Realistic Encrypted network traffic classification based on Large Dataset

Lior Davidyan, Rotem Kochavi, Tamar Revazishvili

October 2024

1 Introduction

In today's digital landscape, the rapid growth of encrypted network traffic poses significant challenges for network administrators, security professionals, and traffic management systems. With the widespread use of encryption protocols like TLS (Transport Layer Security) and SSL (Secure Sockets Layer), a large portion of internet traffic is now hidden from traditional inspection tools, making it difficult to analyze, monitor, and classify network activity accurately. This has critical implications for tasks such as identifying network threats, ensuring compliance with security policies, optimizing bandwidth usage, and improving Quality of Service (QoS) for users. Traditional methods of traffic analysis, such as Deep Packet Inspection (DPI), are becoming increasingly obsolete, as they rely on being able to inspect the contents of data packets. With encryption, these contents are no longer visible, leaving only metadata like packet timing, size, and flow patterns. As a result, network administrators and security systems have limited insight into the types of applications or services being used, leading to potential security blind spots, inefficiencies in traffic management, and an inability to enforce policies effectively. To address this growing issue, this project focuses on encrypted network traffic classification based on a large, custom-built dataset. The project has two primary components: data generation and traffic classification.

2 Problem Statement

The increase in encrypted traffic has significantly complicated the task of identifying the type of application or service generating the traffic. Encryption methods like Transport Layer Security (TLS) obscure the payload data, which previously provided valuable insights for network analysis. As a result, network administrators and security analysts have limited visibility into the nature of traffic traversing their networks. This lack of visibility raises concerns, particularly in the areas of network security, traffic management, and anomaly detection, where accurate identification of network flows is critical. Furthermore, the existing datasets used for training machine learning models for traffic classification are often limited in their scope and diversity. Many of these datasets either do not contain sufficient variations in network conditions or lack comprehensive coverage of modern web applications and services. Consequently, machine learning models trained on these datasets may fail to generalize to real-world traffic, especially in the case of encrypted flows. This project seeks to address these challenges by creating a large, diverse, and highly representative dataset of network traffic, specifically focusing on encrypted flows and various network behaviors.

3 Related Work

The classification of encrypted network traffic has garnered significant attention due to the increasing use of encryption protocols, such as TLS and SSL, which limit traditional methods like Deep Packet Inspection (DPI). Several methodologies have emerged that aim to address the challenges posed by encrypted traffic, primarily focusing on feature extraction, machine learning, and deep learning techniques to classify traffic without directly analyzing the content of encrypted packets. This section provides an overview of the most relevant studies and highlights how our proposed work builds on and differs from these efforts.

3.1 Machine Learning and Deep Learning-Based Approaches

Machine learning has been a primary tool for encrypted traffic classification, with many studies focusing on statistical and flow-based features. [6] explored machine learning approaches for encrypted traffic classification using a variety of features, such as packet size, inter-arrival times, and flow duration. Their approach demonstrated that these non-payload features could achieve reasonable accuracy, but their dataset lacked the variability and diversity needed to generalize to more complex, real-world scenarios. Deep learning models have also gained traction, with [2] proposing ECNet, a model for robust traffic detection using multi-view features. This model combined both content-based and pattern-based features to classify encrypted malicious traffic, achieving improved detection rates compared to traditional methods. However, these works typically focus on malicious traffic rather than the wide variety of encrypted application traffic. Our work extends these approaches by focusing not only on malicious traffic but on a broader range of encrypted traffic, including web browsing, video streaming, downloading and uploading, etc. Furthermore, we emphasize dynamic data generation using tools like Clumsy to introduce real-world conditions such as packet loss and delay, thereby creating a more diverse and comprehensive dataset than previously available.

3.2 Out-of-Distribution Detection in Encrypted Traffic

The problem of Out-of-Distribution (OOD) detection is increasingly important for handling unknown traffic types. [3] developed a method to identify OOD samples using LSTM and PCA for encrypted mobile traffic classification. They achieved significant success in identifying traffic outside of the known training distribution, which is crucial for ensuring the robustness of classification systems in real-world environments.

3.3 Frameworks and Tools for Traffic Classification

Several frameworks have been developed to standardize and facilitate research in encrypted traffic classification. The OSF-EIMTC framework introduced by [1] provides an end-to-end solution for machine learning-based traffic classification, including dataset management, feature extraction, model training, and evaluation. This framework enables researchers to compare models and features consistently, promoting reproducibility and collaborative advancements in the field. While frameworks like OSF-EIMTC focus on standardization, our work emphasizes dynamic traffic generation and labeling. Instead of relying solely on pre-existing datasets, we will create a new, dynamically generated dataset using a custom crawler and network emulator. This approach ensures that our dataset captures a broader range of traffic types and conditions, making it more representative of real-world network environments.

3.4 Few-Shot Learning and Adaptability to New Traffic

Another challenge in encrypted traffic classification is the ability to adapt to new traffic types with limited training data. [8] proposed a few-shot learning model based on Siamese Prototypical Networks (SPN) for traffic classification. Their work highlighted the importance of learning from a small number of labeled samples, particularly for rare or unseen traffic types. Our project differs by focusing on comprehensive data generation to minimize the need for few-shot learning. By dynamically generating a wide variety of traffic types using a custom-built crawler and network emulator, we aim to create a dataset that is large enough to train models without the need for specialized few-shot techniques.

4 Crawler

As part of our project, we developed a custom WebCrawler using Python to automate web interactions, capture network traffic, and perform tasks such as browsing, downloading files, streaming videos, uploading files, and participating in video meetings. Our crawler leverages Selenium WebDriver for automating interactions and Scapy for network traffic capture. The goal is to simulate real-world web behavior and collect traffic data, which is then structured and stored for further analysis.

4.1 Crawler Functionality

Our WebCrawler automates a wide range of web interactions, from downloading files and uploading files to video streaming and joining online meetings. While executing these tasks, it captures network traffic in real-time, allowing us to analyze and organize the resulting data for classification and further use in research.

4.2 Key Features

4.2.1 Automated Web Interaction

The crawler mimics user behavior on various websites using Selenium, automating actions such as navigating URLs, playing videos on YouTube and other platforms, downloading files, uploading files, and joining Google Meet sessions. It can also manage common web elements like cookie consent pop-ups and play buttons, ensuring smooth automation of interactions.

4.2.2 Network Traffic Capture and Simulation

We utilize Scapy and AsyncSniffer to capture detailed packet-level traffic during each interaction. Additionally, we use Clumsy to simulate real-world network conditions such as packet loss, delays, and bandwidth throttling. This allows us to evaluate how websites behave under different network stresses. The captured traffic is saved in PCAP files, enabling us to store and analyze the network data generated by our crawler, including traffic patterns under simulated disruptions.

4.2.3 File Upload Automation

Our crawler is capable of automating file uploads to designated websites. By randomly selecting a file from a specified directory, the crawler can simulate a user uploading a file to services such as file.io or gofile.io, capturing the traffic generated during this process.

4.2.4 Session Management

Each browsing, download, or video session is handled independently, ensuring that traffic data is stored separately for every interaction. This structure allows us to easily associate the traffic data with specific web activities, such as streaming a video or uploading a file.

4.2.5 Video Interaction

Our crawler is designed to interact with various video platforms (e.g., YouTube, CNN, Vimeo), where it automatically plays videos by detecting and clicking play buttons. It monitors whether the video plays successfully and captures the network traffic during streaming.

4.2.6 Google Meet Integration

We've integrated the crawler with Google Meet using the Google Calendar API, allowing it to create and join meetings autonomously. Once in the meeting, the crawler can perform actions like turning off the camera or microphone, and even sending messages in the chat.

4.3 Traffic Capture and PCAP Generation

The network traffic generated during each web interaction is captured and saved in PCAP format. This data includes packet-level details such as source and destination IP addresses, protocol types, and packet timing information. By using Clumsy, we can simulate various network conditions (e.g., delay, packet loss, throttling), further enriching the dataset by providing traffic patterns that reflect network disruptions. We organize the traffic data in a structured format, ensuring that each session's data is labeled with metadata such as the URL, timestamp, and network conditions applied during the session.

4.4 Dynamic Web Interaction

Our WebCrawler is flexible enough to interact with a wide variety of websites, which allows us to generate traffic for different scenarios. By simply updating the list of URLs, we can collect traffic data from:
- Web Browsing: Visiting and interacting with sites like news portals, social media, and search engines.
- Video Streaming: Playing videos on platforms like YouTube, CNN, and Vimeo and much more.
- File Downloads: Detecting file links and automating the download process.
- File Uploads: Uploading files to services such as file.io and capturing the traffic generated during this process.
- Google Meet: Creating and joining video meetings, interacting with the meeting participants, and capturing the traffic data generated during the session.

4.5 PCAP File Organization and Metadata

For each interaction, we store the network traffic in PCAP files, along with useful metadata such as:
- Application Type: The type of interaction (e.g., video streaming, file uploading, file downloading, web browsing).
- Timestamp: The time and date of the interaction, which allows for temporal analysis.
- Network Conditions: We apply specific network conditions like delay or packet loss using Clumsy tool to simulate real-world scenarios and observe the effects on traffic behavior.

This structured storage approach helps us efficiently organize and analyze the captured traffic data, making it easier to label and process for further research.

4.6 Customization and Scalability

We've designed the WebCrawler to be highly customizable and scalable, allowing us to handle a variety of web interactions by adjusting the URLs and operations. This flexibility is critical for adapting the crawler to different sectors, such as media streaming, e-commerce, or cloud storage.

4.7 Conclusion

Our WebCrawler serves as a crucial component of our project, providing a scalable solution for automating web interactions and capturing network traffic. By leveraging Selenium WebDriver for automation and Scapy for network traffic capture, we've built a tool that is capable of dynamically interacting with a wide range of web applications. The resulting traffic is stored in a well-organized structure, enabling easy analysis and classification. This crawler is an essential tool in our efforts to collect diverse datasets, which we can use for further research and development.

5 Dataset

The dataset created for this project is a comprehensive collection of network traffic captures (PCAP files) aimed at facilitating the classification of encrypted network traffic. The dataset consists of traffic generated from a wide variety of applications and web services, captured during automated browsing sessions, file transfers, and media streaming. This dataset is designed to provide a broad and diverse representation of modern network traffic, particularly focusing on encrypted flows, which pose significant challenges for traditional traffic analysis methods.

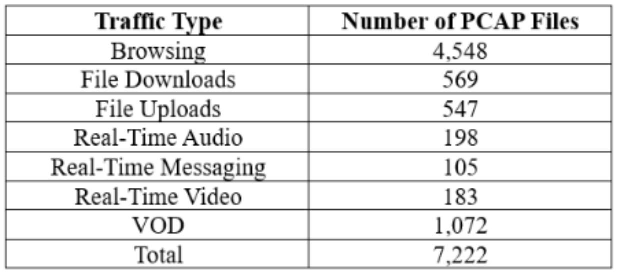
5.1 Traffic Categories

The dataset covers a wide spectrum of traffic types, with each category representing a specific type of interaction or application. The breakdown of the dataset is as follows:
- Browsing: The dataset includes traffic captured while browsing various websites. This category represents the largest portion of the dataset, with 4,548 PCAP files generated from browsing sessions across different websites, ranging from news outlets to e-commerce platforms. Each browsing session simulates typical user behavior, such as loading pages, scrolling, and interacting with content.
- File Downloads: Traffic generated by downloading files from various sources. This includes interactions with government and educational websites, where large datasets or documents are frequently downloaded. A total of 569 PCAP files were captured in this category.
- File Uploads: Similar to file downloads, this category includes traffic generated from uploading files to cloud services like file.io and gofile.io, resulting in 547

PCAP files.
- Real-Time Audio, Messaging and Video: Traffic generated from real-time communication platforms, including voice calls and instant messaging services. The dataset contains 198 PCAP files for real-time audio, 105 PCAP files for real-time messaging, and 183 PCAP for real-time video.
- Video on Demand (VOD): The dataset also includes extensive traffic from streaming platforms, capturing Video on Demand interactions with platforms such as YouTube, BBC, Vimeo, etc. This category includes 1,072 PCAP files.

5.2 Dataset Size and Structure

In total, the dataset consists of 7,222 PCAP files, each representing a distinct session of network traffic. The files are organized into subcategories based on the type of interaction, as shown in the table below:



Traffic Type	Number of PCAP Files
Browsing	4,548
File Downloads	569
File Uploads	547
Real-Time Audio	198
Real-Time Messaging	105
Real-Time Video	183
VOD	1,072
Total	7,222

Figure 1: Types of interactions

The dataset is structured to ensure easy retrieval and processing, with files labeled based on the interaction type and the application or website involved. Each file is stored with metadata that includes the application type, the network conditions, and the timestamp of the capture, providing valuable context for traffic analysis.

5.3 Diversity of Traffic Sources

The dataset spans 338 different applications and websites, ensuring a diverse representation of traffic types. This diversity is essential for developing machine learning models that can generalize well to real-world encrypted traffic scenarios. Below are examples of traffic sources across different categories:
- Browsing: Websites like bbc.com, cnn.com, arstechnica.com, and github.com are among the wide variety of sites included. Domains represent various sectors, including news, technology, and social media.
- File Downloads: Traffic from government and educational websites such as amazonaws.com, census.gov, and govt.nz, providing a range of data related to file transfers from institutional and public sources.
- Real-Time Communication: Real-time messaging, video, and chat traffic is

captured from Google Meet, offering insights into encrypted communication flows during voice and video calls, as well as text-based interactions on this popular platform.
- VOD: Streaming platforms like YouTube, Dailymotion, Vimeo, etc., representing modern media consumption behaviors, particularly focusing on video streaming patterns.

The diverse set of applications covered in the dataset ensures that it includes a variety of network behaviors, including web browsing, file transfers, real-time communication, and media streaming, making it highly representative of modern encrypted traffic.

5.4 Labeling and Metadata

Each session captured in the dataset is carefully labeled with essential metadata to facilitate accurate analysis and training of machine learning models. The metadata includes:
- Application/Website Name: The specific website or service that generated the traffic (e.g., cnn.com, github.com).
- Interaction Type: The type of interaction (e.g., browsing, file download, file upload, real-time video).
- Timestamp: The time at which the session was captured.
- PCAP File Size: The size of each capture, indicating the volume of traffic generated.

This structured approach to data collection ensures that each traffic capture is organized and easily accessible for analysis. The labeling process enhances the dataset's utility for supervised learning tasks, where accurate labeling is critical for model performance.

5.5 Applications and Use Cases

The dataset is designed to be highly versatile, with potential applications across a range of network analysis and security tasks:
- Encrypted Traffic Classification: The primary goal of this dataset is to facilitate the classification of encrypted traffic using machine learning models. The diversity of traffic types, coupled with the metadata and labeling, makes this dataset an excellent resource for training models that can accurately classify traffic.
- Anomaly Detection: The dataset's inclusion of various interaction types and network behaviors enables its use in anomaly detection tasks, where identifying deviations from normal traffic patterns is crucial for network security.
- Performance Optimization and Quality of Service (QoS): By analyzing the different types of traffic in the dataset, network administrators can better understand how different applications behave under varying conditions, helping them optimize bandwidth usage and improve QoS.

5.6 Conclusion

The dataset generated for this project is a comprehensive, well-structured collection of encrypted network traffic, consisting of 7,222 PCAP files across a wide range of applications and interaction types. This dataset is designed to address the challenges posed by encrypted traffic classification, providing a rich resource for training machine learning models capable of accurately classifying encrypted traffic. The diversity of traffic types and applications, coupled with thorough labeling and metadata, ensures that the dataset is both flexible and highly valuable for research in network security, traffic management, and performance optimization.

6 Datasets Comparison

In this comparison, we focus on the normal (non-malicious) traffic within several widely-used datasets for network traffic analysis. The datasets under consideration—ISCX 2016, USTC-TFC2016, CTU-13, and UNIBS Traces—offer a rich variety of labeled traffic types, including encrypted communication, file transfers, and video streaming. While these datasets may contain both malicious and benign traffic, our analysis is limited to their normal traffic components, ensuring a focus on typical, real-world usage. Additionally, Our Data provides a modern and extensive dataset, incorporating various traffic manipulations such as delays, packet loss, and bandwidth throttling. This enables a detailed study of how network traffic behaves under real-world conditions with manipulated traffic. With a larger number of labeled applications, it serves as a comprehensive tool for studying today's complex network environments.

6.1 ISCX-2016

The ISCX dataset, available from [18], includes 7 traffic classes, such as HTTPS, SMTP, and FTP. It was captured in 2016 and is one of the widely used datasets for encrypted traffic analysis. The dataset is labeled and available in full PCAP format. Works that used this data could be found in [9,10].

6.2 USTC-TFC2016

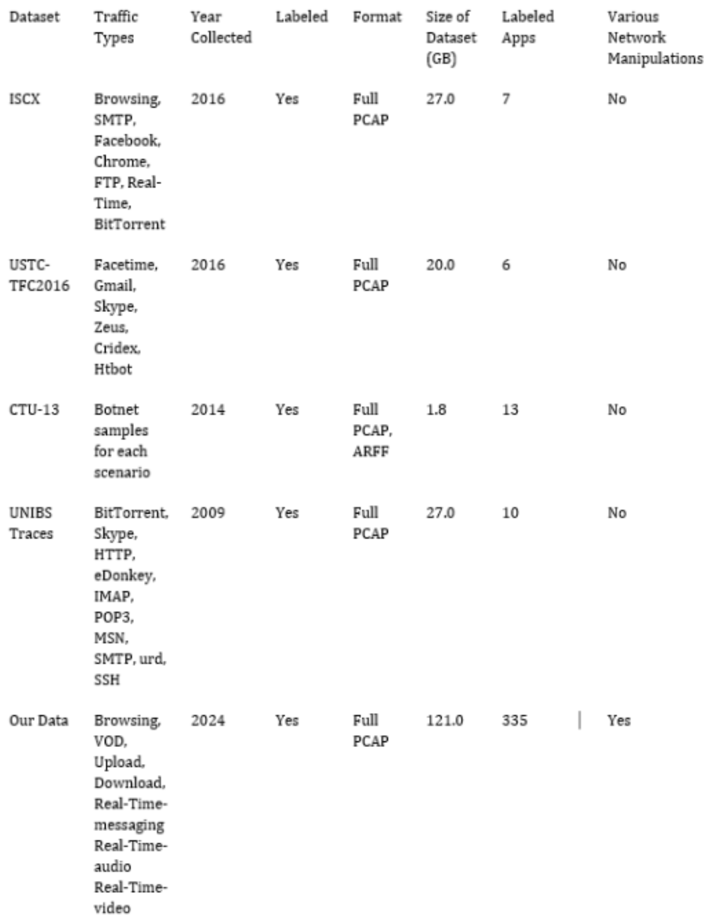
The USTC-TFC2016 dataset , available from [15], includes two major traffic classes, comprising both benign and malicious traffic. Collected in 2016, it captures common traffic types like Skype and Gmail. This dataset is widely used for traffic classification and network security research. Works that used this data could be found in [11,12].

6.3 CTU-13

The CTU-13 dataset, available in both Full PCAP and ARFF format from [16], consists of 13 botnet scenarios and samples, with labeled data for traffic type identification. Each botnet scenario offers different samples to assess real-world traffic conditions. Works that used this data could be found in [13].

6.4 UNIBS traces

The UNIBS traces dataset available from [17] captures 10 classes of traffic, including popular communication protocols such as BitTorrent, Skype, HTTP, and IMAP. This dataset was collected in 2009 and is fully labeled for easy classification of traffic flows. Works that used this data could be found in [14].



Dataset	Traffic Types	Year Collected	Labeled	Format	Size of Dataset (GB)	Labeled Apps	Various Network Manipulations
ISCX	Browsing, SMTP, Facebook, Chrome, FTP, Real-Time, BitTorrent	2016	Yes	Full PCAP	27.0	7	No
USTC-TFC2016	Facetime, Gmail, Skype, Zeus, Cridex, Htbot	2016	Yes	Full PCAP	20.0	6	No
CTU-13	Botnet samples for each scenario	2014	Yes	Full PCAP, ARFF	1.8	13	No
UNIBS Traces	BitTorrent, Skype, HTTP, eDonkey, IMAP, POP3, MSN, SMTP, urd, SSH	2009	Yes	Full PCAP	27.0	10	No
Our Data	Browsing, VOD, Upload, Download, Real-Time-messaging Real-Time-audio Real-Time-video	2024	Yes	Full PCAP	121.0	335	Yes

Figure 2: Dataset comparison with Labeled Apps

7 Feature Exploration

Feature exploration in encrypted traffic classification is critical for accurately identifying different types of traffic, even when the content is hidden due to encryption. Researchers rely on a variety of statistical, temporal, and protocol-specific features to help machine learning models differentiate between traffic types.
Statistical features are commonly used to capture basic traffic characteristics, such as packet size and flow byte count. These features provide insights into the traffic type—larger packet sizes often suggest video streaming, while smaller packets may indicate web browsing or messaging [1]. Similarly, the total number of bytes transferred during a flow can help distinguish between lightweight and heavyweight traffic types.
Temporal features focus on the timing of packets within a flow. Inter-arrival time (IAT), which measures the time between packets, is particularly useful for detecting real-time applications, which have short IATs, while longer IATs suggest non-real-time activities like streaming [8]. Flow duration is another key feature, with longer flows typically indicating services like streaming or downloads, and shorter ones suggesting browsing or quick interactions [3].
Protocol-specific features leverage metadata from protocols like TLS, DNS, and HTTP to provide additional context about the traffic source. For example, the Service Name Indication (SNI) in a TLS handshake can reveal the destination, even if the payload is encrypted [2]. These features complement the statistical and temporal ones, enhancing the accuracy of classification models.
Flow-level features aggregate packet-level data over the entire flow, capturing the overall behavior of a session. By summarizing features like average packet size and packet count, flow-level analysis offers a more comprehensive view, especially for long-lasting connections such as those in streaming services [5].

In recent approaches, deep learning models are increasingly used to automatically learn features from raw traffic. However, manually crafted features, like those mentioned above, remain crucial, especially in hybrid models that combine traditional and deep learning methods [1]. This combination allows models to take advantage of both hand-engineered insights and the representational power of deep learning.
In summary, statistical, temporal, and protocol-specific features form the backbone of encrypted traffic classification. These features, alongside advanced techniques like flow-level aggregation and hybrid models, continue to enhance the performance of machine learning models in classifying encrypted traffic accurately.

7.1 Features Used In The Project

The features used in our project for encrypted traffic classification are based on various packet-level and flow-level characteristics. These features were designed to capture essential aspects of the traffic while avoiding reliance on payload inspection, which is often impossible due to encryption. Below is a detailed exploration of the features used in the project, aligning with the techniques and features outlined in [1].

1. Packet Size Features Packet size is a fundamental metric that provides insight into traffic behavior without requiring access to encrypted content. The features we extracted from the packet capture (PCAP) files include:
- Minimum Packet Size: The smallest packet observed in the flow. This can help distinguish between different types of applications, as for example, file transfers tend to have larger packets compared to web browsing.
- Maximum Packet Size: The largest packet size in the flow, which is often larger for video streaming and file transfers.
- Mean Packet Size: The average size of packets in the flow, which is useful for understanding the general nature of the traffic.
- Small Packet Ratio: The proportion of packets below a certain size threshold (100 bytes in our case). This feature is particularly useful for identifying flows with a high degree of interactivity, such as web browsing, where small packets are more common.

These packet size features provide a window into traffic characteristics without needing to inspect the payload. Packet size distribution, as indicated in [1], can significantly improve classification accuracy by highlighting traffic patterns specific to various applications.

2. TLS Record Features TLS features focus on the metadata available from the TLS handshake and the size of TLS records:
- Minimum, Maximum, and Mean TLS Record Sizes: These features provide information about the encrypted data transmitted during the session. They can be used to differentiate between short-lived interactive traffic (e.g., web browsing) and long-running data transfers (e.g., video streaming).
- Outgoing and Incoming TLS Record Count: The number of TLS records sent from the client to the server (outgoing) and from the server to the client (incoming). These features give insight into the communication direction and flow, which can be useful for detecting client-initiated (outgoing) and server-initiated (incoming) traffic.

TLS features are critical for understanding encrypted traffic, as they allow us to classify the traffic type without needing to decrypt it.

3. Packet-Level Features In addition to packet size, we also capture more nuanced features from individual packets:
- Inter-Arrival Time (min, max, mean): The time between consecutive packets in the flow. This feature is crucial for identifying real-time applications where low inter-arrival times are expected.
- Packet Direction (outgoing, incoming): Whether a packet is outgoing or incoming, based on source and destination ports. This feature helps distinguish

between client-side and server-side communication patterns.
- TCP Window Size (min, max, mean): The size of the TCP window, which is an indicator of the flow's capacity to transmit data. Larger window sizes often correspond to bulk transfers (e.g., file downloads or streaming), while smaller sizes indicate interactive sessions.

These packet-level features provide a deeper understanding of the flow's structure, helping to classify encrypted traffic based on the behavior of the packets involved.

4. Clump Features (Subflows) Clump features aggregate packets into subflows based on inter-arrival times:
- Clump Length: The number of packets in a subflow. Applications like file transfers often have long clumps, while more interactive applications like web browsing have shorter clumps.
- Clump Size: The total size of all packets in a subflow.
- Clump Inter-Arrival Time: The time between the start of consecutive clumps. Shorter inter-arrival times often suggest more continuous traffic, as seen in streaming or large file transfers.

Clumping helps reduce the dimensionality of the dataset and enhances classification accuracy by focusing on high-level traffic patterns, rather than treating every packet individually.

5. DNS and IP Features Domain Name System (DNS) queries provide metadata about the domain names being accessed:
- Number of IP Addresses in Response: The number of IP addresses returned in a DNS query, which can indicate whether a session involves load-balancing or multiple servers.
- TTL (Time-To-Live) Values: These provide insight into the lifetime of DNS responses, which can help identify dynamic vs. static services.
- Domain Name Characteristics: The length of the domain name and the presence of special characters (e.g., hyphens, digits). These text-based statistics are helpful in identifying certain types of services (e.g., content delivery networks or dynamic DNS services).

DNS features are essential for traffic classification when dealing with encrypted flows, as they provide metadata about the services involved, even when the payload is hidden.

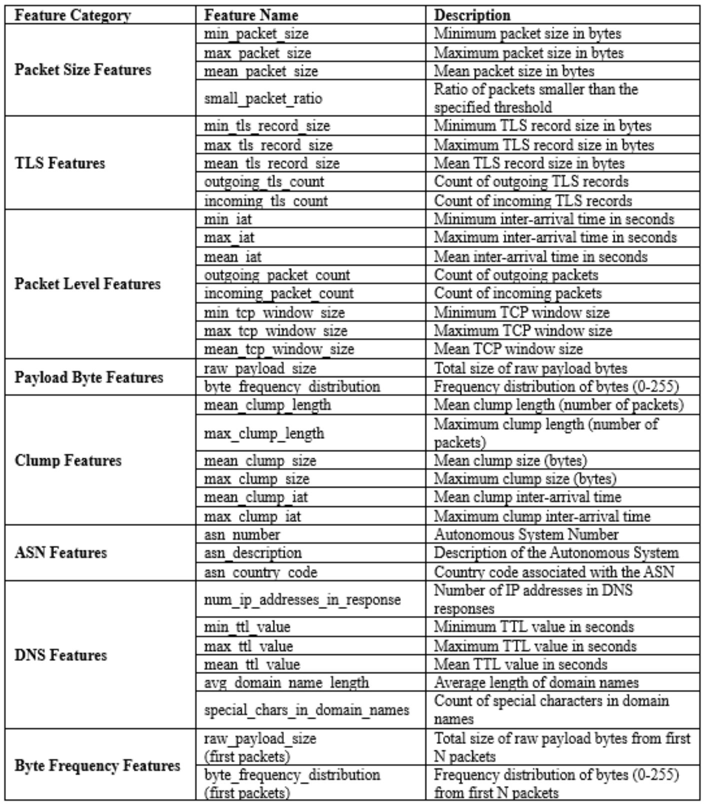
6. ASN Features Autonomous System Numbers (ASN) are used to identify the network provider or organization responsible for the traffic's origin or destination:
- ASN Number: The unique number identifying the network.
- ASN Description: A description of the organization or network responsible for the IP address.

- ASN Country Code: The country where the ASN is registered. This feature is useful for geolocation and identifying international traffic.

ASN features are especially helpful in classifying encrypted traffic based on the network characteristics, allowing for the identification of traffic originating from specific countries or organizations.

7. Payload Byte Features We also extract raw payload byte information:
- Raw Payload Size: The size of the raw payload in the flow.
- Byte Frequency Distribution: The frequency of each byte value (0–255) in the payload, normalized to represent the proportion of each byte type. This can be used to identify patterns in the encrypted traffic flow that might correspond to specific applications.

While the payload itself may be encrypted, the distribution of byte values can provide useful signals for traffic classification.



Feature Category	**Feature Name**	**Description**
Packet Size Features	min_packet_size	Minimum packet size in bytes
	max_packet_size	Maximum packet size in bytes
	mean_packet_size	Mean packet size in bytes
	small_packet_ratio	Ratio of packets smaller than the specified threshold
TLS Features	min_tls_record_size	Minimum TLS record size in bytes
	max_tls_record_size	Maximum TLS record size in bytes
	mean_tls_record_size	Mean TLS record size in bytes
	outgoing_tls_count	Count of outgoing TLS records
	incoming_tls_count	Count of incoming TLS records
Packet Level Features	min_iat	Minimum inter-arrival time in seconds
	max_iat	Maximum inter-arrival time in seconds
	mean_iat	Mean inter-arrival time in seconds
	outgoing_packet_count	Count of outgoing packets
	incoming_packet_count	Count of incoming packets
	min_tcp_window_size	Minimum TCP window size
	max_tcp_window_size	Maximum TCP window size
	mean_tcp_window_size	Mean TCP window size
Payload Byte Features	raw_payload_size	Total size of raw payload bytes
	byte_frequency_distribution	Frequency distribution of bytes (0-255)
Clump Features	mean_clump_length	Mean clump length (number of packets)
	max_clump_length	Maximum clump length (number of packets)
	mean_clump_size	Mean clump size (bytes)
	max_clump_size	Maximum clump size (bytes)
	mean_clump_iat	Mean clump inter-arrival time
	max_clump_iat	Maximum clump inter-arrival time
ASN Features	asn_number	Autonomous System Number
	asn_description	Description of the Autonomous System
	asn_country_code	Country code associated with the ASN
DNS Features	num_ip_addresses_in_response	Number of IP addresses in DNS responses
	min_ttl_value	Minimum TTL value in seconds
	max_ttl_value	Maximum TTL value in seconds
	mean_ttl_value	Mean TTL value in seconds
	avg_domain_name_length	Average length of domain names
	special_chars_in_domain_names	Count of special characters in domain names
Byte Frequency Features	raw_payload_size (first packets)	Total size of raw payload bytes from first N packets
	byte_frequency_distribution (first packets)	Frequency distribution of bytes (0-255) from first N packets

Figure 3: Features Used In The Project

To summarize, these features align closely with the ones described in OSF-EIMTC by [1]. Our project extracts and leverages similar flow and packet-level features, including packet size, TLS record data, clump (subflow) statistics, and DNS/ASN metadata. These feature sets are essential for classifying encrypted traffic accurately without relying on payload inspection, which is critical in modern encrypted traffic environments.

8 Classification Methods

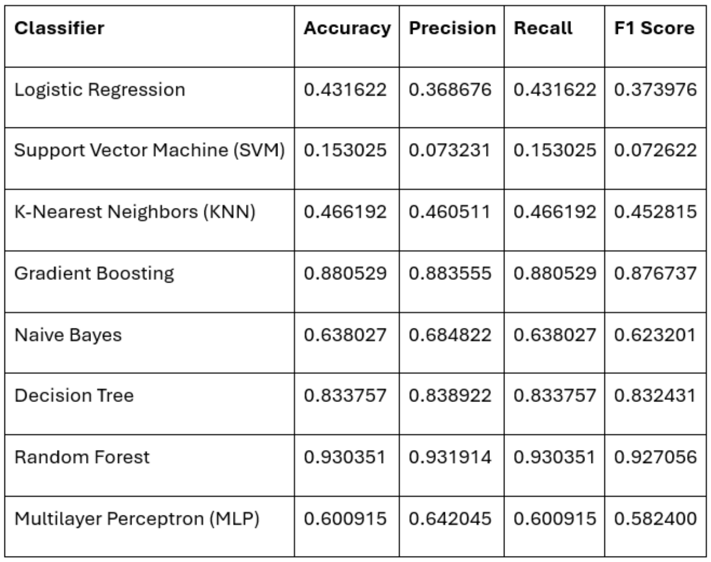
To classify encrypted network traffic based on the features extracted from the dataset, we utilized several well-known machine learning algorithms. These algorithms were chosen for their diverse approaches to classification, ranging from linear models to complex ensemble methods. The models tested are as follows: Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Gradient Boosting, Naive Bayes, Decision Tree, Random Forest, Multilayer Perceptron (MLP).
Each classifier was trained on the feature set and evaluated using common performance metrics: accuracy, precision, recall, and F1 score. To handle missing values in the dataset, we used a mean imputation strategy, filling in the gaps before splitting the data into training and test sets. We employed a standard 70-30 train-test split for the evaluation of all models.

8.1 Results

The performance of the classifiers was evaluated on two different labels: 'Application' and 'Attribution'. Below are the results, with four key metrics: accuracy, precision, recall, and F1 score. Each classifier's performance is compared for both labels.

8.1.1 Results for 'Application' Label

For the 'Application' label, Random Forest had the best performance, achieving the highest accuracy (93.0%) and F1 score (92.7%). Gradient Boosting was the second-best model, with an accuracy of 88.1% and an F1 score of 87.7%. Decision Tree also performed well with an accuracy of 83.4%, making it a strong model for application classification. K-Nearest Neighbors (KNN) provided moderate accuracy, while Logistic Regression and Support Vector Machine (SVM) underperformed, particularly the SVM, which struggled with very low accuracy (15.3%).



Classifier	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.431622	0.368676	0.431622	0.373976
Support Vector Machine (SVM)	0.153025	0.073231	0.153025	0.072622
K-Nearest Neighbors (KNN)	0.466192	0.460511	0.466192	0.452815
Gradient Boosting	0.880529	0.883555	0.880529	0.876737
Naive Bayes	0.638027	0.684822	0.638027	0.623201
Decision Tree	0.833757	0.838922	0.833757	0.832431
Random Forest	0.930351	0.931914	0.930351	0.927056
Multilayer Perceptron (MLP)	0.600915	0.642045	0.600915	0.582400

Figure 4: Classification Results for 'Application' Label

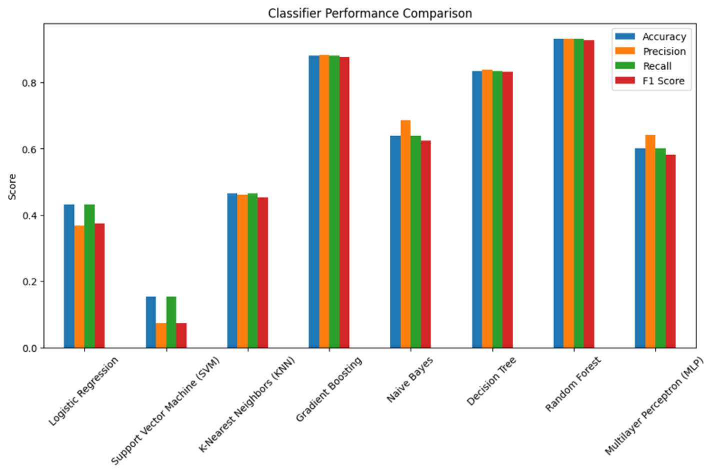
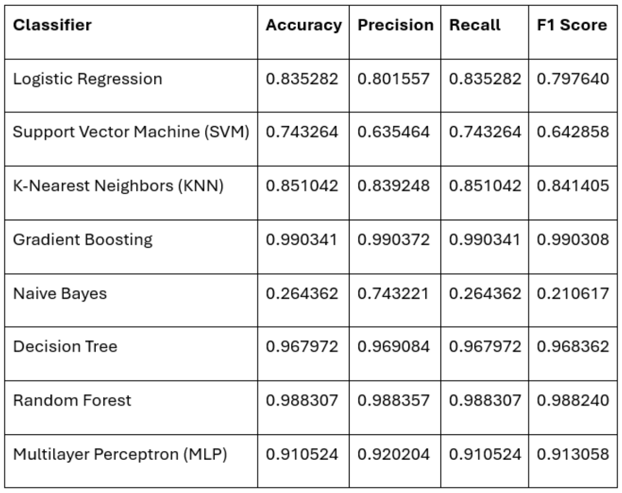


Figure 5: 'Application' Label Performance Comparison

8.1.2 Results for 'Attribution' Label

For the 'Attribution' label, Gradient Boosting achieved the highest performance, with an accuracy of 99.0% and an F1 score of 99.0%. Random Forest followed closely with an accuracy of 98.8% and an F1 score of 98.8%. Decision Tree also performed extremely well with 96.8% accuracy. K-Nearest Neighbors (KNN) and Multilayer Perceptron (MLP) provided good accuracy, above 85% and 91%, respectively. Naive Bayes, however, performed poorly, with an accuracy of only 26.4%, likely due to its simple nature, which is not well-suited for this complex dataset.



Classifier	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Logistic Regression	0.835282	0.801557	0.835282	0.797640
Support Vector Machine (SVM)	0.743264	0.635464	0.743264	0.642858
K-Nearest Neighbors (KNN)	0.851042	0.839248	0.851042	0.841405
Gradient Boosting	0.990341	0.990372	0.990341	0.990308
Naive Bayes	0.264362	0.743221	0.264362	0.210617
Decision Tree	0.967972	0.969084	0.967972	0.968362
Random Forest	0.988307	0.988357	0.988307	0.988240
Multilayer Perceptron (MLP)	0.910524	0.920204	0.910524	0.913058

Figure 6: Classification Results for 'Attribution' Label

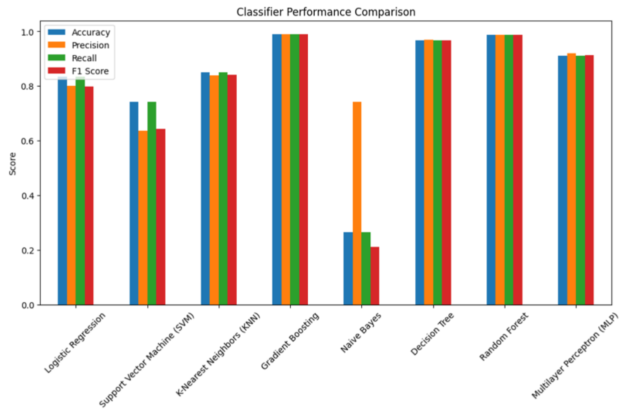


Figure 7: 'Attribution' Label Performance Comparison

8.1.3 Overall Trends

- Random Forest and Gradient Boosting consistently performed the best for both labels, with Gradient Boosting slightly outperforming for the 'Attribution' label.
- Decision Tree provided strong results, making it a reliable choice for both 'Application' and 'Attribution' classification.
- Multilayer Perceptron (MLP) and K-Nearest Neighbors (KNN) performed moderately well but could not match the accuracy of the ensemble methods.
- Naive Bayes struggled particularly for the 'Attribution' label, indicating that more complex models are necessary for such tasks.
- Support Vector Machine (SVM) underperformed in both cases, especially for 'Application', where its accuracy was notably low.

These results indicate that ensemble methods like Random Forest and Gradient Boosting are the most effective models for classifying both 'Application' and 'Attribution' in encrypted traffic, handling the complexity of the dataset more effectively than simpler classifiers.

9 Conclusions

The classification of encrypted network traffic based on a large dataset, as demonstrated in this project, highlights the potential and challenges of applying machine learning techniques to complex traffic patterns. Key findings and implications from the results include:
1. Effectiveness of Ensemble Methods: The results show that ensemble methods like Random Forest and Gradient Boosting consistently outperformed other machine learning models, achieving the highest accuracy and F1 scores for both the 'Application' and 'Attribution' labels. These models excel in handling the complex nature of encrypted traffic, effectively leveraging a variety of statistical, temporal, and protocol-specific features without needing to inspect encrypted content. Their strong performance makes them ideal for real-world applications where high accuracy is critical.
2. Feature Importance in Encrypted Traffic Classification: The choice of features, particularly flow-level features like packet size, inter-arrival times, and TLS metadata, played a crucial role in achieving high classification performance. This aligns with previous research, which emphasizes the importance of focusing on statistical and temporal characteristics in the absence of payload visibility. Additionally, the inclusion of DNS, IP, and ASN features provided valuable context for identifying traffic types, further improving model accuracy.
3. Challenges with Simpler Models: Simpler models, such as Naive Bayes and Support Vector Machine (SVM), struggled to handle the complexity of encrypted traffic classification, especially in distinguishing between diverse applications and services. Their poor performance underscores the need for more sophisticated methods when dealing with encrypted traffic, where traditional

packet inspection methods are not viable.
4. Real-World Relevance of the Dataset: The custom-built dataset created for this project, enriched with a variety of real-world network conditions like packet loss and delay, proved to be a critical factor in training models that can generalize well to actual network traffic. The dataset's diversity, covering a wide range of applications and interaction types, ensures that the models trained on it are robust and adaptable to various real-world scenarios. This is particularly important in the context of encrypted traffic, where limited visibility into payloads demands highly representative datasets.
5. Potential for Future Work: While this project has demonstrated the effectiveness of machine learning models in encrypted traffic classification, there is room for further exploration. Techniques like deep learning, particularly using raw traffic features, could potentially enhance classification performance even further.

In conclusion, this project demonstrates that encrypted traffic can be effectively classified using machine learning techniques, particularly ensemble methods, when combined with a comprehensive and diverse dataset. The results highlight the importance of feature selection and model choice in overcoming the challenges posed by encrypted traffic, offering valuable insights for both researchers and practitioners in the field of network security and traffic management. Future work should continue to explore advanced techniques, larger datasets, and adaptive methods to improve the classification of encrypted traffic in increasingly complex network environments.

References

[1] Bader, O., Lichy, A., Dvir, A., Dubin, R., Hajaj, C. (2024). *OSF-EIMTC: An open-source framework for standardized encrypted internet traffic classification.*

[2] Han, X., Liu, S., Liu, J., Jiang, B., Lu, Z., Liu, B. (2024). *ECNet: Robust malicious network traffic detection with multi-view feature and confidence mechanism.*

[3] Tong, Y., Chen, Y., Hwee, G. B., Cao, Q., Razul, S. G., Lin, Z. (2024). *A method for out-of-distribution detection in encrypted mobile traffic classification*

[4] Jorgensen, S., Holodnak, J., Dempsey, J., de Souza, K., Raghunath, A., Rivet, V., DeMoes, N., Alejos, A., Wollaber, A. (2024). *Extensible machine learning for encrypted network traffic application labeling via uncertainty quantification*

[5] Yang, L., Finamore, A., Jun, F., Rossi, D. (2021). *Deep learning and zero-day traffic classification: Lessons learned from a commercial-grade dataset*

[6] Zion, Y. (2018).*Classification and enrichment of encrypted traffic using machine learning algorithms*

[7] Luo, Z., Li, Y., Tan, S. (2024). *Enhancing flow embedding through trace: A novel self-supervised approach for encrypted traffic classification*

[8] Miao, G., Wu, G., Zhang, Z., Tong, Y., Lu, B. (2023). *SPN: A method of few-shot traffic classification with out-of-distribution detection based on Siamese prototypical network*

[9] Tian, S., Gong, F., Mo, S., Li, M., Wu, W., Xiao, D. *End-to-end encrypted network traffic classification method based on deep learning*

[10] B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz and M. E. Karsligil. (2017). *Application identification via network traffic classification*

[11] B. Wang, Y. Su, M. Zhang and J. Nie. *A Deep Hierarchical Network for Packet-Level Malicious Traffic Detection*

[12] Wang, B., Su, Y., Zhang, M., Nie, J. *A deep hierarchical network for packet-level malicious traffic detection*

[13] A. Patil and A. Deshpande (2023). *Evaluating ML models on CTU-13 and IOT-23 Datasets*

[14] JosephS. B., DadaE. G., YakubuH. J. (2023). *AUTONOMOUS NETWORK MONITORING USING COLLABORATIVE LEARNING FOR DISTRIBUTED TRAFFIC CLASSIFICATION*

[15] USTC-TFC2016 dataset. https://github.com/yungshenglu/USTC-TFC2016.git

[16] CTU-13 dataset. https://impactcybertrust.org/dataset$_view?idDataset$ = 945

[17] UNIBS-2009 dataset. http://netweb.ing.unibs.it/ ntw/tools/traces/

[18] ISCX-2016 dataset. https://www.unb.ca/cic/datasets/vpn.html