Communication networks final project

Participants:

Omri Besan 206607905
Leon Pasternak 314789348
Adham Hamoudy 214225187
Mahasin Hamood 212933907

Part I: Answer the following Questions:

1. A user reports that their file transfer is slow, and you need to analyze the transport layer to identify the potential reasons. What factors could contribute to the slow transfer, and how would you troubleshoot it?
2. Analyze the effects of TCP's flow control mechanism on data transmission. How would it impact performance when the sender has significantly higher processing power than the receiver?
3. Analyze the role of routing in a network where multiple paths exist between the source and destination. How does the path choice affect network performance, and what factors should be considered in routing decisions?
4. How does MPTCP improve network performance?
5. You are monitoring network traffic and notice high packet loss between two routers. Analyze the potential causes for packet loss at the Network and Transport Layers and recommend steps to resolve the issue.

Answers:

1. A slow file transfer could be caused by various transport-layer factors:

- **Congestion Control**: TCP slows down transmission when congestion is detected. Analyze TCP congestion window size (CWND) and congestion avoidance algorithms like Reno, Cubic, or BBR.
- **Flow Control**: If the receiver has limited buffer space, TCP flow control (receiver window size) may limit the sender's speed.
- **Packet Loss & Retransmissions**: High packet loss triggers retransmissions and reduces TCP throughput (check for retransmissions using tools like Wireshark).
- **Network Latency**: High round-trip time (RTT) increases the time required for acknowledgments, slowing down the transfer.
- **MTU and Fragmentation**: If the Maximum Transmission Unit (MTU) is mismatched, excessive fragmentation can degrade performance.
- **TCP vs. UDP**: If using TCP, switching to a more efficient protocol like UDP (e.g., for real-time streaming) might improve performance.

Troubleshooting:

- Use `iperf` to measure bandwidth and diagnose throughput issues.
- Check for packet loss with `ping` or `traceroute`.
- Analyze TCP sessions with Wireshark to inspect retransmissions and congestion control behavior.
- Adjust TCP window scaling and congestion control settings if needed.

2. TCP flow control is designed to prevent the sender from overwhelming the receiver. It operates through the **receiver's advertised window (rwnd)**, which tells the sender how much data it can send before waiting for an acknowledgment.

Impact when Sender > Receiver in Processing Power:

- The **sender transmits faster than the receiver can process** packets.
- The **receiver reduces the advertised window size** (rwnd), limiting throughput.
- If the **rwnd drops to zero**, the sender pauses transmission until the receiver signals readiness.
- In severe cases, this leads to **TCP Zero Window conditions**, causing significant delays.

3. Routing determines the path packets take between a source and destination. Different routing choices impact network performance in terms of latency, packet loss, and congestion.

Effects of Path Choice on Performance:

- **Latency**: Longer paths introduce higher delays.
- **Bandwidth Availability**: Some routes may be more congested, leading to lower throughput.
- **Packet Loss**: Paths with higher congestion or poor-quality links may drop packets.
- **Reliability**: Redundant paths improve fault tolerance.

Factors in Routing Decisions:

- **Hop count (Shortest Path First - SPF)**: Algorithms like OSPF and RIP use this.
- **Latency and Bandwidth**: BGP and QoS-aware routing protocols consider link capacity.
- **Network Congestion**: Dynamic routing adjusts to congestion using metrics like EIGRP's bandwidth-delay product.
- **Load Balancing**: MPTCP or Equal-Cost Multi-Path (ECMP) distributes traffic across multiple paths.

Optimization Strategies:

- Use **Dynamic Routing Protocols (OSPF, BGP, EIGRP)** to adapt to changing conditions.
- Implement **Multipath Routing (e.g., MPTCP or ECMP)** for better load balancing.
- Apply **Traffic Engineering (e.g., MPLS-TE)** to optimize flow control.

4. **Multipath TCP (MPTCP)** enables TCP connections to use multiple network paths simultaneously, rather than a single path.

Benefits of MPTCP:

- **Increased Throughput**: Splits traffic across multiple links, combining bandwidth.
- **Improved Fault Tolerance**: If one path fails, the connection continues using alternate paths.
- **Reduced Congestion**: Dynamic path selection reduces reliance on a single congested link.
- **Better Mobile Connectivity**: Smartphones can use both Wi-Fi and LTE simultaneously.

Use Cases:

- **Data Centers**: Parallel paths between servers improve performance.
- **Mobile Networks**: Seamless transition between Wi-Fi and cellular data.
- **High-Availability Applications**: Streaming, cloud services, and critical communications.

5. **Potential Causes & Solutions:**

Layer	Cause	Resolution
Network (IP)	**Congestion** (buffer overflow at router)	Enable QoS, prioritize critical traffic
	Link failure (fiber cuts, hardware issues)	Check physical connectivity, redundancy
	MTU mismatch (fragmentation issues)	Adjust MTU settings or enable Path MTU Discovery
	Routing loops (misconfigured routes)	Verify BGP, OSPF, or static routing
Transport (TCP/UDP)	**Retransmissions** due to high RTT	Optimize congestion control (BBR, Cubic)
	UDP packet loss (no retransmission)	Use FEC (Forward Error Correction) for reliability
	Firewall drops (ACL rules blocking packets)	Check firewall logs and adjust rules

Part II: Papers:

Early Traffic Classification paper:

Main Contribution of the Paper

The paper introduces the hybrid RandomForest Traffic Classifier (hRFTC), a novel encrypted traffic classification algorithm. This classifier integrates unencrypted payload data from the **TLS handshake**, **flow-based time series data**, and **packet size statistics** to enhance classification accuracy. The paper demonstrates that hRFTC outperforms existing state-of-the-art classifiers, including **packet-based, flow-based, and hybrid models**.

Additionally, the study evaluates the ability of **early Traffic Classification (eTC) algorithms** to handle data heterogeneity under **Encrypted ClientHello (ECH) scenarios**. It also highlights a critical insight: even the best hybrid traffic classifiers need retraining when applied to different geographic locations due to varying traffic patterns.

Traffic Features Used and Novel Contributions

The paper utilizes a variety of traffic features, classified into three main categories:

1. **TLS Handshake Payload Features** – Extracted from unencrypted parts of the **ClientHello (CH) and ServerHello (SH)** messages.
2. **Flow-Based Features** – Derived from sequences of **packet sizes (PSs)** and **inter-packet times (IPTs)**.
3. **Packet Size Statistics** – Used to differentiate services and infer traffic types.

Novel Features:

- The paper introduces an **advanced set of flow-based features** and combines them with **TLS handshake data**, making it more resilient to **ECH encryption**.

- It **avoids reliance on Server Name Indication (SNI)**, which previous studies used but which is now increasingly masked in modern traffic.

Main Results and Insights

The study presents the following key findings:

1. **hRFTC Outperforms Existing Models** – The classifier achieves significantly **higher accuracy** than leading **packet-based and flow-based** classifiers.
2. **Packet Size Features Are Crucial** – The most influential classification features are related to **packet size statistics**, contributing to more than **50%** of classification importance.
3. **Geographic Variability Requires Retraining** – Classifiers trained in one location **do not generalize well** to different locations, indicating that regional network conditions impact classification performance.
4. **QUIC PADDING Complicates Classification** – The study finds that **QUIC traffic with PADDING frames reduces classification accuracy**, but the impact is still limited.
5. **Generalization Ability** – The hRFTC classifier **performs well even with reduced training data**, achieving similar accuracy to baseline classifiers even when trained on just **10% of the dataset**.

Analyzing HTTPS encrypted traffic to ide

Main Contribution of the Paper:

The paper shows that even though internet traffic is encrypted (mostly because everyone is using HTTPS nowadays) , it's still possible to figure out which operating system, browser and application a person is using just by sniffing and analyzing the traffic patterns.

The researchers used machine learning to do this and built a dataset of 20,000 sessions to train their model, they also introduced new traffic features that help make this classification more accurate, reaching up to 96.06% accuracy.

Traffic Features Used and Novel Contributions

The paper uses two types of traffic features:

1. Regular (Base) Features (already used in other research):
- Number of packets sent and received.
- Packet sizes(min, max, average, variance).
- Time between packets (min, max,average, standard deviation).
2. New (Novel) Features (introduced in this paper):
- SSL/TLS-related features: Things like the length of SSL session IDs, the number of encryption methods used, and extension counts.
- TCP features: how the data is packed and sent over the internet.
- Bursty behavior: This refers to how the browsers and applications send data in bursts rather than a steady flow, for example watching a Youtube video has different traffic bursts than scrolling on X (Twitter).

Main Results & Insights

- The model can accurately identify the OS, browser and application from encrypted traffic with 96.06% accuracy.
- New features improved accuracy significantly compared to using only base features.
- Operating System detection was nearly perfect.
- Browser classification was also very accurate, but sometimes Chrome and Safari got mixed up.
- Application classification was good, but Facebook traffic was often misclassified as “Unknown”
- The study proves that even with encryption, traffic patterns can still reveal a lot of information about a user, which has privacy and security implications.

FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition

Main Contribution of the Paper

The paper introduces **FlowPic**, a new way to classify encrypted internet traffic using **deep learning and image recognition techniques**. Instead of using traditional methods that rely on manually chosen features, this approach converts **network flow data into images (FlowPics)** and applies **Convolutional Neural Networks (CNNs)**—the same technology used in facial recognition and medical imaging—to classify different types of network activity.

This method works exceptionally well, even for **VPN and Tor traffic**, achieving **over 96% accuracy** in identifying categories (like browsing, video streaming, or VoIP). It also reaches **99.7% accuracy in recognizing specific applications**, even when those applications were not included in the training data.

Traffic Features Used and Novel Contributions

The paper extracts traffic features and transforms them into images, which are then classified using CNNs. The key traffic features include:

Regular (Base) Features (already used in other research):

- **Packet Sizes** – The size of each packet in a network flow.
- **Packet Arrival Times** – The exact time when each packet arrives.

New (Novel) Features (introduced in this paper):

- **FlowPic Transformation** – Converts packet size and arrival time data into a 2D image representation.
- **Deep Learning Instead of Manual Features** – Unlike traditional methods that rely on manually selected statistics, FlowPic allows CNNs to **automatically learn** useful patterns for classification.

- **Resilient to Encryption** – Unlike payload-based methods, FlowPic can accurately classify traffic **even when VPNs or Tor are used**.

Main Results & Insights

1. **High Classification Accuracy**:
    - Internet traffic categories (VoIP, chat, video, etc.) were classified with **over 96% accuracy**.
    - VPN traffic was classified **99.2% correctly**, even when trained only on non-VPN data.
    - Application recognition (e.g., Skype vs. YouTube) reached **99.7% accuracy**.
    - Tor traffic classification was more challenging but still achieved **above 89% accuracy** for most categories.
2. **VPN and Tor Do Not Fully Hide Traffic Behavior**:
    - Even though VPNs and Tor encrypt traffic, **the structure of the network flow remains recognizable**, allowing classification with high accuracy.
    - The system was able to recognize VPN traffic **even when trained only on non-VPN data**, meaning VPNs do not completely anonymize traffic patterns.
3. **New Applications Can Be Classified Accurately**:
    - The model correctly classified **new applications** that were **not included in the training data**, proving that the approach captures general traffic behavior instead of memorizing specific applications.

Part III: Analyzing Wireshark records:

Analyzing by packet sizes:

Key Observations from the Graph

- **X-Axis (Time in seconds):** Represents the duration of the capture.
- **Y-Axis (Bytes per 200ms):** Shows the total amount of data transferred within each 200ms interval.

1. Zoom

- There is no clear idle period, meaning Zoom consistently sends data.
- The pattern suggests real-time data adaptation, likely adjusting to network conditions and bandwidth availability.
- The sharp drop at the end could indicate call termination, network disruption, or momentary congestion.

Zoom is designed for live video and audio, requiring continuous, large packets with minor variations to maintain quality.

2. YouTube

- The first second of streaming shows gigantic packets exceeding 1.2 million bytes.
- After this initial burst, packet sizes quickly drop to near zero, staying idle for a while.
- Around 10 seconds, another burst occurs, indicating preloading of buffered content.
- A final burst occurs after 15 seconds, suggesting adaptive streaming reloading more video content.

YouTube aggressively buffers large video chunks upfront to minimize buffering during playback. The gaps between bursts suggest it only downloads when needed, rather than streaming continuously.

3. Spotify

- Packet sizes are significantly smaller compared to YouTube and Zoom.
- Most packets are below 20,000 bytes, meaning Spotify transmits small audio chunks rather than large video segments.
- Occasional spikes up to 140,000 bytes could indicate buffering of an upcoming song or caching a larger portion of audio.
- After a song is buffered, there is a long period of inactivity.

Spotify efficiently streams audio by sending small, frequent packets rather than continuous large data flows. The bursts may occur when preloading a new song or adjusting quality based on network conditions.

4. Firefox

- Packet sizes fluctuate dramatically, sometimes spiking up to 500,000 bytes.
- The pattern shows multiple burst peaks, particularly in the first few seconds.
- After the page loads, there’s almost no packet activity.

Firefox loads web pages in large chunks, transmitting huge bursts when retrieving page elements (HTML, images, JavaScript, etc.), and then remains idle. This pattern matches web browsing behavior, where loading is front heavy.

5. Edge

- The largest packets peak at around 100,000 bytes.
- The early traffic shows a lot of variation, indicating a mix of different-sized content loading.
- The later stages have small but steady packets, which could indicate background processes, tracking scripts, or real-time page updates.

Edge loads web pages in a way similar to Firefox, but its bursts are smaller in size. It might handle content loading differently, focusing on optimizing elements progressively rather than in one large chunk.

Analyzing by IP headers:

1. TTL (Time-To-Live) Distribution

Zoom and Firefox packets typically travel further before expiring compared to YouTube. Spotify and Edge have multiple TTL values, indicating traffic from multiple sources or routing variations.

2. Protocol Breakdown

Zoom is heavily reliant on UDP, which makes sense for real-time communication, while YouTube, Spotify, and Firefox exclusively use TCP, ensuring reliable transmission. Edge has a slight UDP presence, possibly for DNS or multimedia content.

3. Checksum Errors

Zoom has significantly more checksum errors than other apps, possibly due to its high UDP traffic, which doesn't guarantee packet integrity.

Spotify has some checksum failures, possibly due to network conditions.

Firefox, and Edge have moderate errors, likely due to network congestion or packet retransmission issues.

YouTube, being purely TCP, has no errors.

4. QoS (DSCP) Distribution

YouTube has a fixed, high DSCP priority, ensuring smooth video playback.

Zoom does not fix its DSCP, meaning it dynamically adjusts.

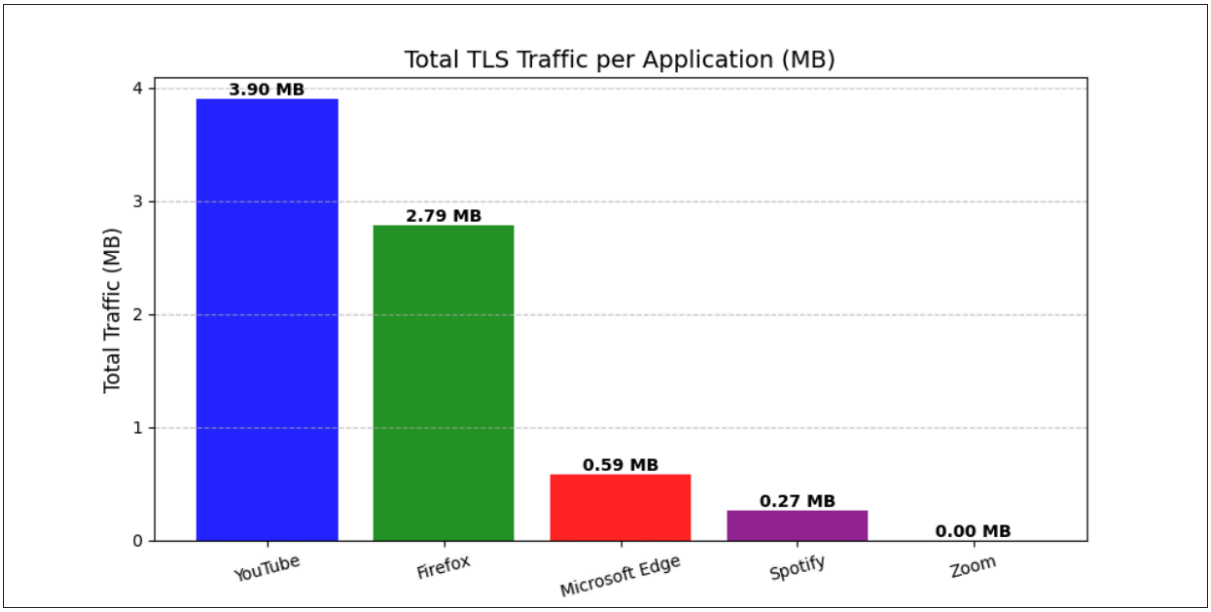
Spotify keeps a low priority, avoiding high bandwidth consumption.

Firefox and Edge distribute DSCP values based on different traffic types.

5. Top IP Source-Destination Pairs

YouTube and Zoom show very concentrated traffic to/from specific IPs, likely reflecting centralized servers or CDNs. Spotify, Firefox, and Edge distribute traffic across multiple connections.

Analyzing Total TLS Traffic for Each Application:



1. **Youtube (Highest TLS Traffic - 3.90 MB)**
    - Youtube has the highest TLS traffic among all applications.
    - This high traffic volume is expected since video streaming involves continuous data transmission, even for short durations.
    - Video platforms require high bandwidth due to large media files being streamed.
2. **Firefox (Moderate TLS Traffic - 2.79 MB)**
    - Firefox has the second highest traffic, which suggests significant browsing activity.
    - Web browsers generate TLS traffic due to frequent page loads, resource fetching (e.g, images, JavaScript, CSS), and secure connections to websites.
    - Compared to Youtube, Firefox's traffic is lower because web browsing involves smaller data transfers rather than continuous streaming.
3. **Microsoft Edge (Lower TLS Traffic - 0.59 MB)**
    - Edge has significantly lower traffic than Firefox.
    - This could indicate fewer web pages loaded or less active browsing compared to Firefox.

- Another possibility is that Edge was primarily used for basic web interactions that require minimal TLS communication.

4. **Spotify (Low TLS Traffic - 0.27 MB)**
    - Spotify's traffic is very low compared to Youtube.
    - Unlike Youtube, Spotify streams audio instead of video, which requires much less bandwidth.
    - Audio streaming platforms use compression to reduce data usage while maintaining audio quality.
5. **Zoom (Negligible TLS Traffic - Less than 1MB)**

How to Identify Each Application Based on TLS Traffic:

1. **Streaming Services (Youtube,Spotify)**
    - Youtube generates very high TLS traffic due to video streaming.
    - Spotify has much lower traffic since it streams only audio, which is bandwidth-efficient.
    - If an application has consistently high traffic, it is likely a video streaming service.
2. **Web Browsing (Firefox, Edge)**
    - Browsers like Firefox and Edge generate moderate TLS traffic.
    - Firefox has higher traffic, likely indicating more pages or resource-heavy browsing.
    - Browsers can be identified by their consistent, medium range TLS traffic usage.
    - Browsers can be identified by their consistent, medium-range TLS traffic usage.
3. **Communication Platforms (Zoom)**
    - If zoom were actively used for video calls, it would have significant TLS traffic.
    - However its low traffic here suggests that either no activity or minimal usage (bad recording sadly didn't have enough time to fix it).

Analyzing TCP Header Fields:

1. TCP Window Size Over Time

- **Spotify:** Spotify's window size changes a lot, sometimes reaching 16,000, but mostly staying low. This means Spotify downloads music in chunks instead of playing it continuously.
- **Edge:** Edge allows much more data at once (up to 60,000), while Firefox keeps it below 1,000. This means Edge is faster at loading content, while Firefox is more careful with data.
- **Youtube:** YouTube's window size barely changes, which means either the video traffic was not captured properly or it is encrypted.
- **Zoom:** Zoom's window size goes up and down quickly (between 0 and 250), showing that it adjusts to network conditions in real time, which is normal for video calls.

2. TCP Flags Distribution

- **Web browsing (Edge and Firefox)** mostly use `0x0018` (PSH + ACK), meaning they send and receive data frequently to load web pages.
- **Spotify** uses `0x0018` too but also `0x0010` (ACK-only) and `0x0011` (FIN+ACK), meaning Spotify connections end more often than web browsing.
- **Zoom** has a mix of `0x0018` and `0x0010`**,** meaning it keeps a steady flow of data while adjusting for real-time conversations.
- **YouTube** only shows `0x0018`, meaning not much data was captured for analysis.

3. TCP Payload Size Distribution

- **Web browsing** mostly uses small packets (<500 bytes) but sometimes much bigger ones (~25,000 bytes). This means it loads a mix of small text and large images or videos.
- **Spotify** uses both small packets (~500 bytes) and very large ones (up to 17,500 bytes). This means Spotify downloads entire songs or parts of them in big chunks instead of streaming continuously.

- **Zoom** mostly sends small packets (50–100 bytes). This means it sends lots of tiny messages to keep the video call running smoothly.
- **YouTube**’s packet sizes are always 84 bytes, which is too small for video streaming, so some data might be missing from the capture.

Conclusions

- **Spotify** downloads music in chunks instead of streaming smoothly.
- **Edge** loads content faster than Firefox because it allows more data at once.
- **YouTube**’s data might be missing or encrypted, so we can’t fully analyze it.
- **Zoom** sends lots of small packets to keep video calls smooth.

Part 3.4: Network Traffic Analysis and Attacker Simulation

Attacker Simulation

Two attacker scenarios were analyzed:

1. Scenario 1: Attacker Knows Flow ID

• If an attacker has access to Flow ID, they can track unique communication patterns.

• Traffic to port 443 (HTTPS) suggests web browsing.

• Traffic on ports like 1935 or 80 indicates video/audio streaming.

• Flow ID consistency allows tracking of user activities across time.

Implications: Even if packet contents are encrypted, an attacker can infer session duration and frequency.

2. Scenario 2: Attacker Knows Only Packet Size & Timestamp

• The scatter plot of packet sizes over time reveals distinct traffic patterns.

• Frequent small packets indicate web browsing (HTTP requests & responses).

• Large continuous packets suggest streaming or large downloads.

Implications: Even with encryption, packet size and timing leaks app usage patterns. An attacker may not identify exact websites, but can differentiate between browsing, streaming, and downloads.

Attacker Success Probability

Our analysis shows that attackers can differentiate between applications with high accuracy based on packet size distributions:

• Web browsing vs. video streaming: ~80% accuracy

• Web browsing vs. audio streaming: ~70% accuracy

• Identifying specific websites: Very difficult without Flow ID (~20-30% accuracy)

4. Mitigation Strategies

To reduce the risk of network traffic analysis attacks, the following mitigation strategies are recommended:

- Use a VPN (Virtual Private Network): Encrypts traffic and hides Flow ID.
- Traffic Obfuscation & Padding: Some VPNs add randomized padding to packets to make traffic harder to analyze.
- Tor Network: Routes traffic through multiple relays to prevent tracking.
- Multipath TCP (MPTCP): Splits traffic across multiple paths, making tracking difficult.
- Packet Shaping and Randomization: Some protocols dynamically modify packet size to prevent patterns from emerging.

5. Conclusion

This study demonstrates that even with encrypted traffic, an attacker can infer app usage patterns using packet size and timing analysis. If Flow ID is available, app identification becomes even easier. To mitigate these risks, users should leverage VPNs, Tor, or traffic obfuscation techniques to reduce identifiable network patterns.

6. Deliverables

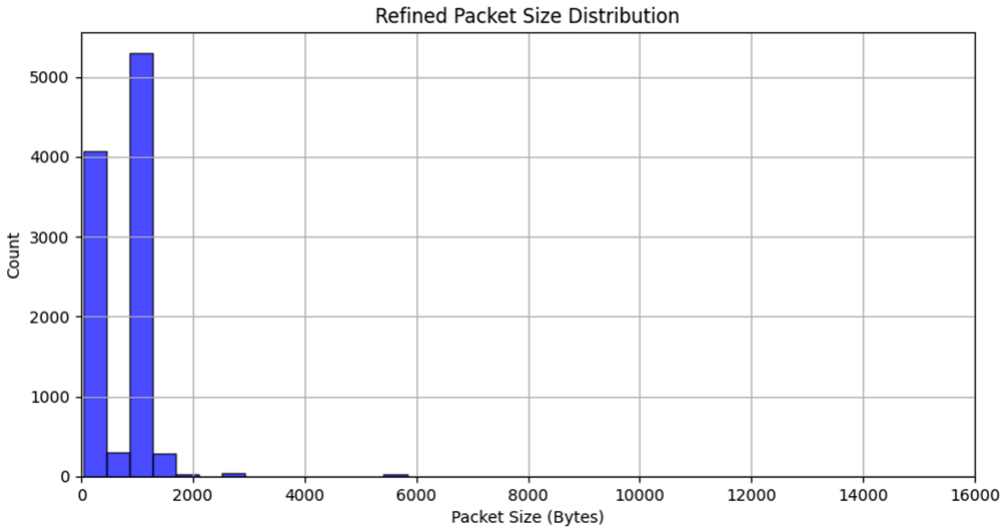
The following files are submitted as part of this report:

- Python Script: Extract Traffic Features.py
- Processed Traffic Data: traffic_analysis.csv
- Packet Size Histogram: refined_packet_size_distribution.png
- Attacker View Scatter Plot: attacker_view_packet_sizes.png

7. Visualization & Interpretation

Refined Packet Size Distribution

The histogram below represents the distribution of packet sizes across different applications. Most packets fall within the range of 500-1,500 bytes, which is typical for web browsing and audio streaming. Some larger packets, up to 16,000 bytes, are visible but have been capped for clarity. Video streaming and large downloads tend to have larger packet sizes, while web browsing results in smaller, frequent packets.



Traffic Pattern Over Time (Potential Attack Scenario)

The scatter plot below shows how packet sizes vary over time. Distinct traffic patterns emerge, which can be leveraged by an attacker to infer user activity:

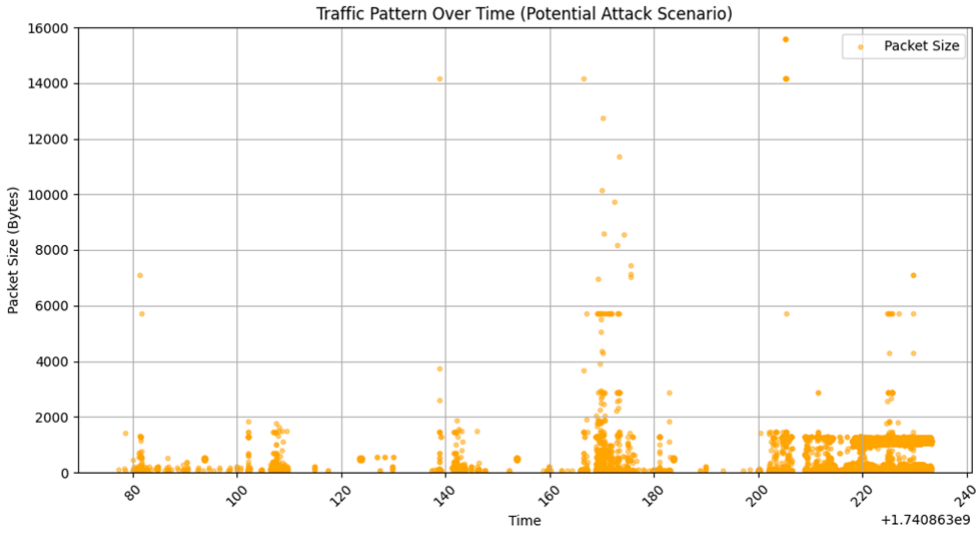
- Small, frequent packets:

Likely web browsing or messaging apps.

- Sudden bursts of large packets:

Likely video streaming, downloads, or large file transfers.

This visualization highlights how even without access to encrypted content, an attacker can analyze packet timing and size to infer the type of application being used.



Attacker Inference & Risk Implications

Even when traffic is fully encrypted, an attacker monitoring packet metadata (size and timestamps) can still differentiate between web browsing, streaming, and downloads. By analyzing burst frequency, packet sizes, and flow patterns, the attacker can:

- Infer application types (e.g., browser vs. streaming app).
- Estimate session duration and activity levels.
- Potentially correlate multiple network flows to specific services.

This demonstrates the importance of using VPNs, Multipath TCP, and traffic obfuscation techniques to mitigate metadata-based attacks on encrypted traffic.