OS assignment 3

Capture and Analyze Packets

I captured packets while browsing the websites Medium and PubMed. I started Wireshark, selected the Wi-Fi interface (en0), and began capturing network traffic. While the capture was running, I opened my browser and visited Medium and PubMed. I browsed articles and interacted with the websites to generate network activity. Once sufficient traffic was captured, I stopped the capture and saved the packet data for analysis.

Verifying Device IP Address

To confirm that the captured packets originated from my device, I ran the ifconfig command on my MacBook. The output showed the following details:

- The active network interface in use was en0.
- The assigned IPv4 address for my device was 10.100.102.12.
- This matches the source IP seen in the Wireshark packet captures, confirming that the analyzed packets were generated by my device.

TCP packets :

The Transmission Control Protocol (TCP) is a transport layer protocol designed for reliable communication. It ensures data is delivered in order and without errors between devices. Below is a filtered view of TCP packets from my Wireshark capture:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	104.208.16.89	10.100.102.12	TLSv1…	165	Hello Retry Request, Change Cipher Spec
2	0.001664	10.100.102.12	104.208.16.89	TCP	66	61172 → 443 [ACK] Seq=1 Ack=100 Win=2051 Len=0 TSval=2938810829 TSecr=20817158
3	0.001669	10.100.102.12	104.208.16.89	TLSv1…	451	Change Cipher Spec, Client Hello (SNI=self.events.data.microsoft.com)
4	0.184090	104.208.16.89	10.100.102.12	TLSv1…	1506	Server Hello
5	0.186273	104.208.16.89	10.100.102.12	TCP	1506	443 → 61172 [ACK] Seq=1540 Ack=386 Win=16382 Len=1440 TSval=208171768 TSecr=29
6	0.186277	104.208.16.89	10.100.102.12	TCP	1506	443 → 61172 [ACK] Seq=2980 Ack=386 Win=16382 Len=1440 TSval=208171768 TSecr=29
7	0.186283	104.208.16.89	10.100.102.12	TCP	1506	443 → 61172 [ACK] Seq=4420 Ack=386 Win=16382 Len=1440 TSval=208171768 TSecr=29
8	0.186288	104.208.16.89	10.100.102.12	TLSv1…	712	Application Data
9	0.187550	10.100.102.12	104.208.16.89	TCP	66	61172 → 443 [ACK] Seq=386 Ack=1540 Win=2028 Len=0 TSval=2938811013 TSecr=20817
10	0.188292	10.100.102.12	104.208.16.89	TCP	66	61172 → 443 [ACK] Seq=386 Ack=6506 Win=2048 Len=0 TSval=2938811016 TSecr=20817
11	0.202600	10.100.102.12	104.208.16.89	TLSv1…	140	Application Data
12	0.204225	10.100.102.12	104.208.16.89	TLSv1…	479	Application Data
13	0.204558	10.100.102.12	104.208.16.89	TCP	1494	61172 → 443 [ACK] Seq=873 Ack=6506 Win=2048 Len=1428 TSval=2938811033 TSecr=20
14	0.204561	10.100.102.12	104.208.16.89	TCP	1494	61172 → 443 [ACK] Seq=2301 Ack=6506 Win=2048 Len=1428 TSval=2938811033 TSecr=2
15	0.204584	10.100.102.12	104.208.16.89	TLSv1…	1447	Application Data
16	0.361691	104.208.16.89	10.100.102.12	TCP	66	443 → 61172 [ACK] Seq=6506 Ack=873 Win=16380 Len=0 TSval=208171948 TSecr=29388

To better understand the packet details, I will analyze **Packet 16** from the filtered results:

- Timestamp: 0.361691 seconds since the start of the capture.
- Source: 104.208.16.89 (remote server).
- Destination: 10.100.102.12 (my device).
- Ports:
  - Source Port: 443 (used for HTTPS communication).
  - Destination Port: 61172 (ephemeral port assigned to my device).
- Packet Length: 66 bytes.
- Header Details:
  - Sequence Number: 6506 (indicates the current sequence of data from the server).
  - Acknowledgment Number: 873 (acknowledges receipt of all data up to sequence number 872).

- Window Size: 16380 bytes (server's buffer space for receiving additional data).
- Payload Length: 0 bytes (no data in this acknowledgment packet).
- Flags: [ACK] (indicating acknowledgment).

Since the **payload length is** 0 **bytes**, the entire packet consists of headers from various protocol layers.

UDP packets :

The User Datagram Protocol (UDP) is a transport layer protocol designed for fast, connectionless communication. Unlike TCP, UDP does not guarantee delivery or maintain order, making it ideal for time-sensitive applications like DNS queries, video streaming, or online gaming.

Below is a filtered view of UDP packets from my Wireshark capture:

No.	Time	Source	Destination	Protocol	Length	Info
27	0.622493	10.100.102.3	255.255.255.…	UDP	230	49154 → 6667 Len=188
28	2.084989	10.100.102.12	10.100.102.1	DNS	79	Standard query 0x7e9d HTTPS clients1.google.com
29	2.085283	10.100.102.12	10.100.102.1	DNS	79	Standard query 0x20ad A clients1.google.com
30	2.096912	10.100.102.1	10.100.102.12	DNS	153	Standard query response 0x7e9d HTTPS clients1.google.com CNAME clients
31	2.097603	10.100.102.1	10.100.102.12	DNS	119	Standard query response 0x20ad A clients1.google.com CNAME clients.l.g
32	2.098018	10.100.102.12	10.100.102.1	DNS	80	Standard query 0x86ee HTTPS clients.l.google.com
33	2.101997	10.100.102.1	10.100.102.12	DNS	130	Standard query response 0x86ee HTTPS clients.l.google.com SOA ns1.goog
66	2.655306	10.100.102.12	172.224.169.…	UDP	83	56923 → 443 Len=41
67	2.713139	172.224.169.…	10.100.102.12	UDP	74	443 → 56923 Len=32
103	4.562486	10.100.102.12	142.250.75.68	UDP	79	62615 → 443 Len=37
104	4.572973	142.250.75.68	10.100.102.12	UDP	77	443 → 62615 Len=35
105	4.586514	10.100.102.12	142.250.75.68	UDP	159	62615 → 443 Len=117
106	4.593579	142.250.75.68	10.100.102.12	UDP	82	443 → 62615 Len=40
107	4.602382	10.100.102.12	142.250.75.68	UDP	76	62615 → 443 Len=34

To better understand the packet details, I will analyze **Packet 103** from the filtered results:

- Timestamp: 4.562486 seconds since the start of the capture.
- Source: 10.100.102.12 (my device sending the packet).
- Destination: 142.250.75.68 (the remote server receiving the packet).
- Ports:
  - Source Port: 62615
  - Destination Port: 443
- Packet Length: 79 bytes.
  - Payload Length: 37 bytes (the application data sent by this packet).
- Protocol: UDP (User Datagram Protocol).
- Header Details:
  - UDP does not have flags like TCP. Instead, it relies on its simple structure with only source port, destination port, length, and checksum fields.
  - The remaining bytes (42 bytes) are composed of protocol headers, including the UDP header (8 bytes) and IP header (20 bytes).

HTTP packets :

The Hypertext Transfer Protocol (HTTP) is a widely used application-layer protocol for transferring data between clients and servers. Modern web communication typically occurs over HTTPS, which combines HTTP with encryption using TLS for security.

In the captured packets, **two HTTP packets are visible**. The first HTTP packet uses the **CONNECT method**, which is part of the process for establishing a secure tunnel for HTTPS communication. The **CONNECT method** is often used when a client connects to a proxy server to request a connection to an external host over HTTPS. The second HTTP packet is the server's response (HTTP/1.0 200 Connection established), confirming that the secure tunnel has been successfully created.

After the secure tunnel is established, all subsequent HTTP traffic is encrypted and transmitted over **TLSv1.3 packets**. These packets are labeled as **Application Data** in Wireshark, making their contents (HTTP requests and responses) inaccessible without decrypting the TLS session. This explains why only the initial HTTP setup packets are visible in the captured data.

By encrypting HTTP traffic, HTTPS ensures confidentiality and integrity, preventing unauthorized access or modification of the transmitted data.

No.	Time	Source	Destination	Protocol	Length	Info
5668	30.3220…	10.100.102.12	2.16.149.156	HTTP	219	CONNECT proxy-safebrowsing.googleapis.com:443 HTTP/1.1
5670	30.3766…	2.16.149.156	10.100.102.12	HTTP	105	HTTP/1.0 200 Connection established
5672	30.3779…	10.100.102.12	2.16.149.156	TLSv1.3	583	Client Hello (SNI=proxy-safebrowsing.googleapis.com)
5678	30.4453…	2.16.149.156	10.100.102.12	TLSv1.3	1506	Server Hello, Change Cipher Spec
5685	30.4988…	2.16.149.156	10.100.102.12	TLSv1.3	239	Application Data
5687	30.5037…	10.100.102.12	2.16.149.156	TLSv1.3	130	Change Cipher Spec, Application Data
5688	30.5048…	10.100.102.12	2.16.149.156	TLSv1.3	730	Application Data
5689	30.5589…	2.16.149.156	10.100.102.12	TLSv1.3	680	Application Data, Application Data
5691	30.5594…	10.100.102.12	2.16.149.156	TLSv1.3	97	Application Data
5692	30.5608…	2.16.149.156	10.100.102.12	TLSv1.3	97	Application Data
5693	30.5608…	2.16.149.156	10.100.102.12	TLSv1.3	1506	Application Data
5694	30.5608…	2.16.149.156	10.100.102.12	TLSv1.3	1506	Application Data
5696	30.6125…	2.16.149.156	10.100.102.12	TLSv1.3	1506	Application Data
5698	30.6149…	2.16.149.156	10.100.102.12	TLSv1.3	340	Application Data, Application Data, Application Data
5700	30.6152…	10.100.102.12	2.16.149.156	TLSv1.3	105	Application Data

Packet **5668** Analysis: HTTP CONNECT Request

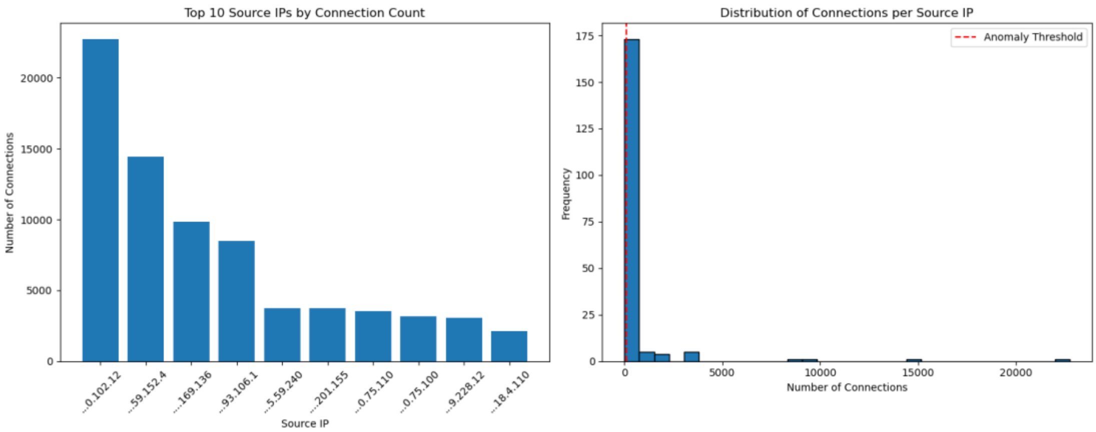
- Timestamp: 30.322 seconds since the start of the capture.
- Source: 10.100.102.12 (my device).
- Destination: 2.16.149.156 (proxy server for Google Safe Browsing).
- Length: 219 bytes.
- Packet Info:
  - Request Method: CONNECT.
  - Target Host: proxy-safebrowsing.googleapis.com.
  - Port: 443 (used for secure HTTPS communication).

Identify Abnormal Patterns

I decided to focuses on identifying sources that connect to an unusually high number of unique destinations, which might indicate network scanning or reconnaissance activity.

The main workflow is:

1. Capture network traffic in Wireshark and save as a PCAP file
2. The script reads this file packet by packet
3. For each packet, it tracks:
    - Who sent it (source IP)
    - Where it went (destination IP)
    - How many times each source-destination pair occurs
4. It looks for sources that connect to many different destinations
    - This could indicate scanning behavior
    - If a source connects to more than threshold destinations, it's flagged
5. Finally, it visualizes:
    - Most active source IPs
    - Distribution of how many connections each source makes



Looking at the network traffic analysis visualization, there are some interesting patterns to observe:

1. <u>From the left graph (Top 10 Source IPs):</u>

- One IP address (ending in 102.12) shows significantly higher activity, with over 20,000 connections
- There's a steep drop-off to the second most active IP (ending in 152.4) with around 14,000 connections
- The remaining IPs show a gradual decrease in activity from about 9,000 down to 2,000 connections

2. <u>From the right graph (Distribution of Connections):</u>

- The vast majority of source IPs (around 175) have very few connections (clustered near 0). There's a long tail distribution, with very few IPs having high connection counts

This pattern suggests:

- Highly asymmetric network usage, with a few IPs generating most of the traffic
- The IP ending in 102.12 shows unusually high activity that might warrant investigation.

Note: I attached the code as a separate file