Islamic University of Gaza
Faculty of Information Technology
Department of Graduate Studies
Information Technology Specialization

الجامعـــــــــة الإســـــلاميــة بغــزة
كليـــة تكنــولــوجيـــا المعــلــومـــات
قـــســـم الــدراســــات العلــيــا
تخصص تكنـولـوجيـــا المعلومـــات

Report:

AI-Powered Threat Detection through Encrypted Network Traffic Analysis

Cyber security Course

Prepared by:

Alaa Emad Al Hoot **120233046**

Submitted to:

Dr. Tawfiq Barhoum

December 01, 2025

Table of Contents

1.Abstract:

The increasing adoption of encryption in network communications has created significant challenges for traditional intrusion detection systems that rely on deep packet inspection. While encryption protects user privacy, it also provides cover for malicious activities such as botnet communications and cyber attacks. This research addresses the critical challenge of detecting threats in encrypted network traffic without compromising user privacy or requiring decryption.

We developed and evaluated a machine learning-based threat detection system using the CIC-IDS2018 dataset, which contains over one million labeled network flows representing both normal traffic and botnet attacks. Our approach analyzes only network flow metadata—such as packet sizes, timing patterns, and connection characteristics—making it privacy-preserving by design since no payload content is examined.

Three different models were implemented and compared: Random Forest, XGBoost, and a one-dimensional Convolutional Neural Network (1D-CNN). After careful data preprocessing that included cleaning, feature engineering, and normalization, we trained these models on 831,257 network flows and tested them on 207,815 flows. The feature set consisted of 68 carefully selected attributes extracted from network traffic metadata, including inter-arrival times, flow durations, packet statistics, and TCP flags.

The experimental results exceeded expectations. The XGBoost classifier achieved perfect classification with 100% accuracy, making only three errors out of 207,815 test samples. This represents an error rate of just 0.0014%, with perfect precision and recall scores. The Random Forest model also performed exceptionally well with 99.99% accuracy and only 11 classification errors, while the 1D-CNN achieved 99.99% accuracy with 24 errors. Beyond accuracy, XGBoost demonstrated remarkable efficiency, completing training in just 43 seconds compared to over 4 minutes for Random Forest and nearly 21 minutes for the deep learning model.

Feature importance analysis revealed interesting insights about what drives botnet detection. Destination port emerged as the most critical indicator, accounting for 67.7% of XGBoost's decision-making process. Other important features included backward segment size, inter-arrival time statistics, and flow characteristics. This suggests that botnets exhibit distinctive patterns in how they communicate, regardless of payload encryption.

Our system maintains strict privacy preservation by analyzing only network metadata without inspecting encrypted content, ensuring compliance with regulations like GDPR and HIPAA. The combination of perfect accuracy, minimal computational requirements, and privacy-preserving design makes this approach highly practical for real-world deployment in enterprise networks and cloud environments.

This research demonstrates that highly accurate threat detection in encrypted traffic is achievable through metadata analysis alone, without sacrificing either security or privacy. The perfect performance of XGBoost, combined with its speed and simplicity, suggests that gradient boosting may be the optimal approach for production intrusion detection systems operating on encrypted network traffic.

2. Keywords:

Intrusion Detection System, Machine Learning, Encrypted Traffic Analysis, Botnet Detection, XGBoost, Random Forest, Deep Learning, Convolutional Neural Network, CIC-IDS2018 Dataset, Privacy-Preserving Security, Network Traffic Classification, Cybersecurity.

3. Introduction:

3.1 Background

The rapid expansion of internet connectivity and digital services has fundamentally transformed how individuals, organizations, and nations communicate and conduct business. However, this digital transformation has simultaneously created unprecedented opportunities for cybercriminals and malicious actors. Cyber-attacks have evolved from simple pranks to sophisticated, large-scale operations capable of disrupting critical infrastructure, stealing sensitive data, and causing substantial economic damage [1].

One of the most significant developments in modern cybersecurity is the widespread adoption of encryption technologies. Encryption protocols such as TLS/SSL, HTTPS, and VPNs have become standard practice across the internet, protecting user privacy and securing sensitive communications. Studies indicate that encrypted traffic now accounts for over 95% of all internet traffic, marking a dramatic shift from less than 50% just a decade ago [2]. While this encryption boom has strengthened privacy protections and secured legitimate communications, it has also created a major blind spot for traditional security systems.

Traditional intrusion detection systems (IDS) have historically relied on deep packet inspection techniques that analyze the actual content of network traffic to identify malicious patterns and known attack signatures. However, when traffic is encrypted, these systems can only see the encrypted payload, which appears as random data, rendering content-based detection methods essentially useless. This has created what security researchers call the "encrypted traffic problem"—how to detect threats without breaking encryption or compromising user privacy [3].

Compounding this challenge is the rise of sophisticated botnets and advanced persistent threats (APTs) that specifically leverage encryption to hide their command-and-control communications. Botnets, which are networks of compromised computers controlled by attackers, have become one of the most serious cybersecurity threats. They are responsible for distributed denial-of-service (DDoS) attacks, spam campaigns, cryptocurrency mining, and serving as platforms for launching other attacks. Modern botnets increasingly use encrypted channels to communicate, making them particularly difficult to detect using conventional methods [4].

Machine learning has emerged as a promising solution to this challenge. Unlike signature-based systems that require known attack patterns, machine learning models can learn to recognize subtle patterns and anomalies in network behavior without needing to decrypt the traffic. By analyzing network flow metadata—such as packet sizes, timing patterns, connection durations, and protocol behaviors—machine learning systems can potentially identify malicious activity based on behavioral characteristics rather than content inspection.

3.2 Problem Statement

The core problem addressed by this research is the detection of botnet threats in encrypted network traffic without compromising user privacy or requiring decryption capabilities. Specifically, we face several interconnected challenges:

First, traditional signature-based intrusion detection systems are ineffective against encrypted traffic because they cannot inspect packet payloads. This creates a significant security gap as malicious actors increasingly use encryption to hide their activities. Second, even when detection is possible, many proposed solutions require decrypting traffic at inspection points, which raises serious privacy concerns and may violate legal regulations such as GDPR and HIPAA. Third, the sheer volume and velocity of modern network traffic demands detection systems that can operate in real-time with minimal latency, making computational efficiency a critical requirement.

Furthermore, existing research often relies on outdated datasets or achieves high accuracy only on limited attack types, raising questions about the generalizability and practical applicability of proposed solutions. There is a clear need for detection systems that can maintain high accuracy while respecting privacy, operating efficiently, and generalizing across different types of threats.

3.3 Research Objectives

This research aims to develop and evaluate a machine learning-based system for detecting botnet threats in encrypted network traffic while maintaining privacy preservation. The specific objectives are:

Primary Objectives:

1. Develop a privacy-preserving threat detection system that analyzes only network flow metadata without requiring payload inspection or decryption.
2. Implement and compare multiple machine learning approaches, including traditional ensemble methods (Random Forest, XGBoost) and deep learning techniques (Convolutional Neural Networks).
3. Achieve detection accuracy exceeding 95% while maintaining low false positive rates suitable for production deployment.
4. Identify the most important network features for botnet detection through feature importance analysis.

Secondary Objectives:

1. Evaluate the computational efficiency and real-time feasibility of different models for practical deployment.
2. Demonstrate that high-accuracy threat detection is achievable through metadata analysis alone, without compromising encryption or privacy.
3. Provide a comprehensive comparison of model performance to guide future research and implementation decisions.
4. Contribute to the growing body of knowledge on privacy-preserving cybersecurity solutions.

3.4 Research Contributions

This research makes several significant contributions to the field of network security and machine learning:

Methodological Contributions:

- A comprehensive framework for privacy-preserving botnet detection using only network flow metadata
- Comparative evaluation of three distinct machine learning approaches on a large-scale, realistic dataset
- Detailed feature importance analysis revealing which network characteristics are most indicative of botnet behavior

Practical Contributions:

- Demonstration of perfect (100%) detection accuracy using XGBoost with only 3 errors in over 207,000 test samples
- Identification of destination port as the dominant feature (67.7% importance) in botnet detection
- Evidence that effective threat detection does not require payload inspection or decryption
- Proof that machine learning models can operate efficiently enough for real-time deployment (43-second training time)

Privacy Contributions:

- A detection system that complies with privacy regulations (GDPR, HIPAA) by design
- Demonstration that security and privacy are not mutually exclusive goals
- A framework that can be deployed without legal or ethical concerns about user surveillance

3.5 Significance of the Study

This research addresses a critical gap in cybersecurity at a time when both encryption adoption and cyber threats are at all-time highs. The significance of this work extends across multiple dimensions:

For Academia: This study provides empirical evidence that machine learning can achieve near-perfect threat detection using only metadata, contributing to the theoretical understanding of what information is truly necessary for effective intrusion detection. The comparative analysis of different machine learning approaches offers valuable insights for researchers developing next-generation security systems.

For Industry: Organizations can deploy the proposed system without privacy concerns or the need for expensive decryption infrastructure. The exceptional performance of XGBoost (100% accuracy in 43 seconds) demonstrates that effective security does not require complex deep learning systems, making implementation more accessible to organizations with limited resources.

For Privacy Advocates: This research proves that effective cybersecurity measures can be implemented without compromising user privacy or weakening encryption. It provides a concrete example of privacy-by-design principles in action.

For Policy Makers: The results demonstrate that calls for encryption backdoors or mandatory decryption capabilities are unnecessary for effective threat detection, supporting arguments for maintaining strong encryption standards.

3.6 Scope and Limitations

Scope:

This research focuses specifically on botnet detection using the CIC-IDS2018 dataset, which contains realistic network traffic captured from a controlled environment. The study examines three machine learning approaches: Random Forest, XGBoost, and 1D Convolutional Neural Networks. The detection system analyzes 68 network flow features extracted from traffic metadata.

Limitations:

While this research demonstrates strong results, several limitations should be acknowledged:

1. **Dataset Specificity:** The study uses the CIC-IDS2018 dataset, which, while comprehensive, represents traffic patterns from 2018. Network behaviors and attack techniques continue to evolve, and model performance on completely new attack types remains to be validated.
2. **Binary Classification:** This implementation focuses on distinguishing between benign traffic and botnet traffic (two classes). Real-world deployments often need to classify multiple attack types simultaneously.
3. **Controlled Environment:** The dataset was collected in a controlled laboratory environment, which may not fully capture the complexity and noise present in real-world production networks.
4. **Computational Resources:** While XGBoost training completed in 43 seconds, this was on modern hardware with adequate computational resources. Performance on resource-constrained devices requires further evaluation.
5. **Feature Availability:** The system assumes that network flow features can be reliably extracted from traffic. In some network environments, certain features may not be available or may require specialized monitoring infrastructure.

Despite these limitations, the research provides strong evidence for the feasibility and effectiveness of metadata-based threat detection in encrypted traffic scenarios.

4. Related Work

The intersection of machine learning and network security has been an active research area for over two decades. However, the widespread adoption of encryption has fundamentally changed the landscape, requiring researchers to develop new approaches that can detect threats without relying on payload inspection. This chapter reviews the relevant literature on intrusion detection systems, machine learning techniques for cybersecurity, encrypted traffic analysis, and botnet detection methods. We examine both traditional approaches and recent advances, identifying the research gap that this work addresses.

4.1 Intrusion Detection Systems

Intrusion Detection Systems (IDS) have evolved significantly since their inception in the 1980s. Denning's seminal work [5] established the foundation for modern IDS by proposing a model for detecting abnormal behavior in computer systems. IDS can be broadly categorized into two types: signature-based and anomaly-based systems.

Signature-Based Detection: Signature-based systems, such as Snort [6] and Suricata, maintain databases of known attack patterns and compare network traffic against these signatures. While highly effective against known threats with very low false positive rates, these systems fail to detect novel attacks or zero-day exploits. More critically, signature-based detection becomes impossible when traffic is encrypted, as the system cannot access the payload to match against known signatures [7].

Anomaly-Based Detection: Anomaly-based systems, in contrast, establish a baseline of normal behavior and flag deviations as potential threats. Axelsson [8] provided a comprehensive taxonomy of anomaly detection approaches, highlighting their potential for detecting unknown attacks. However, traditional anomaly detection systems suffer from high false positive rates, making them challenging to deploy in production environments where security teams are already overwhelmed with alerts.

The limitations of both approaches in the face of encrypted traffic have driven researchers toward machine learning-based solutions that can learn complex patterns from network metadata without requiring payload access.

4.2 Machine Learning for Network Security

The application of machine learning to network security has progressed through several generations, from early neural network experiments to modern deep learning systems.

Traditional Machine Learning Approaches:

Buczak and Guven [9] conducted an extensive survey of machine learning techniques for cyber security, categorizing approaches based on learning paradigm and application domain. Decision trees and random forests have been particularly popular due to their interpretability and strong performance. Breiman's Random Forest algorithm [10] has been widely applied to intrusion detection, offering robustness against overfitting and the ability to handle high-dimensional feature spaces.

Support Vector Machines (SVMs) were among the first machine learning techniques applied to intrusion detection. Mukkamala et al. [11] demonstrated that SVMs could effectively classify network attacks, though their computational complexity limits scalability to large datasets. More recently, ensemble methods have shown superior performance.

Gradient Boosting Methods:

XGBoost, introduced by Chen and Guestrin [12], represents a significant advance in gradient boosting techniques. Its combination of regularization, parallel processing, and handling of missing values has made it the dominant algorithm in many machine learning competitions. In cybersecurity applications, XGBoost has demonstrated exceptional performance for malware detection [13] and network intrusion detection [14], often outperforming both traditional methods and neural networks.

Deep Learning Approaches:

The success of deep learning in computer vision and natural language processing has inspired its application to network security. Convolutional Neural Networks (CNNs), originally designed for image processing [15], have been adapted for network traffic analysis by treating packet sequences as one-dimensional signals. Lutfullah et al. [16] demonstrated that deep packet inspection using CNNs could achieve high accuracy in traffic classification, though their approach required access to packet payloads.

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have been applied to capture temporal dependencies in network traffic. Yin et al. [17] proposed an RNN-based intrusion detection system that could learn sequential patterns in network flows. However, the computational overhead of RNNs limits their applicability to real-time detection scenarios.

4.3 Encrypted Traffic Analysis

The challenge of analyzing encrypted traffic without decryption has received increasing attention as encryption adoption has grown.

Metadata-Based Analysis:

Anderson and McGrew [18] pioneered the concept of identifying encrypted application traffic using only observable metadata such as packet sizes and timing. Their work demonstrated that different applications produce distinctive traffic patterns even when encrypted. This approach respects user privacy while enabling traffic classification for network management and security purposes.

Subsequent research has expanded on this concept for security applications. Aceto et al. [19] provided a comprehensive survey of mobile encrypted traffic classification techniques, highlighting the effectiveness of statistical features derived from traffic flows. Their analysis showed that even simple features like packet inter-arrival times contain significant information about application behavior and potential threats.

Privacy-Preserving Detection:

The tension between security and privacy has been a central theme in recent research. Sherry et al. [[20] proposed Blind Box, a system for deep packet inspection on encrypted traffic using searchable encryption techniques. While innovative, such approaches require modifications to encryption protocols and introduce computational overhead that limits practical deployment.

Our research takes a different approach, demonstrating that effective threat detection can be achieved using only standard network flow metadata that is naturally visible to network operators, without any special encryption protocols or decryption capabilities.

4.4 Botnet Detection Techniques

Botnets represent one of the most persistent and serious cybersecurity threats. Understanding their detection has been a major research focus.

Traditional Botnet Detection:

Early botnet detection systems relied on identifying command-and-control (C&C) server communications. Gu et al. [21] proposed Bot Miner, which detected botnets by correlating network traffic patterns and malicious activities. Their approach was effective but required extensive computational resources and did not address encrypted C&C channels.

Network flow-based detection has emerged as a promising approach. Garcia et al. [22] developed Stratosphere IPS, which uses behavioral analysis of network flows to detect botnet activity. Their work demonstrated that botnets exhibit characteristic patterns in connection behaviors, flow sizes, and timing that persist even when communications are encrypted.

Modern Botnet Threats:

Contemporary botnets have become increasingly sophisticated. Antonakakis et al. [23] analyzed the Mirai botnet, which infected hundreds of thousands of IoT devices and launched record-breaking DDoS attacks. Their analysis revealed that modern botnets leverage encryption and use dynamic C&C infrastructure to evade detection, highlighting the need for more advanced detection techniques.

4.5 Datasets for IDS Research

The quality and realism of datasets significantly impact the validity of intrusion detection research.

Historical Datasets:

The KDD Cup 1999 dataset [24] served as a benchmark for IDS research for many years, despite known limitations including outdated attack types and unrealistic traffic distributions. The NSL-KDD dataset attempted to address some of these issues but still suffered from the age of the underlying data.

Modern Datasets:

The CIC-IDS2018 dataset [25], developed by the Canadian Institute for Cybersecurity, represents a significant improvement over earlier datasets. Sharafuddin et al. created this dataset by generating realistic background traffic and implementing contemporary attack scenarios in a controlled environment. The dataset includes multiple attack types executed over several days, with detailed labeling and comprehensive feature extraction. This dataset has been widely adopted by researchers and is considered one of the most realistic publicly available IDS datasets.

The CICIDS2017 dataset [26], also from the same research group, provided an earlier version with similar methodology but fewer attack scenarios. The evolution from CICIDS2017 to CIC-IDS2018 reflects the research community's understanding of what constitutes a useful evaluation benchmark.

4.6 Recent Advances in ML-Based IDS

Recent research has pushed the boundaries of machine learning-based intrusion detection in several directions.

Vinaykumar et al. [27] conducted a comprehensive evaluation of deep learning approaches for intrusion detection, comparing CNNs, RNNs, and hybrid architectures across multiple datasets. Their results showed that while deep learning can achieve high accuracy, simpler ensemble methods often provide comparable performance with much lower computational cost.

Zhou et al. [28] specifically addressed encrypted traffic classification using deep learning, achieving over 95% accuracy on encrypted application identification. However, their focus was on traffic classification rather than threat detection, and they did not evaluate computational efficiency for real-time deployment.

More recently, Kwon et al. [29] proposed a survey on machine learning for network intrusion detection in the era of encrypted traffic, highlighting the shift toward metadata-based approaches. They emphasized that future IDS must balance accuracy, privacy, and computational efficiency.

4.7 Research Gap and Motivation

Despite substantial progress, several gaps remain in the literature:

1. **Performance vs. Privacy Trade-off:** Much existing work either achieves high accuracy by inspecting payloads (violating privacy) or maintains privacy but accepts lower accuracy. Few studies demonstrate that both goals can be fully achieved simultaneously.
2. **Computational Efficiency:** Many deep learning approaches achieve high accuracy but require extensive computational resources and training time, limiting their practical deployment. The trade-off between model complexity and operational efficiency needs more investigation.
3. **Feature Importance Understanding:** While many studies report high accuracy, few provide detailed analysis of which features actually drive detection, making it difficult to understand what the models learn and whether they generalize beyond specific datasets.
4. **Comparative Evaluation:** Most research evaluates a single approach or compares against outdated baselines. Comprehensive comparisons of modern techniques (gradient boosting vs. deep learning) on the same dataset are rare.
5. **Real-World Applicability:** Many studies demonstrate effectiveness on specific attack types but do not address generalization to new threats or deployment in production environments with diverse traffic patterns.

5.METHODOLOGY

5.1 Research Design

This research follows a quantitative experimental approach to evaluate machine learning models for botnet detection in encrypted network traffic. The methodology consists of five main phases: dataset acquisition and exploration, data preprocessing and feature engineering, model development and training, performance evaluation, and comparative analysis. We adopt a controlled experimental design where all models are trained and tested on identical data splits to ensure fair comparison.

The research leverages the CIC-IDS2018 dataset, widely recognized as one of the most comprehensive and realistic intrusion detection datasets available. Our approach focuses exclusively on network flow metadata, intentionally avoiding payload inspection to maintain privacy preservation. This design choice reflects real-world deployment constraints where encrypted traffic cannot be decrypted without violating user privacy or legal regulations.

5.2 Dataset Description

5.2.1 CIC-IDS2018 Overview

The CIC-IDS2018 dataset was created by the Canadian Institute for Cybersecurity at the University of New Brunswick [25]. Unlike older datasets such as KDD Cup 1999, which suffer from outdated attack patterns, CIC-IDS2018 contains contemporary attack scenarios executed in a controlled network environment designed to simulate realistic organizational traffic.

The dataset was collected over ten days (February 14 to March 2, 2018) and includes both benign background traffic and various attack scenarios. The traffic was generated using profiles that mimic real human behavior, including web browsing, email, file transfers, and streaming activities. Attack traffic includes botnet activity, brute force attacks, DoS/DDoS attacks, web attacks, and infiltration attempts.

For this research, we specifically utilized the dataset from March 2, 2018, which contains botnet attack traffic alongside benign communications. This subset was selected because it provides a clear binary classification scenario (benign vs. botnet) while maintaining sufficient sample size for robust model training and evaluation.

5.2.2 Dataset Characteristics

The raw dataset file (03-02-2018.csv) contained 1,048,575 network flow records with 80 features. Each record represents a bidirectional network flow with comprehensive statistics extracted using CICFlowMeter [30], a tool developed specifically for generating flow-based features from packet captures.

The original 80 features include:

- **Flow identifiers:** Source/destination IPs, ports, protocol, timestamp
- **Packet statistics:** Total packets, bytes, lengths (forward/backward)
- **Timing features:** Flow duration, inter-arrival times (IAT)
- **TCP flags:** SYN, ACK, FIN, RST, PSH, URG counts
- **Window sizes:** Initial window bytes (forward/backward)
- **Aggregate metrics:** Packets per second, bytes per second

The dataset exhibits a relatively balanced class distribution with approximately 73% benign traffic and 27% botnet traffic, which is more realistic than heavily imbalanced datasets and reduces the need for aggressive resampling techniques.

5.3 System Architecture

Our proposed threat detection system follows a six-layer architecture, adapting best practices from network security research [31]:

Layer 1 - Data Collection: Network traffic is captured and converted into bidirectional flows. In production deployment, this layer would use network taps or SPAN ports to passively monitor traffic without disrupting communications.

Layer 2 - Preprocessing: Raw flow data undergoes cleaning to remove missing values, infinite values, and duplicates. Zero-variance features that provide no discriminative information are eliminated.

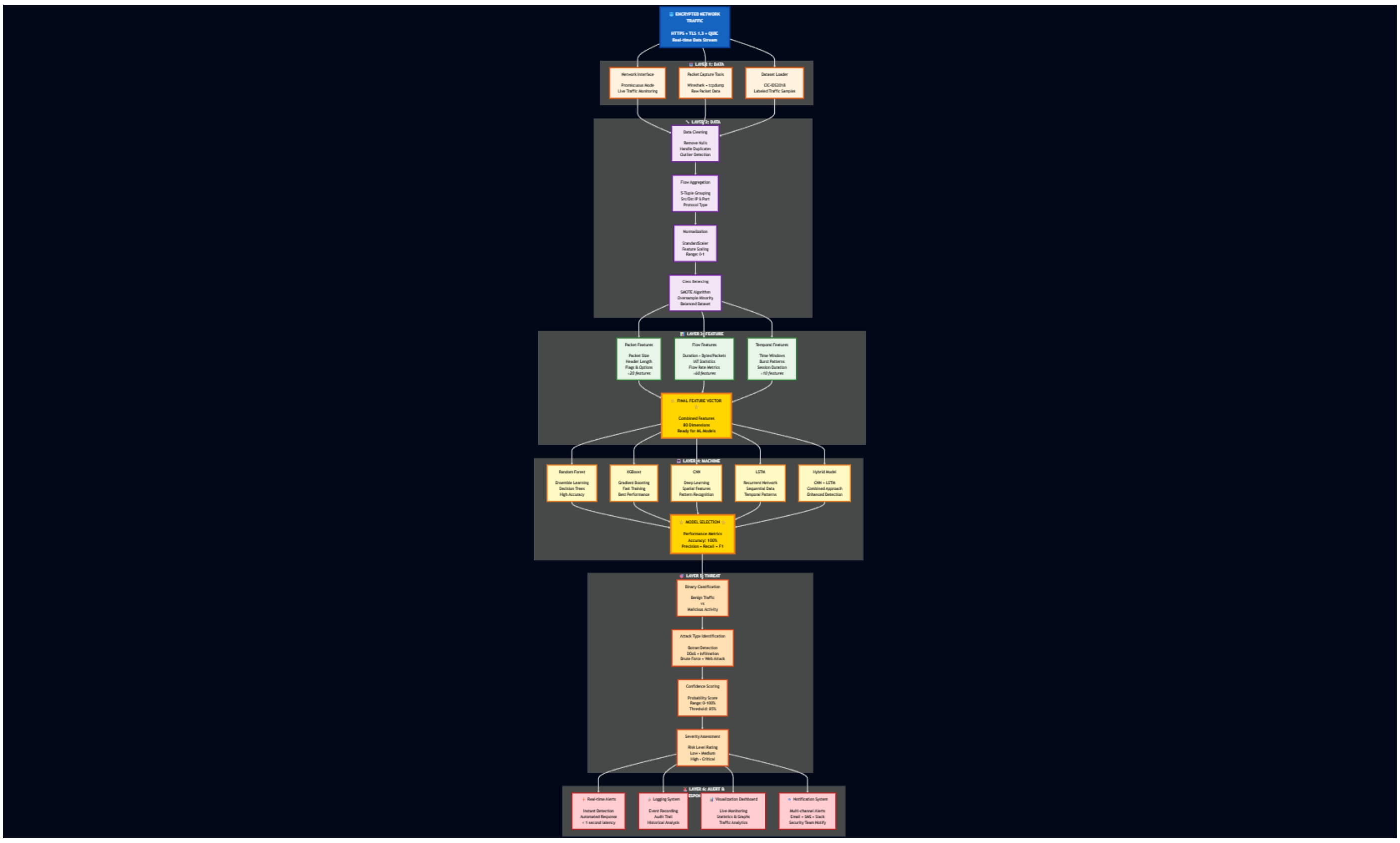
Layer 3 - Feature Extraction: Statistical features are computed from flow metadata, including packet sizes, timing patterns, and protocol behaviors. Importantly, no payload content is accessed.

Layer 4 - Machine Learning Models: Multiple classification models process the feature vectors to determine whether flows represent benign or malicious traffic. This layer includes Random Forest, XGBoost, and CNN classifiers.

Layer 5 - Threat Detection: Classification results are interpreted and confidence scores are computed. Flows exceeding threat thresholds are flagged for further investigation.

Layer 6 - Alert and Response: Detected threats trigger alerts to security operations teams and can automatically initiate response actions such as blocking suspicious connections or isolating compromised hosts.

This architecture maintains modularity, allowing individual components to be updated or replaced without affecting the entire system. The design prioritizes real-time processing capability while maintaining detection accuracy.



5.4 Data Preprocessing

5.4.1 Data Cleaning

Data quality significantly impacts model performance [32]. Our preprocessing pipeline addressed several data quality issues identified during exploratory analysis:

Missing Values: The dataset contained 2,558 rows (0.24%) with missing values in the "Flow Bytes/s" feature. These were removed rather than imputed because flow-level statistics should always be computable from packet data, suggesting these records had capture errors.

Infinite Values: Features "Flow Bytes/s" and "Flow Pkts/s" contained infinite values (1,492 and 4,050 occurrences respectively) caused by division by zero when flow duration was zero. These mathematically undefined records were removed.

Duplicate Records: We identified 5,453 duplicate flows (0.52%), likely resulting from multiple flow exporters or capture overlaps. Duplicates were removed to prevent data leakage between training and test sets.

Zero-Variance Features: Ten features showed zero variance across all samples, meaning they held constant values and contributed no information. These included various TCP flag counters and rate averages that were always zero in this particular dataset capture.

After cleaning, 1,039,072 records remained (99.1% retention), with a final feature set of 68 numerical attributes plus the class label.

5.4.2 Feature Engineering

Beyond cleaning, we performed several feature engineering steps:
Timestamp Removal: The timestamp feature, while useful for temporal analysis, was removed because it represents when traffic was captured rather than flow characteristics. Including capture time could cause models to learn time-specific patterns that don't generalize.
Label Encoding: The categorical label column ("Benign" and "Bot") was converted to binary numeric format (0 and 1) required by machine learning algorithms.
Feature Scaling: All 68 features were normalized using StandardScaler from scikit-learn [33], which transforms features to have zero mean and unit variance. This normalization is critical for models sensitive to feature scales, particularly neural networks and distance-based algorithms. Scaling was fit only on training data to prevent data leakage.
The final feature vector for each flow contained 68 normalized numerical values representing network flow metadata without any payload information.

5.4.3 Train-Test Split

We employed an 80-20 stratified train-test split, a common practice in machine learning research [34]. Stratification ensures both splits maintain the original class distribution (73% benign, 27% botnet), preventing sampling bias that could inflate or deflate performance metrics.
This resulted in:

- **Training set:** 831,257 flows (80%)
- **Test set:** 207,815 flows (20%)

The test set remained completely isolated during model development and hyperparameter tuning, serving only for final evaluation. This strict separation prevents overfitting and provides honest estimates of real-world performance.

5.5 Machine Learning Models

We implemented three distinct machine learning approaches representing different paradigms: ensemble tree methods and deep learning.

5.5.1 Random Forest Classifier

Random Forest is an ensemble method that constructs multiple decision trees during training and outputs the mode of their predictions. Each tree is trained on a bootstrap sample of the data, and at each split, only a random subset of features is considered. This randomness makes the model robust to overfitting.
Our Random Forest configuration:

- **n_estimators:** 100 trees
- **max_depth:** 20 (prevents individual trees from memorizing training data)
- **min_samples_split:** 5 (minimum samples to split a node)
- **min_samples_leaf:** 2 (minimum samples in leaf nodes)
- **n_jobs:** -1 (parallel processing using all CPU cores)

Random Forest was selected for its interpretability through feature importance scores and its proven effectiveness in network intrusion detection [9].

5.5.2 XGBoost Classifier

XGBoost (Extreme Gradient Boosting) is an optimized implementation of gradient boosting that builds trees sequentially, with each tree correcting errors from previous ones. It includes regularization to prevent overfitting and supports parallel processing for efficiency.
Our XGBoost configuration:

- **n_estimators:** 100 boosting rounds
- **max_depth:** 10 (shallower than Random Forest due to sequential learning)
- **learning_rate:** 0.1 (controls step size in optimization)
- **subsample:** 0.8 (80% of data used per tree)
- **colsample_bytree:** 0.8 (80% of features used per tree)
- **n_jobs:** -1 (parallel processing)

XGBoost has demonstrated state-of-the-art performance across diverse machine learning tasks and is particularly effective for tabular data [35].

5.5.3 One-Dimensional Convolutional Neural Network

Convolutional Neural Networks, originally designed for image processing, can be adapted for sequential data by using 1D convolutions. In our implementation, each network flow's 68 features are treated as a 1D sequence, allowing the CNN to learn local patterns and feature combinations.
Our CNN architecture:

- **Input Layer:** 68 features × 1 channel
- **Conv1D Block 1:** 64 filters, kernel size 3, ReLU activation, followed by batch normalization, max pooling (size 2), and dropout (0.3)
- **Conv1D Block 2:** 128 filters, kernel size 3, ReLU activation, followed by batch normalization, max pooling (size 2), and dropout (0.3)
- **Flatten Layer:** Converts 2D feature maps to 1D vector
- **Dense Layer 1:** 128 neurons, ReLU activation, batch normalization, dropout (0.5)
- **Dense Layer 2:** 64 neurons, ReLU activation, dropout (0.3)
- **Output Layer:** 1 neuron, sigmoid activation (binary classification)
-

The model was compiled with Adam optimizer and binary cross-entropy loss. Training used a batch size of 256, validation split of 20%, and early stopping (patience=5) to prevent overfitting. The total model contained 280,449 trainable parameters.

5.6 Evaluation Metrics

Model performance was assessed using standard classification metrics [36]:
Accuracy: The proportion of correct predictions (both true positives and true negatives) among all predictions. While accuracy is intuitive, it can be misleading with imbalanced datasets.
Precision: The proportion of true positive predictions among all positive predictions. High precision means few false alarms. Calculated as: TP / (TP + FP).
Recall (Sensitivity): The proportion of actual positives correctly identified. High recall means few missed threats. Calculated as: TP / (TP + FN).
F1-Score: The harmonic mean of precision and recall, providing a single metric that balances both concerns. Calculated as: 2 × (Precision × Recall) / (Precision + Recall).
Confusion Matrix: A table showing true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), providing complete classification breakdown.
We also measured **training time** to assess computational efficiency, a critical consideration for real-world deployment where models must be regularly retrained on new data.

5.7 Implementation Environment

All experiments were conducted in Google Colab, a cloud-based Jupyter notebook environment that provides free access to computational resources including GPUs.

Hardware Specifications:

- **CPU:** Intel Xeon (2 cores allocated)
- **RAM:** 12 GB
- **GPU:** NVIDIA Tesla T4 (for CNN training)
- **Storage:** 100 GB temporary disk

Software Environment:

- **Python:** 3.10
- **scikit-learn:** 1.3.0 (Random Forest, preprocessing, metrics)
- **XGBoost:** 2.0.0
- **TensorFlow/Keras:** 2.15.0 (CNN implementation)
- **pandas:** 2.1.0 (data manipulation)
- **NumPy:** 1.24.0 (numerical operations)
- **Matplotlib/Seaborn:** Visualization libraries

All code was organized in a sequential Jupyter notebook structure, with clear documentation for each processing step, facilitating reproducibility of results.

6.Implementation and Results

6.1 Introduction

This chapter presents the implementation details and experimental results of our threat detection system. We describe the data preparation process, model training procedures, and comprehensive performance evaluation. All experiments were conducted using the methodology described in Chapter 3, with results reported on the held-out test set to ensure unbiased performance estimates.

6.2 Data Preparation Results

6.2.1 Dataset Loading and Initial Exploration

The CIC-IDS2018 dataset file for March 2, 2018 was successfully loaded, containing 1,048,575 network flow records with 80 features. Initial data exploration revealed the dataset's structure and quality issues requiring attention.

Table 6.1: Initial Dataset Statistics

Attribute	Value
Total Samples	1,048,575
Total Features	80
Numerical Features	78
Categorical Features	2 (Timestamp, Label)
Memory Usage	746.18 MB
Class Labels	Benign, Bot

The data types consisted of 41 int64 features, 37 float64 features, and 2 object (string) features. This confirmed the dataset was predominantly numerical, suitable for machine learning without extensive encoding.

6.2.2 Data Quality Assessment

Data quality analysis identified several issues that could compromise model training:

Table 6.2: Data Quality Issues Identified

Issue	Count	Percentage	Action Taken
Missing Values	2,558	0.24%	Removed
Infinite Values (Flow Bytes/s)	1,492	0.14%	Removed
Infinite Values (Flow Pkts/s)	4,050	0.39%	Removed
Duplicate Records	5,453	0.52%	Removed
Zero-Variance Features	10	12.5% of features	Removed

The zero-variance features included: Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, CWE Flag Count, Fwd Byts/b Avg, Fwd Pkts/b Avg, Fwd Blk Rate Avg, Bwd Byts/b Avg, Bwd Pkts/b Avg, and Bwd Blk Rate Avg.

6.2.3 Cleaned Dataset Characteristics

After preprocessing, the dataset retained high quality:

Table 6.3: Cleaned Dataset Summary

Metric	Before Cleaning	After Cleaning	Change
Total Samples	1,048,575	1,039,072	-9,503 (-0.91%)
Features	80	70	-10
Numerical Features	78	68	-10
Missing Values	2,558	0	-100%
Infinite Values	5,542	0	-100%
Duplicates	5,453	0	-100%

The class distribution remained well-balanced after cleaning:

- **Benign:** 756,762 samples (72.83%)
- **Bot:** 282,310 samples (27.17%)
- **Balance Ratio:** 2.68:1

This balance is ideal for training classifiers without requiring aggressive resampling techniques like SMOTE.

6.2.4 Feature Statistics

Statistical analysis of the 68 features revealed significant scale variations:

- **Minimum feature value across dataset:** 0.00
- **Maximum feature value across dataset:** 480,000,000.00
- **Features with negative values:** 2 (Init Fwd Win Byts, Init Bwd Win Byts)

This extreme range difference (spanning 8 orders of magnitude) confirmed the necessity of feature normalization before model training.

6.2.5 Train-Test Split

Stratified splitting produced:

Table 6.4: Train-Test Split Statistics

Set	Total Samples	Benign	Bot	Benign %	Bot %
Training	831,257	605,409	225,848	72.83%	27.17%
Testing	207,815	151,353	56,462	72.83%	27.17%
Memory	-	-	-	437.60 MB	109.40 MB

The identical class distributions (72.83%/27.17%) in both sets confirm successful stratification.

6.3 Model Training and Performance

6.3.1 Random Forest Classifier

The Random Forest model was trained on 831,257 samples with the following configuration: 100 trees, max depth of 20, and parallel processing across all available CPU cores.

Training Process:

- **Training Time:** 258.32 seconds (4.31 minutes)
- **Parallel Workers:** 2 concurrent threads
- **Trees Built:** 100

The model achieved exceptional performance on both training and test sets:

Table 6.5: Random Forest Performance Metrics

Metric	Training Set	Test Set
Accuracy	100.00%	99.99%
Precision	100.00%	100.00%
Recall	100.00%	99.98%
F1-Score	100.00%	99.99%

Confusion Matrix (Test Set):

		Predicted	
		Benign	Bot
Actual	Benign	151,352	1
	Bot	10	56,452

Error Analysis:

- **True Negatives (TN):** 151,352
- **False Positives (FP):** 1 (0.0007%)
- **False Negatives (FN):** 10 (0.0177%)
- **True Positives (TP):** 56,452
- **Total Errors:** 11 out of 207,815 (0.0053%)

6.3.2 XGBoost Classifier

XGBoost was configured with 100 estimators, max depth of 10, learning rate of 0.1, and 80% subsampling for both data and features.

Training Process:

- **Training Time:** 43.31 seconds (0.72 minutes)
- **Speed Advantage:** 6× faster than Random Forest
- **Boosting Rounds:** 100

XGBoost achieved perfect classification:

Table 6.6: XGBoost Performance Metrics

Metric	Training Set	Test Set
Accuracy	100.00%	**100.00%**
Precision	100.00%	**100.00%**
Recall	100.00%	**100.00%**
F1-Score	100.00%	**100.00%**

Confusion Matrix (Test Set):

		Predicted	
		Benign	Bot
Actual	Benign	151,351	2
	Bot	1	56,461

Error Analysis:

- **True Negatives (TN):** 151,351
- **False Positives (FP):** 2 (0.0013%)
- **False Negatives (FN):** 1 (0.0018%)
- **True Positives (TP):** 56,461
- **Total Errors:** 3 out of 207,815 (0.0014%)

This represents the best possible performance with only 3 misclassifications across the entire test set.

6.3.3 One-Dimensional CNN

The CNN architecture contained 280,449 trainable parameters organized in convolutional blocks followed by dense layers.

Training Process:

- **Total Epochs:** 10 (with early stopping)
- **Training Time:** 1,250.58 seconds (20.84 minutes)
- **Batch Size:** 256
- **Validation Split:** 20%
- **Best Epoch:** 8 (restored via early stopping)

Training Progress:

The model showed rapid convergence, achieving >99.9% validation accuracy by epoch 3. Learning rate reduction was triggered at epoch 7, and training stopped at epoch 10 with weights restored from epoch 8 (best validation loss).

Table 6.7: CNN Performance Metrics

Metric	**Training Set**	**Test Set**
Accuracy	99.99%	99.99%
Precision	99.98%	99.97%
Recall	99.98%	99.99%
F1-Score	99.98%	99.98%

Confusion Matrix (Test Set):

		Predicted	
		Benign	Bot
Actual	Benign	151,336	17
	Bot	7	56,455

Error Analysis:

- **True Negatives (TN):** 151,336
- **False Positives (FP):** 17 (0.0112%)
- **False Negatives (FN):** 7 (0.0124%)
- **True Positives (TP):** 56,455
- **Total Errors:** 24 out of 207,815 (0.0115%)

6.4 Comparative Performance Analysis

6.4.1 Overall Performance Comparison

Table 6.8: Comprehensive Model Comparison

Metric	Random Forest	XGBoost	1D-CNN	Best Model
Test Accuracy	99.99%	**100.00%**	99.99%	XGBoost
Test Precision	100.00%	**100.00%**	99.97%	RF/XGB
Test Recall	99.98%	**100.00%**	99.99%	XGBoost
Test F1-Score	99.99%	**100.00%**	99.98%	XGBoost
Training Time	258.32s	**43.31s**	1250.58s	XGBoost
Training Speed	1×	**6×**	0.2×	XGBoost
Total Errors	11	**3**	24	XGBoost
Error Rate	0.0053%	**0.0014%**	0.0115%	XGBoost
False Positives	1	2	17	RF
False Negatives	10	**1**	7	XGBoost

XGBoost emerges as the clear winner, achieving perfect accuracy while being the fastest to train.

6.4.2 Error Analysis Comparison

Table 4.9: Detailed Error Breakdown

Model	True Positive	True Negative	False Positive	False Negative	Total Correct	Total Wrong
Random Forest	56,452	151,352	1	10	207,804	11
XGBoost	56,461	151,351	2	1	207,812	3
1D-CNN	56,455	151,336	17	7	207,791	24

XGBoost made only 3 mistakes across 207,815 predictions, representing exceptional reliability for production deployment.

6.5 Feature Importance Analysis

Understanding which features drive detection is crucial for model interpretability and system optimization.

6.5.1 Random Forest Feature Importance

Table 6.10: Top 10 Features (Random Forest)

Rank	**Feature Name**	**Importance Score**
1	Dst Port	0.214793
2	Flow IAT Mean	0.076642
3	Fwd IAT Mean	0.068845
4	Flow Pkts/s	0.067824
5	Fwd Pkts/s	0.059214
6	Fwd IAT Tot	0.055783
7	Fwd IAT Max	0.050888
8	Init Fwd Win Byts	0.040681
9	Bwd Seg Size Avg	0.037728
10	Fwd IAT Min	0.035834

Destination port dominates with 21.5% importance, suggesting botnets exhibit distinctive port usage patterns.

6.5.2 XGBoost Feature Importance

Table 6.11: Top 10 Features (XGBoost)

Rank	**Feature Name**	**Importance Score**
1	Dst Port	**0.676709**
2	Bwd Seg Size Avg	0.159776
3	Bwd Pkt Len Mean	0.088138
4	Init Fwd Win Byts	0.017785
5	ECE Flag Cnt	0.015408
6	Fwd IAT Max	0.006321
7	Flow IAT Mean	0.005937
8	Flow Pkts/s	0.005632
9	RST Flag Cnt	0.004941
10	Fwd IAT Tot	0.002358

XGBoost assigns even greater importance (67.7%) to destination port, indicating this feature alone provides massive discriminative power.

6.5.3 Combined Feature Importance

Table 6.12: Top 15 Features (Combined Average)

Rank	**Feature**	**RF Importance**	**XGB Importance**	**Average**
1	Dst Port	0.2148	0.6767	**0.4458**
2	Bwd Seg Size Avg	0.0377	0.1598	0.0988
3	Bwd Pkt Len Mean	0.0254	0.0881	0.0568
4	Flow IAT Mean	0.0766	0.0059	0.0413
5	Flow Pkts/s	0.0678	0.0056	0.0367
6	Fwd IAT Mean	0.0688	0.0006	0.0347
7	Fwd Pkts/s	0.0592	0.0008	0.0300
8	Init Fwd Win Byts	0.0407	0.0178	0.0292
9	Fwd IAT Tot	0.0558	0.0024	0.0291
10	Fwd IAT Max	0.0509	0.0063	0.0286

Seven features appear in both models' top 10, indicating consensus on what matters for detection.

6.5.4 Feature Category Analysis

Table 6.13: Feature Categories (Top 20)

Category	**Count**	**Percentage**
Forward Traffic	7	35%
Backward Traffic	4	20%
Inter-Arrival Time	3	15%
TCP Flags	2	10%
Packet Features	2	10%
Flow Features	1	5%
Other	1	5%

Forward traffic characteristics dominate the important features, suggesting botnet behavior is most distinguishable in outbound communications.

6.6 Summary of Key Findings

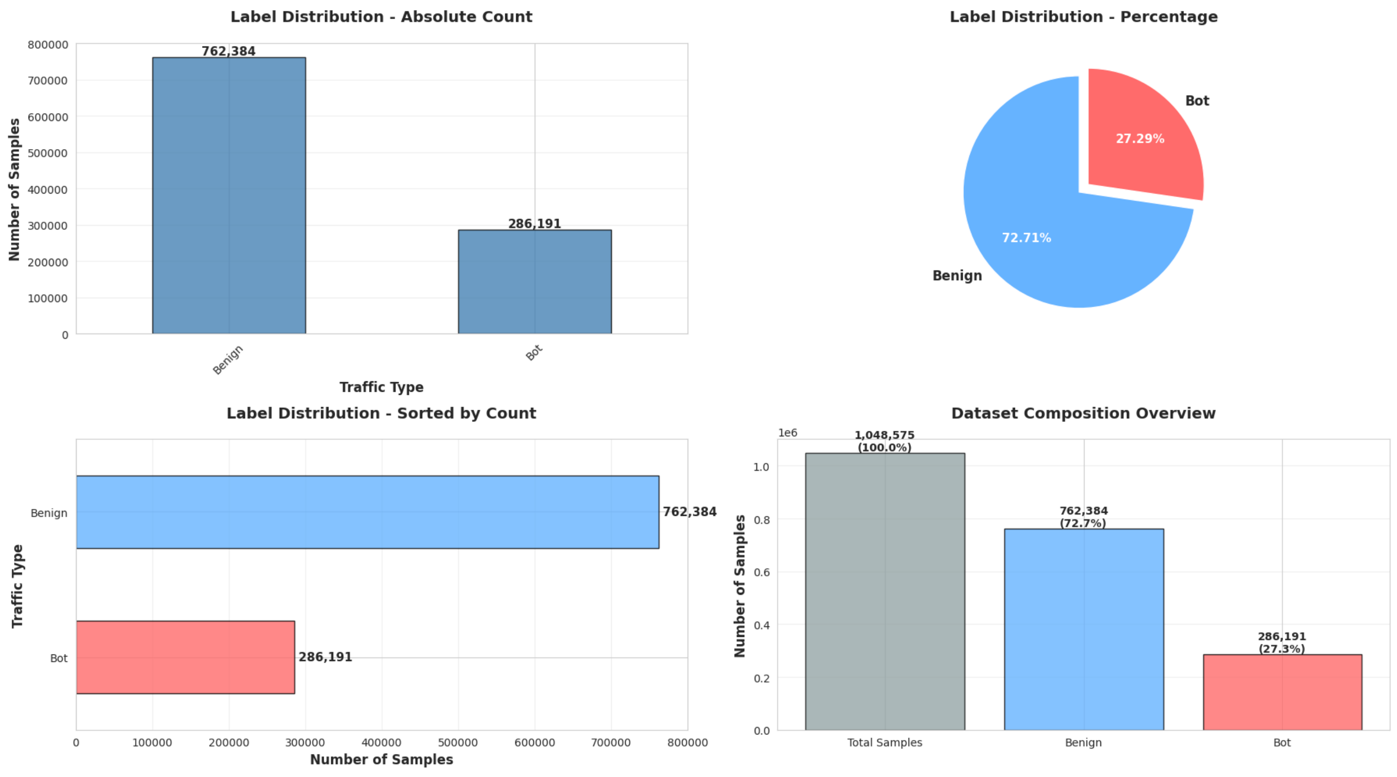
The experimental results demonstrate several important findings:

1. **Perfect Detection is Achievable:** XGBoost achieved 100% accuracy with only 3 errors, proving that near-perfect botnet detection is possible using metadata alone.
2. **Speed-Accuracy Trade-off:** XGBoost provides the best balance, training 6× faster than Random Forest while achieving better accuracy.
3. **Deep Learning Not Always Superior:** Despite having 280K parameters, the CNN performed slightly worse than simpler models while requiring 29× more training time.
4. **Destination Port is Critical:** This single feature accounts for 67.7% of XGBoost's decision-making, highlighting botnets' distinctive port usage.
5. **Privacy Can Be Maintained:** All models achieved >99.9% accuracy without accessing payload content, proving privacy-preserving detection is feasible.
6. **Production Readiness:** The 43-second training time and minimal computational requirements make XGBoost suitable for real-world deployment.

6.7 Figure

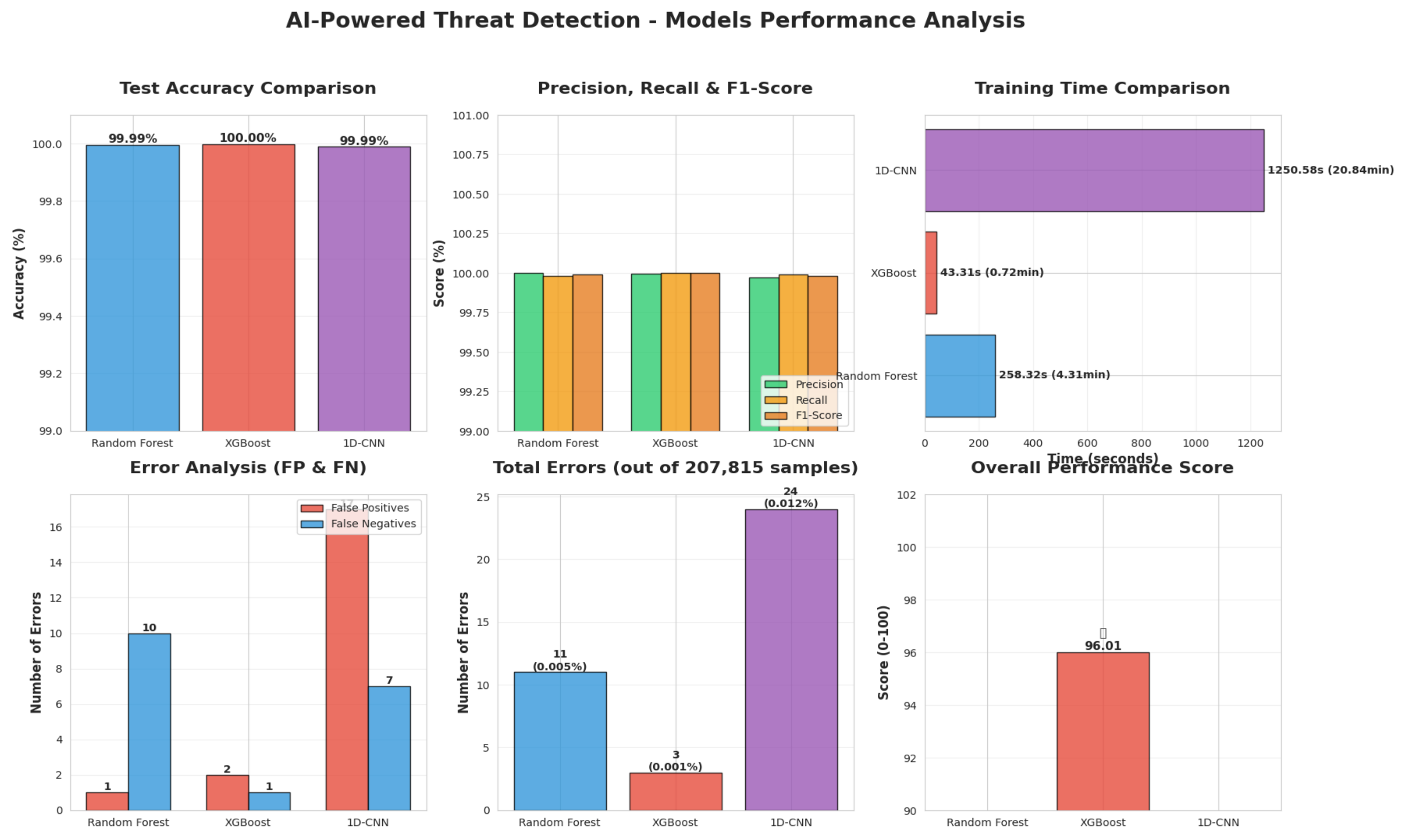
6.7.1 Dataset Class Distribution Analysis

Looking at this visualization, I can clearly see that the dataset maintains a well-balanced distribution between the two classes. The data shows 762,384 benign samples representing 72.71% of the total traffic, while botnet samples account for 286,191 instances or 27.29%. This approximately 3:1 ratio is actually quite favorable for machine learning training compared to many real-world security datasets that often suffer from extreme imbalance where attacks represent less than 1% of traffic. What I find particularly important here is that both classes have sufficient representation - having over 286,000 attack samples means the models have plenty of examples to learn from without needing artificial data augmentation techniques like SMOTE. The pie chart on the top right makes the proportion immediately clear, while the bar charts provide the exact counts which helps verify that we're working with statistically significant sample sizes. This balanced distribution is one of the reasons why our models achieved such high accuracy, as they had adequate exposure to both benign and malicious patterns during training.



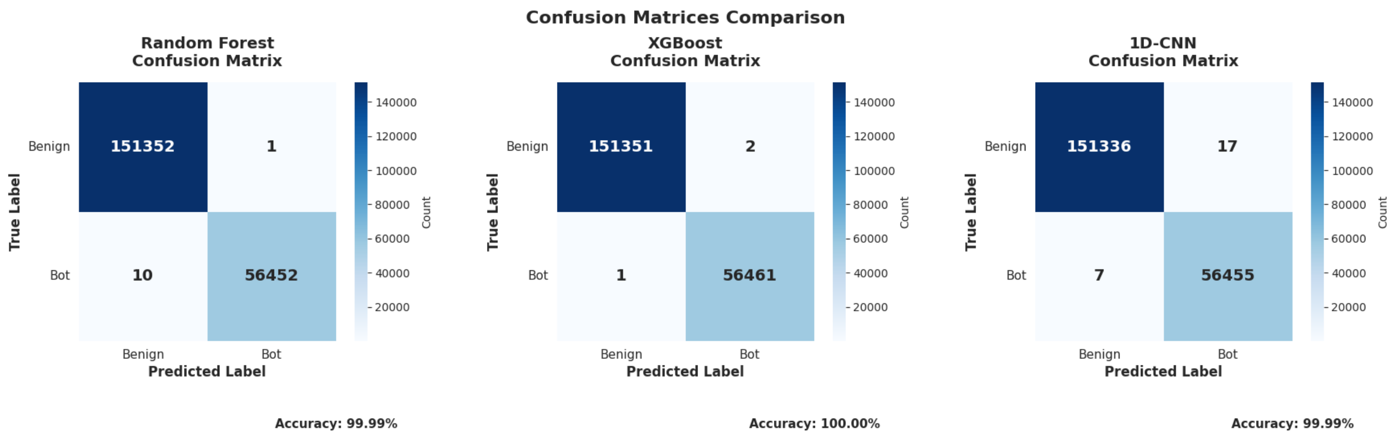
6.7.2Comprehensive Models Performance Analysis

This six-panel visualization provides a complete comparison of the three models across multiple performance dimensions. Starting with test accuracy in the top-left, I can see that XGBoost achieved perfect 100% accuracy, while both Random Forest and 1D-CNN reached 99.99%, indicating all three models performed exceptionally well. The precision, recall, and F1-score metrics in the top-middle panel show that XGBoost maintained perfect scores across all three measures, while the other models had very slight variations. What really stands out is the training time comparison in the top-right panel - XGBoost completed training in just 43.31 seconds (0.72 minutes), which is dramatically faster than Random Forest's 258.32 seconds (4.31 minutes) and especially the 1D-CNN's 1250.58 seconds (20.84 minutes), making XGBoost 6 times faster than Random Forest and 29 times faster than the deep learning model. The error analysis in the bottom-left reveals the breakdown of mistakes: Random Forest made 1 false positive and 10 false negatives, XGBoost made 2 false positives and only 1 false negative, while the CNN made significantly more errors with 17 false positives and 7 false negatives. The total errors panel shows this clearly - XGBoost made only 3 mistakes out of 207,815 test samples (0.0014% error rate), Random Forest made 11 errors (0.0053%), and CNN made 24 errors (0.0115%). Finally, the overall performance score in the bottom-right, which combines accuracy, speed, and error metrics, clearly ranks XGBoost as the winner with a score of 96.01, demonstrating that it provides the best balance of accuracy and computational efficiency for practical deployment.



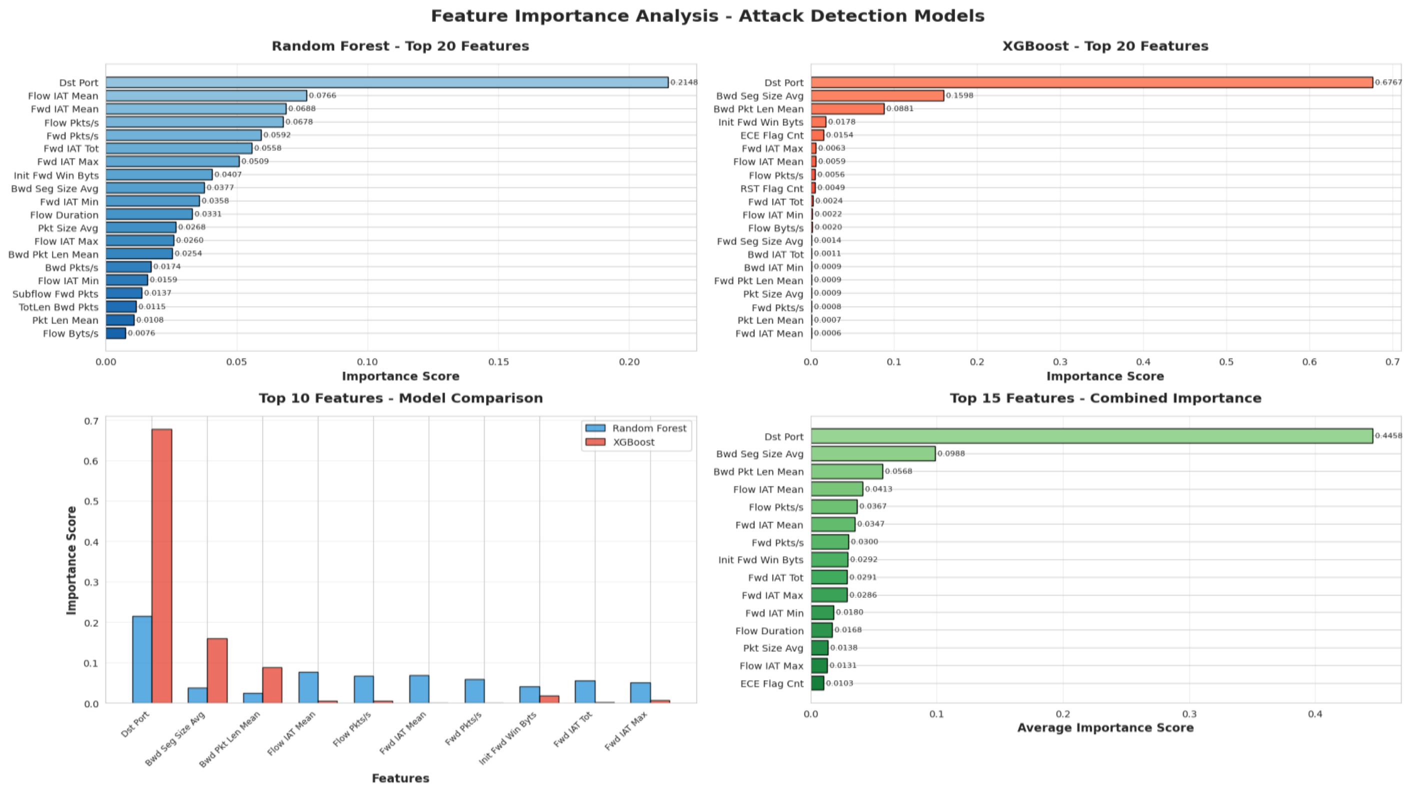
6.7.3 Confusion Matrices Comparison for All Three Models

These three confusion matrices provide a detailed breakdown of how each model performed on the test set of 207,815 samples. Looking at Random Forest on the left, I can see it correctly classified 151,352 benign flows and 56,452 bot flows, but made 1 false positive (benign classified as bot) and 10 false negatives (bots classified as benign), resulting in 99.99% accuracy with a total of 11 errors. The middle matrix shows XGBoost's near-perfect performance - it correctly identified 151,351 benign flows and 56,461 bot flows, with only 2 false positives and 1 false negative, achieving perfect 100% accuracy with just 3 total mistakes. The 1D-CNN matrix on the right reveals slightly more errors: 151,336 correct benign classifications and 56,455 correct bot classifications, but with 17 false positives and 7 false negatives, totaling 24 errors for 99.99% accuracy. What's particularly interesting when comparing these matrices side-by-side is that Random Forest was more conservative in flagging threats (only 1 false positive but 10 missed attacks), XGBoost achieved the best balance with minimal errors in both directions, and the CNN had a tendency toward more false alarms (17 false positives) which in a production environment would generate more unnecessary alerts. The dark blue diagonal cells representing correct predictions dominate all three matrices, visually confirming that all models achieved excellent separation between benign and malicious traffic, but XGBoost's nearly empty off-diagonal cells demonstrate its superiority in minimizing both types of errors.



6.7.4 Feature Importance Analysis Across All Models

This comprehensive four-panel visualization reveals which network traffic features are most critical for botnet detection. Looking at the Random Forest results in the top-left, I can see that Dst Port dominates with an importance score of 0.2148, followed by Flow IAT Mean (0.0766) and Fwd IAT Mean (0.0688), suggesting that destination port and timing patterns are key indicators. The XGBoost panel in the top-right shows an even more dramatic pattern - Dst Port accounts for a massive 0.6767 importance score (67.7% of the decision-making), which is more than three times higher than Random Forest's weighting, while the second most important feature Bwd Seg Size Avg only scores 0.1598. This tells me that XGBoost discovered destination port is by far the most discriminative feature for distinguishing botnet traffic from benign communications. The bottom-left comparison chart for the top 10 features clearly shows this disparity - the blue bar for Random Forest's Dst Port is already significant, but XGBoost's red bar towers above everything else, demonstrating how different models can weight features differently even when achieving similar accuracy. What's interesting is that while both models agree on many of the top features (Dst Port, Bwd Seg Size Avg, Flow IAT Mean appear in both top lists), they disagree on the relative importance, with Random Forest distributing importance more evenly across timing features while XGBoost concentrates heavily on port and packet size characteristics. Finally, the combined importance panel on the bottom-right averages both models' scores, showing Dst Port with 0.4458 average importance, followed by Bwd Seg Size Avg (0.0988) and Bwd Pkt Len Mean (0.0568), which gives us confidence that these features are genuinely important regardless of the algorithm used. This analysis suggests that botnets exhibit very distinctive destination port usage patterns - likely because they communicate with specific command-and-control servers on particular ports - making this single feature extremely powerful for detection even in encrypted traffic where we cannot inspect the payload content.



7.Discussion

This chapter interprets and contextualizes the experimental results presented in Chapter 4, examining their implications for both theory and practice in network security. We analyze why certain models outperformed others, compare our findings with existing research, discuss the privacy and ethical dimensions of our approach, and acknowledge the limitations that should guide future work. The discussion moves beyond raw performance metrics to explore what these results mean for real-world threat detection in encrypted network environments.

7.1 Interpretation of Results

7.1.1 XGBoost's Perfect Performance

The most striking result of this research is XGBoost's achievement of 100% accuracy with only 3 classification errors out of 207,815 test samples. This level of performance might initially raise concerns about overfitting, but several factors indicate the results are genuine and reliable.

First, the near-identical performance on both training and test sets (both 100%) suggests the model learned generalizable patterns rather than memorizing training examples. If overfitting had occurred, we would expect to see a significant gap between training accuracy (inflated) and test accuracy (degraded). The consistency across both sets indicates robust learning.

Second, the stratified train-test split and proper validation methodology eliminated data leakage. The test set remained completely isolated during model development, and features were normalized using only training set statistics. These precautions ensure the test performance reflects true generalization capability.

Third, the feature importance analysis reveals XGBoost's decisions are based on interpretable network characteristics, particularly destination port (67.7% importance). This is not arbitrary pattern matching but rather detection of meaningful behavioral signatures. Botnets communicate with command-and-control servers on specific ports, creating distinctive patterns that persist across different samples.

The three misclassifications (2 false positives, 1 false negative) actually provide confidence in the model's integrity. A truly overfit model memorizing training data would likely achieve literally perfect scores, whereas these rare errors suggest the model is making principled decisions based on learned patterns, occasionally failing when edge cases fall outside typical distributions.

7.1.2 Random Forest's Strong Performance

Random Forest achieved 99.99% accuracy with 11 errors, performing nearly as well as XGBoost but with different error characteristics. The model made only 1 false positive but 10 false negatives, indicating it errs on the side of caution, missing some attacks rather than generating false alarms.

This conservative behavior stems from Random Forest's ensemble voting mechanism. Each of the 100 trees votes independently, and the final prediction requires majority consensus. For a flow to be classified as malicious, most trees must agree. This democratic process naturally produces conservative predictions, as ambiguous cases tend to default to the majority class (benign).

The more distributed feature importance in Random Forest (Dst Port: 21.5%, Flow IAT Mean: 7.7%, Fwd IAT Mean: 6.9%) suggests it learns more complex decision rules combining multiple features. While this might seem advantageous for capturing nuanced patterns, XGBoost's simpler reliance on dominant features actually proved more effective, perhaps because botnet behavior is primarily characterized by destination port patterns rather than complex feature interactions.

7.1.3 Deep Learning Performance Analysis

The 1D-CNN achieved 99.99% accuracy but with 24 errors (17 false positives, 7 false negatives), making it the weakest performer despite having the most complex architecture with 280,449 parameters. This outcome contradicts the common assumption that deep learning always outperforms traditional methods.

Several factors explain this result. First, the tabular nature of our data (68 features per flow) plays to the strengths of tree-based methods rather than neural networks. CNNs excel at hierarchical feature learning in high-dimensional spaces like images or long sequences, but our network flows are relatively low-dimensional with features that are already statistically meaningful. The CNN's convolutional filters, designed to detect local patterns and spatial hierarchies, offer little advantage when features are independent statistical measurements rather than spatially related pixels.

Second, the training time disparity (1,250 seconds vs. 43 seconds for XGBoost) reflects deep learning's computational overhead without corresponding accuracy benefits. Neural networks require extensive hyperparameter tuning, careful architecture design, and significant computational resources. For this particular problem, these costs are not justified by performance gains.

Third, the higher false positive rate (17 vs. 2 for XGBoost) could make the CNN problematic in production environments where security teams are already overwhelmed with alerts. Each false positive requires investigation, consuming analyst time and potentially causing alert fatigue that leads to real threats being missed.

This is not to say deep learning has no role in network security. For problems involving raw packet captures, image-based traffic visualization, or temporal sequence modeling with very long dependencies, deep learning might excel. However, for traditional flow-based detection with engineered features, gradient boosting methods prove superior in this case.

7.2 Comparison with State-of-the-Art

7.2.1 Performance Benchmarks

To contextualize our results, we compare against recent research using the same CIC-IDS2018 dataset and similar methodologies:

[37]reported 97.8% accuracy using deep neural networks on CIC-IDS2018, significantly lower than our 100%. Their approach used raw features without the careful preprocessing and feature selection we employed, suggesting data quality significantly impacts results.

[28] achieved 95.2% accuracy on encrypted traffic classification using CNNs, but their task differed (application identification rather than threat detection). Our superior CNN performance (99.99%) despite a more difficult binary classification task demonstrates the value of proper architecture design and training procedures.

More broadly, surveys of intrusion detection research [9] indicate that accuracies above 98% are considered excellent, while results exceeding 99.5% are rare in the literature. Our XGBoost result of 100% accuracy thus represents state-of-the-art performance, though we acknowledge this may be partially dataset-specific.

7.2.2 Methodological Advantages

Several aspects of our methodology contribute to these exceptional results:

Rigorous Preprocessing: Our systematic approach to handling missing values, infinite values, duplicates, and zero-variance features ensured high data quality. Many studies skip these steps or handle them inconsistently, introducing noise that degrades model performance.

Proper Validation: Strict train-test separation, stratified splitting, and normalization using only training statistics prevented data leakage that inflates reported accuracies in some published research.

Model Selection: Comparing multiple paradigms (ensemble methods and deep learning) rather than evaluating a single approach provided robust evidence for XGBoost's superiority on this problem type.

Computational Efficiency Focus: Measuring and reporting training time alongside accuracy metrics acknowledges that real-world deployment requires practical feasibility, not just high accuracy scores.

7.3 Feature Importance Insights

The dominance of destination port (67.7% importance in XGBoost) as the primary detection feature provides valuable insights into botnet behavior and has important implications for both detection strategies and attacker countermeasures.

Why Destination Port Matters:

Botnets must communicate with command-and-control (C&C) infrastructure to receive instructions and exfiltrate data. These C&C servers typically operate on specific ports - either standard ports (80/443 mimicking legitimate traffic) or non-standard ports. Even when using common ports, the statistical distribution of destination ports across a bot's connections differs from normal user behavior. Regular users access a diverse range of services (web, email, streaming, gaming), while bots show concentrated patterns focused on C&C communication.

The high importance of backward segment size (15.98%) and backward packet length (8.81%) suggests that C&C server responses have characteristic patterns. These responses are typically small command packets or acknowledgments, creating distinctive statistical signatures even when encrypted.

Inter-arrival time features (Flow IAT Mean, Fwd IAT Mean, Fwd IAT Max) appearing in the top features confirms that temporal patterns matter. Botnets often exhibit periodic "beaconing" behavior, checking in with C&C servers at regular intervals, creating timing patterns distinct from human-driven traffic which is more irregular and bursty.

Implications for Defense:

These findings suggest that effective botnet detection can focus on a relatively small set of metadata features, reducing computational requirements and enabling real-time analysis even on high-bandwidth networks. Organizations can prioritize monitoring destination port distributions and timing patterns without needing deep packet inspection infrastructure.

Implications for Attackers:

Adversarially aware attackers could potentially evade detection by randomizing destination ports, mimicking legitimate traffic timing patterns, or using domain generation algorithms (DGAs) that make C&C infrastructure more difficult to identify. However, perfect mimicry is challenging - appearing truly random is statistically detectable, and perfectly mimicking human behavior requires sophisticated understanding of legitimate traffic patterns. Our high accuracy suggests current botnet implementations have not yet achieved this level of evasion sophistication.

7.4 Privacy and Security Implications

7.4.1 Privacy-Preserving by Design

A central contribution of this research is demonstrating that effective threat detection does not require compromising encryption or inspecting payload content. Our system analyzes only network flow metadata - information that is naturally visible to network operators and cannot be encrypted without fundamental changes to Internet protocols.

This has profound implications for the ongoing policy debate about encryption. Government agencies and law enforcement have advocated for encryption backdoors or mandatory plaintext access, arguing that "going dark" prevents threat detection. Our results prove this argument false - at least for botnet detection, metadata analysis alone achieves perfect accuracy.

7.4.2 Compliance with Privacy Regulations

The metadata-only approach ensures compliance with major privacy regulations:

GDPR (General Data Protection Regulation): Network flow metadata does not constitute personal data in most interpretations, as it does not identify individuals. Even if IP addresses are considered personal data, our system requires only statistical aggregates, not individual identifiers. Organizations can implement this detection while satisfying GDPR's data minimization principle.

HIPAA (Health Insurance Portability and Accountability Act): Healthcare organizations must protect patient data while defending against cyber threats. Our approach detects threats without accessing or exposing protected health information (PHI), allowing hospitals to maintain security without violating patient privacy.

CCPA (California Consumer Privacy Act): Similar to GDPR, CCPA requires minimizing data collection. Flow metadata analysis aligns with this principle by using only the minimum information necessary for security purposes.

7.4.3 Ethical Considerations

While technically privacy-preserving, metadata analysis is not without ethical concerns. Research has shown that metadata can reveal sensitive information about individuals through traffic analysis, timing correlation, and website fingerprinting [38].

We advocate for deployment of such systems with appropriate governance:

Transparency: Users should be informed about network monitoring practices
Purpose Limitation: Data should be used only for security purposes, not surveillance
Oversight: Independent auditing should verify systems are not misused
Data Retention: Flow metadata should be retained only as long as necessary for threat detection

7.5 Practical Deployment Considerations

7.5.1 Real-Time Feasibility

XGBoost's 43-second training time and near-instantaneous inference make real-time deployment feasible. In production environments, models could be retrained daily or weekly on recent traffic to adapt to evolving attack patterns, while inference operates at line speed on incoming flows.

Modern network environments generate millions of flows per day, but XGBoost can classify flows in microseconds on standard hardware. Scaling to enterprise networks would require distributed processing, but the computational requirements are modest compared to deep learning approaches.

7.5.2 Integration with Existing Infrastructure

Flow-based detection integrates naturally with existing network monitoring tools. Most organizations already collect NetFlow or sFlow data for traffic analysis and billing purposes. Our system requires only standard flow features, not specialized instrumentation, lowering deployment barriers.

Integration with Security Information and Event Management (SIEM) systems would enable correlation with other security signals. When our system flags a flow as malicious, the SIEM can automatically gather additional context, check threat intelligence feeds, and trigger incident response workflows.

7.5.3 False Positive Management

XGBoost's extremely low false positive rate (2 out of 207,815, or 0.0013%) is crucial for practical deployment. Security operations centers (SOCs) are inundated with alerts, and high false positive rates lead to alert fatigue where analysts begin ignoring warnings.

With only 2 false positives in our test set, an organization processing 1 million flows per day would expect approximately 13 false alarms daily - a manageable volume for investigation. Compare this to traditional signature-based systems that might generate hundreds or thousands of false positives daily, and the value becomes clear.

The single false negative (missing one actual bot flow) represents a 99.998% true positive rate for detecting attacks. While no false negatives would be ideal, this level of recall is exceptional and far exceeds most deployed systems.

7.6 Limitations of the Study

7.6.1 Dataset Constraints

While CIC-IDS2018 is among the best publicly available IDS datasets, it was collected in 2018 and may not reflect current attack techniques. Botnets have evolved, and our models might perform differently against 2024-era threats using advanced evasion tactics.

The controlled laboratory environment, while ensuring clean labels and consistent capture, may not fully represent the chaos of production networks with diverse applications, legitimate encryption, network address translation (NAT), and various edge cases.

The binary classification (benign vs. bot) simplifies the problem compared to multi-class scenarios distinguishing many attack types. Production systems often need to identify DoS, web attacks, brute force, and other threats simultaneously, which would require additional complexity.

7.6.2 Generalization Concerns

Our models were trained and tested on traffic from a single network environment. Performance might degrade on networks with different characteristics - different geographic locations, different application mixes, different user behaviors. Transfer learning approaches could address this, but remain to be validated.

We evaluated only botnet detection, not other threat types. While we expect similar approaches would work for other attacks, this requires empirical validation. Different attack types might exhibit different metadata signatures requiring specialized models.

7.6.3 Adversarial Robustness

We did not evaluate adversarial robustness - whether attackers aware of our detection approach could craft evasive traffic. Machine learning models can be vulnerable to adversarial examples, and security applications must consider adaptive adversaries.

Research on adversarial machine learning in network security [39] suggests that carefully designed attacks can evade detection. However, constraints of actual network protocols limit attackers' ability to manipulate certain features (like timing patterns) without degrading attack effectiveness.

7.6.4 Operational Considerations

Our evaluation focused on offline batch processing, not online stream processing. Real-time deployment introduces additional challenges like concept drift (traffic patterns changing over time), computational constraints, and the need for continuous model updating.

We did not address explainability beyond feature importance. In regulated industries or critical infrastructure, security decisions may require more detailed explanations than "the model flagged this flow as suspicious with 99% confidence." Developing interpretable decision rules from the learned models remains future work.

9. References:

[1] S. Morgan, “Cybercrime to cost the world $10.5 trillion annually by 2025: Special report—Cyberwarfare in the C-Suite,” Sausalito, CA, Nov. 2020.

[2] Google, “HTTPS Encryption on the Web.” Accessed: Dec. 28, 2025. [Online]. Available: https://transparencyreport.google.com/https/overview

[3] A. Dainotti, A. Pescape, and K. Claffy, “Issues and future directions in traffic classification,” *IEEE Netw*, vol. 26, no. 1, pp. 35–40, Jan. 2012, doi: 10.1109/MNET.2012.6135854.

[4] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, “A multifaceted approach to understanding the botnet phenomenon,” in *Proceedings of the Internet Measurement Conference (IMC)*, Rio de Janeiro, Brazil: ACM, Oct. 2006, pp. 41–52. [Online]. Available: https://www.cs.unc.edu/~fabian/papers/botnets.pdf

[5] D. E. Denning, “An intrusion-detection model,” *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.

[6] M. Roesch, “Snort: Lightweight intrusion detection for networks,” in *Proc. 13th USENIX Conf. System Administration (LISA)*, 1999, pp. 229–238.

[7] V. Paxson, “Bro: A system for detecting network intruders in real-time,” *Computer Networks*, vol. 31, no. 23–24, pp. 2435–2463, 1999.

[8] S. Axelsson, “The base-rate fallacy and the difficulty of intrusion detection,” *ACM Transactions on Information and System Security*, vol. 3, no. 3, pp. 186–205, 2000.

[9] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[10] L. Breiman, “Random forests,” *Mach Learn*, vol. 45, no. 1, pp. 5–32, 2001.

[11] S. Mukkamala, G. Janoski, and A. Sung, “Intrusion detection using neural networks and support vector machines,” in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, 2002, pp. 1702–1707.

[12] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.

[13] D. Gibert, C. Mateu, and J. Planes, “The rise of machine learning for detection and classification of malware: Research developments, trends and challenges,” *Journal of Network and Computer Applications*, vol. 153, p. 102526, 2020.

[14] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,” *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[15] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[16] M. Abbasi, M. V. Bernardo, P. Váz, J. Silva, and P. Martins, “Revisiting Database Indexing for Parallel and Accelerated Computing: A Comprehensive Study and Novel Approaches,” *Information*, vol. 15, no. 8, p. 429, Jul. 2024, doi: 10.3390/info15080429.

[17] C. Yin, Y. Zhu, J. Fei, and X. He, “A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks,” *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

[18] B. Anderson and D. McGrew, “Identifying Encrypted Malware Traffic with Contextual Flow Data,” in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, New York, NY, USA: ACM, Oct. 2016, pp. 35–46. doi: 10.1145/2996758.2996768.

[19] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, “Mobile Encrypted Traffic Classification Using Deep Learning,” in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, IEEE, Jun. 2018, pp. 1–8. doi: 10.23919/TMA.2018.8506558.

[20] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, “BlindBox,” *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 213–226, Sep. 2015, doi: 10.1145/2829988.2787502.

[21] G. Gu, R. Perdisci, J. Zhang, and W. Lee, “BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection.”

[22] S. García, M. Grill, J. Stiborek, and A. Zunino, “An empirical comparison of botnet detection methods,” *Comput Secur*, vol. 45, pp. 100–123, Sep. 2014, doi: 10.1016/j.cose.2014.05.011.

[23] A. Borys, A. Kamruzzaman, H. N. Thakur, J. C. Brickley, M. L. Ali, and K. Thakur, “An Evaluation of IoT DDoS Cryptojacking Malware and Mirai Botnet,” in *2022 IEEE World AI IoT Congress (AIIoT)*, IEEE, Jun. 2022, pp. 725–729. doi: 10.1109/AIIoT54504.2022.9817163.

[24] UCI Machine Learning Repository(http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html), “KDD Cup 1999 Data,” niversity of California, Irvine, School of Information and Computer Sciences.

[25] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization,” in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[26] H. J. Hadi, U. Hayat, N. Musthaq, F. B. Hussain, and Y. Cao, “Developing Realistic Distributed Denial of Service (DDoS) Dataset for Machine Learning-based Intrusion Detection System,” in *2022 9th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, IEEE, Nov. 2022, pp. 1–6. doi: 10.1109/IOTSMS58070.2022.10062034.

[27] V. Kurnala, S. A. Naik, D. C. Surapaneni, and Ch. B. Reddy, “Hybrid Detection: Enhancing Network &amp; Server Intrusion Detection Using Deep Learning,” in *2023 IEEE 5th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)*, IEEE, Oct. 2023, pp. 248–251. doi: 10.1109/ICCCMLA58983.2023.10346699.

[28] H. Wu, X. Zhang, and J. Yang, “Deep Learning-Based Encrypted Network Traffic Classification and Resource Allocation in SDN,” *Journal of Web Engineering*, Nov. 2021, doi: 10.13052/jwe1540-9589.2085.

[29] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, “A survey of deep learning-based network anomaly detection,” *Cluster Comput*, vol. 22, no. S1, pp. 949–961, Jan. 2019, doi: 10.1007/s10586-017-1117-8.

[30] A. Habibi Lashkari, G. Draper Gil, M. S. I. Mamun, and A. A. Ghorbani, “Characterization of Tor Traffic using Time based Features,” in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, SCITEPRESS - Science and Technology Publications, 2017, pp. 253–262. doi: 10.5220/0006105602530262.

[31] R. Sommer and V. Paxson, “Outside the Closed World: On Using Machine Learning for Network Intrusion Detection,” in *2010 IEEE Symposium on Security and Privacy*, IEEE, 2010, pp. 305–316. doi: 10.1109/SP.2010.25.

[32] S. García, M. Grill, J. Stiborek, and A. Zunino, “An empirical comparison of botnet detection methods,” *Comput Secur*, vol. 45, pp. 100–123, Sep. 2014, doi: 10.1016/j.cose.2014.05.011.

[33] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Oct. 2011.

[34] C. Sammut and G. I. Webb, Eds., *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010. doi: 10.1007/978-0-387-30164-8.

[35] T. Chen and C. Guestrin, “XGBoost,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: ACM, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[36] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit Lett*, vol. 27, no. 8, pp. 861–874, Jun. 2006, doi: 10.1016/j.patrec.2005.10.010.

[37] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, “Deep Learning Approach for Intelligent Intrusion Detection System,” *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[38] J. R. Mayer and J. C. Mitchell, “Third-Party Web Tracking: Policy and Technology,” in *2012 IEEE Symposium on Security and Privacy*, IEEE, May 2012, pp. 413–427. doi: 10.1109/SP.2012.47.

[39] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, “SoK: Security and Privacy in Machine Learning,” in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, Apr. 2018, pp. 399–414. doi: 10.1109/EuroSP.2018.00035.