. It involves identifying the root cause of connectivity issues, slow network speeds, packet loss, or security breaches.

Common steps in network troubleshooting include:

- Packet capture and analysis: Using tools like Wireshark to capture and examine network traffic for abnormalities or failures.
- Protocol analysis: Understanding whether the protocols involved are functioning correctly.
- Connectivity testing: Using commands like ping and traceroute to check communication paths.
- Configuration verification: Ensuring devices and network settings are correct.
- Performance monitoring: Tracking network latency, throughput, and errors to identify bottlenecks.
- Effective network troubleshooting relies on careful observation, systematic testing, and knowledge of network protocols and architectures to restore smooth and secure network operation.

CONCLUSION:

The network traffic capture and analysis using Wireshark provided valuable insights into the functioning of different network protocols such as TCP, DNS, and HTTP. Through protocol analysis and filtering, it was possible to isolate specific communications, understand packet structures, and observe their behaviors in real network conditions.

This practical exercise enhanced understanding of the TCP/IP protocol suite and its role in facilitating reliable internet communication. It also demonstrated the importance of network troubleshooting by revealing normal and potential anomalous traffic patterns.

Overall, the task strengthened skills in using advanced network analysis tools, applying filters effectively, and documenting technical findings comprehensively. Such expertise is vital for real-world cybersecurity operations, network management, and incident response, underscoring the significance of packet capturing and protocol analysis in maintaining secure and efficient networks.

my_task5.pcap

****