An MQTT Application-Layer Traffic Analyzer for Interpretable Flow-Level Intrusion Detection and Zero-Day Threat Identification in IoT Environment Using TabNet

Arefeh KouhiRonaghi

A Thesis Submitted to the Faculty of Graduate Studies
In Partial Fulfillment of the Requirements
for the Degree of Master of Arts

Graduate Program in Information and System Technology

York University
Toronto, Ontario

August 2025

Abstract

Message Queuing Telemetry Transport (MQTT) is widely used in IoT systems; however, its lightweight design makes it vulnerable to various cyberattacks. This research reviews existing intrusion detection methods for MQTT and shows their limitations in detecting new and complex threats. This study presents a comprehensive intrusion detection framework that uses raw PCAP data and employs flow-based behavioral analysis to detect both known and novel attacks. We present MQTTFlowLyzer, a protocol-aware analyzer designed to extract detailed MQTT flow features and generate an augmented dataset, BCCC-MQTT-IDS-2025, that captures realistic and diverse attack scenarios. The extracted features train a TabNet-based learning model capable of integrated feature selection, classification, and confidence-based detection of zero-day threats. Our approach highlights the behavioral uniqueness of each attack class and uses attention-driven interpretability for in-depth analysis. Experimental results demonstrate that the model effectively detects attacks while maintaining high performance across other categories. The system successfully flags previously unseen traffic by profiling class-specific behaviors and incorporating confidence thresholds. These results demonstrate the potential of flow-based, interpretable learning for real-time and resilient MQTT intrusion detection.

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Prof. Arash Habibi Lashkari, for his invaluable guidance and support throughout this journey. I would also like to thank my family for their unwavering encouragement and constant support. Lastly, I extend my appreciation to all my friends for their constant encouragement and companionship along the way.

Contents

Page

Abstract ii

Page

Acknowledgement iii

Contents iv

List of Tables vi

List of Figures vii

1 Introduction 1

2 Literature Review 3

2.1 MQTT in IoT communications . . . 3

2.2 Previous Works . . . 5

2.2.1 Signature-Based Detection Models . . . 5

2.2.2 Classical Anomaly Detection Models . . . 6

2.2.3 Supervised Anomaly Detection Models: . . . 10

2.2.4 Unsupervised Anomaly Detection Models: . . . 12

2.2.5 Semi-Supervised Anomaly Detection Models: . . . 14

2.2.6 Federated Learning-Based IDS Models: . . . 18

2.2.7 Blockchain-Based Security Solutions . . . 20

2.3 Synthesis . . . 22

3 Proposed Model 24

3.1 MQTTFlowLyzer: MQTT Layer Traffic Feature Extractor . . . 24

3.1.1 Flow Creation . . . 25

3.1.2 Feature Extraction . . . 26

3.2 Data Preprocessing . . . 33

3.2.1 Handling Missing Values . . . 33

3.2.2 Addressing Class Imbalances . . . 33

3.2.3 Categorical Feature Processing . . . 33

3.2.4 Feature Removal . . . 34

3.2.5 Feature-Label Separation and Label Encoding . . . 34

3.2.6 Data Splitting . . . 34

3.3 TabNet-Based Learning Framework for IoT Intrusion Detection . . . 34

3.3.1 Selection Strategy . . . 34

3.3.2 Architecture Overview . . . 35
3.3.3 Feature and Attentive Transformers . . . 36
3.3.4 Feature Selection . . . 37
3.3.5 Feature-Level Interpretability . . . 38
3.3.6 Classification Method . . . 40
3.4 Confidence-Based Zero-Day Detection . . . 40
3.5 Hyperparameter Tuning . . . 42

4 Experiments and Results **43**
4.1 Available Datasets . . . 43
4.1.1 Datasets . . . 43
4.1.2 Dataset Evaluation Criteria . . . 45
4.1.3 Dataset Scoring and Results . . . 46
4.2 New Augmented Dataset . . . 48
4.3 Performance Results . . . 49
4.4 Top-Ranked Features . . . 52
4.5 Class-wise Behavioral Profiling . . . 53
4.6 Zero-Day Detection Scenario . . . 54

5 Analysis and Discussion **55**
5.1 Interpretation of the Most Important Features . . . 56
5.2 Zero-Day Attack Analysis Across Baseline Models . . . 57
5.3 Class-wise Feature Attribution and Behavioral Signatures . . . 58
5.4 Computational Efficiency . . . 61
5.5 Feature Dimensionality and Computational Cost Analysis . . . 62
5.6 Detection Metrics . . . 63
5.7 Comparative Performance Evaluation . . . 64

6 Conclusion and Future Works **66**

Bibliography **67**

Vita **72**

List of Tables

1 MQTT Control Packet types 4
2 MQTTFlowLyzer Features 28
3 Summary of Feature Categories Extracted from MQTT Flows 33
4 Feature Selection Comparison 35
5 Example Calculation of the Weighted Score for MQTT-IoT-IDS2020 47
6 Evaluation of MQTT Attack Datasets Based on Key Criteria 47
7 BCCC-MQTT-IDS-2025 Dataset Overview 49
8 Label distribution, after cleaning and downsampling 50
9 TabNet Hyperparameters - Default Ranges vs. Optimized Values (Optuna) 51
10 Classification Results of the Proposed Model 51
11 Top 30 Most Important Features Identified by the TabNet Model 52
12 Computational analysis of MQTTFlowLyzer feature extraction on the MQTTset dataset . . . 62

List of Figures

1 Example of MQTT publish/subscribe communication in an IoT environment 3
2 Proposed Bayesian Rule Learning-based IDS by [1] 6
3 Taxonomy of MQTT IDS methods 7
4 Proposed Secure-MQTT fuzzy logic-based IDS by [2] 8
5 Process tree-based anomaly detection approach adapted from [3]. 9
6 DNN-based IDS framework for MQTT intrusion detection, adapted from [4]. 11
7 Proposed TNN-IDS model showing the multi-head attention, convolutional, and dense layers, adapted from [5]. 12
8 Architecture of the proposed ECSSA-LightGBM-based IDS [6] 13
9 Proposed architecture of EdgeIDS for anomaly detection in IoT environments [7] 14
10 Architecture of the proposed GAN-AE model for unsupervised anomaly detection [8] 15
11 Online-semisupervised anomaly detection algorithm proposed by [9]. The model iteratively selects and labels the most informative samples from the unlabeled pool to improve detection performance. 16
12 MS-Res module structure within the SS-Deep-ID model [10]. The module captures multi-scale temporal dependencies via residual connections and dilated convolutions. 16
13 Anomaly detection framework configurations using ocGAN and bcGAN models [11]. (a)–(d): Binary-class settings; (e): Multiclass generation strategy. 17
14 The architecture of the federated deep learning-based IoT intrusion detection system [12]. . . 18
15 MV-FLID architecture for federated multi-view intrusion detection using Grey Wolves Optimizer and ensemble learning [13]. 19
16 Architecture of the GEMLIDS-MIoT federated learning-based intrusion detection system [14]. It enables secure and energy-efficient anomaly detection across distributed medical IoT gateways. 20
17 Sequence diagram of the proposed MQTT-Blockchain integration model [15]. It shows how sensor data is collected, validated, and recorded onto a blockchain ledger via MQTT message brokers acting as validator nodes. 21
18 Proposed blockchain-sharding-based MQTT security architecture, adapted from [16]. 22
19 End-to-end architecture of the proposed TabNet-based MQTT intrusion detection framework. 24
20 MQTTFlowLyzer constructs flows by grouping packets with the same source and destination IPs, ports, ClientID, and Topic. This hybrid flow definition combines transport-level identifiers with MQTT-specific semantics to enable accurate application-layer session reconstruction and feature extraction. 25
21 Hierarchical categorization of extracted features from MQTT traffic. 32
22 Feature transformer composed of shared and step-specific blocks using FC, BN, and GLU units. 37
23 Attentive transformer using prior scales and sparsemax to generate sparse feature masks. . . 38

24 Heatmap of class-wise local feature importance based on TabNet's attention mechanism. Brighter cells denote a higher influence of the corresponding feature for that class. 54
25 Confidence level of the TabNet model on the MQTT-BF zero-day attack. 55
26 Confidence levels of various ML and DL models on the zero-day MQTT-BF attack. 59
27 Feature Importance bar chart by Class. 60
28 Comparison of Confusion Matrices between (a) RF model (baseline) and (b) proposed MQT-TFlowLyzer–TabNet. 65

1 Introduction

The rise of MQTT as a lightweight messaging protocol has amplified its adoption across diverse IoT ecosystems, enabling low-overhead machine-to-machine communication. Its simple publish/subscribe model and efficient bandwidth usage make it highly suitable for constrained environments such as smart homes, smart agriculture, industrial automation, and health monitoring systems [17, 18, 19]. As a result, MQTT has become a key enabler for many modern IoT applications across domains like home automation [20], irrigation systems [21], and robotics [22].

However, this widespread use also increases the exposure of the protocol to cyber threats. The minimal built-in security features of MQTT, such as lack of authentication enforcement and default use of unencrypted communication, make it an attractive target for attackers [23, 24]. Traditional intrusion detection systems (IDS) often struggle to identify sophisticated or zero-day attacks within MQTT traffic, as they rely on packet-level inspection or opaque machine learning models [25]. Furthermore, recent studies have emphasized that building effective machine learning-based intrusion detection systems, especially for IoT environments, critically depends on the availability of realistic, diverse, and feature-rich datasets[26]. Thus, an interpretable, flow-based behavioral profiling approach is needed for MQTT-specific communication patterns.

This paper addresses the deficiencies of existing MQTT-based IDS solutions by introducing a novel methodology grounded in protocol-aware flow semantics. In the Literature review section, we reviewed prior research efforts, highlighting gaps such as insufficient granularity, reliance on TCP-based features, and lack of interpretability in deep learning frameworks. These gaps underscore the need for a more principled approach that captures behavioral context and provides transparent decision-making.

Then we propose a comprehensive learning framework that begins with MQTTFlowLyzer, a custom tool for extracting 404 enriched flow-level features from raw PCAP traces. These features reflect temporal, statistical, and MQTT-specific control attributes that characterize benign and malicious behavior. We then employ TabNet, a deep learning architecture that performs instance-wise feature selection and classification in a unified model. To extend our system's robustness, we incorporate a softmax-based confidence module for flagging zero-day threats and introduce a class-wise interpretability module to facilitate behavioral analysis and forensic profiling.

After that, we presented a thorough empirical evaluation using multiple benchmark datasets, including our custom-augmented BCCC-MQTT-IDS-2025. This dataset synthesizes diverse MQTT attack scenarios and supports realistic testing of intrusion detection methods. We evaluate classification accuracy, inference latency, interpretability outputs, and robustness to unknown threats.

We interpret the learned feature importance profiles and analyze our model's computational efficiency. We also demonstrate how flow-level representations enable high-resolution behavioral profiling and improve the detection of previously unseen (zero-day) threats, surpassing traditional models reliant on static packet attributes.

The main contribution is designing a deep learning-based intrusion detection model for MQTT-enabled IoT environments. The model uses TabNet to perform automatic feature selection and flow-level classification in a unified and interpretable manner. This allows the system to detect known and zero-day attacks while

offering insights into the behavior of each traffic instance.

The second contribution is the development of MQTTFlowLyzer, a protocol-aware traffic analyzer that extracts 404 flow-level features from raw PCAP files. These features capture MQTT-specific timing, sizing, and statistical behaviors essential for identifying attack patterns.

The third contribution is the creation of BCCC-MQTT-IDS-2025, an augmented dataset built from raw packet captures and enhanced with a wide range of realistic and diverse MQTT-based attacks, supporting comprehensive model evaluation.

The fourth contribution is implementing a zero-day attack detection mechanism that uses confidence scores to flag unfamiliar and out-of-distribution traffic patterns.

The fifth contribution is a class-wise interpretability module that explains the model's decision-making process and helps generate clear behavioral profiles for different traffic classes.

Finally, this paper also provides an up-to-date taxonomy of MQTT-based intrusion detection systems, which helps position our approach within the broader research landscape and highlights the gaps addressed by our framework. The remainder of the paper is structured as follows: Section 2 discusses related work; Section 3 outlines the methodology; Section 4 presents experimental results; Section 5 contains analysis and discussion; and Section 6 concludes the paper with directions for future research.

2 Literature Review

This chapter comprehensively reviews Intrusion Detection Systems (IDS) techniques for securing the MQTT protocol, categorizing recent notable studies by methodology. Fig. 3 illustrates an organized taxonomy of MQTT IDS methods, highlighting the approaches employed by researchers. Although some work has focused on enhancing MQTT communication efficiency and time sensitivity, such as the introduction of PrioMQTT [27] for low-latency messaging based on priority, relatively few studies have addressed security concerns at the application flow level. Previous research limitations are identified, highlighting areas for potential contributions, which are discussed in the synthesis section. This thesis addresses these gaps, proposing a traffic analyzer and a model as the optimal solution to enhance MQTT security.

2.1 MQTT in IoT communications

The Message Queuing Telemetry Transport (MQTT) is a lightweight, publish/subscribe-based communication protocol designed specifically for machine-to-machine (M2M) and Internet of Things (IoT) environments. MQTT operates at the application layer and enables efficient and asynchronous data exchange between devices using a centralized broker. It is particularly well-suited for resource-constrained environments due to its low bandwidth requirements, minimal overhead, and simplicity.

In MQTT's publish/subscribe architecture, data producers—publishers—send messages to specific topics. Data consumers—called subscribers—express interest in those topics. The MQTT broker acts as an intermediary that matches topics of incoming messages to subscriber interests and forwards messages accordingly. This approach decouples communication between devices in both time and space, meaning that publishers and subscribers do not need to be connected simultaneously or know each other's identity. Figure 1 illustrates this basic publish/subscribe mechanism.

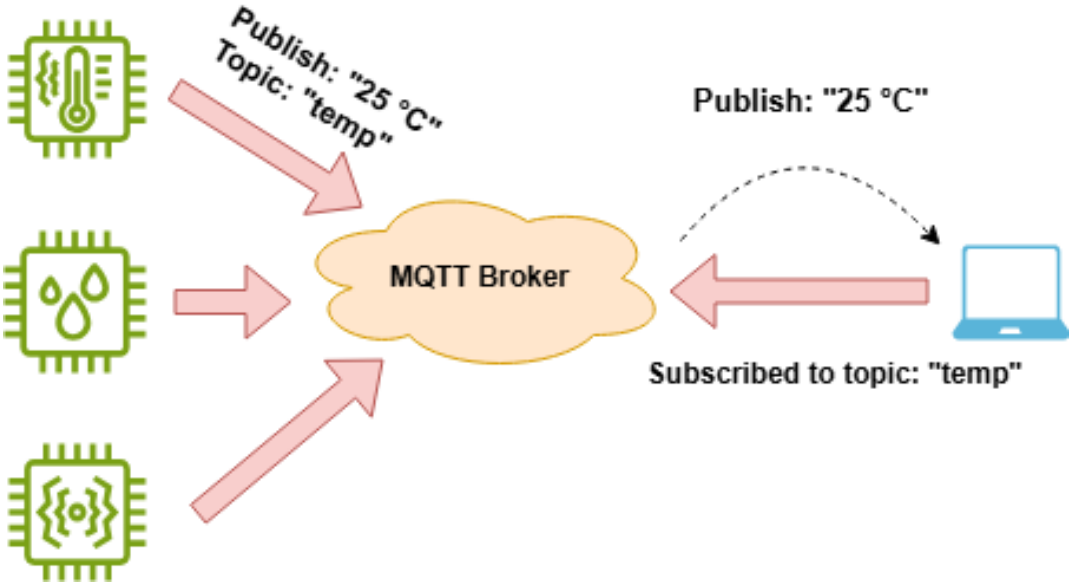

Figure 1: Example of MQTT publish/subscribe communication in an IoT environment

Name	Value	Direction of Flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Connection request
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment (QoS 1)
PUBREC	5	Client to Server or Server to Client	Publish received (QoS 2 delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (QoS 2 delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (QoS 2 delivery part 3)
SUBSCRIBE	8	Client to Server	Subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server or Server to Client	Disconnect notification
AUTH	15	Client to Server or Server to Client	Authentication exchange

Table 1: MQTT Control Packet types

One of the broker's primary functions is message filtering, which determines how messages are routed to subscribers. Publish/subscribe systems typically use either content-based or topic-based filtering. In content-based filtering, messages are delivered based on specific content attributes. MQTT, however, employs topic-based filtering, where subscribers receive only the messages whose topic matches their subscription.

MQTT defines several control packets to manage communication, each consisting of up to three components: a fixed header, a variable header, and an optional payload. These control packets enable key protocol operations and message routing. Table 1 lists the most common control packet types.

Table 1 provides a comprehensive summary of the MQTT control packet types defined in the MQTT v5.0 specification. Each packet type is associated with a specific function that enables communication between clients and the broker. The table lists the packet name, its corresponding numeric value, the direction of message flow (i.e., from client to server, server to client, or bidirectional), and a brief description of its role within the protocol. These packets manage various aspects of the MQTT session lifecycle, including connection setup, message publishing, subscription handling, and session termination. Understanding these control packets is essential for analyzing and implementing MQTT-based communication in IoT systems.

MQTT was built on the TCP/IP protocol to ensure reliable delivery. It supports an event-driven model consisting of three primary operations: connect, publish, and subscribe, each with corresponding acknowledgments to confirm success. This design provides a robust mechanism for asynchronous communication across distributed devices.

According to the MQTT v5.0 specification [28], each control packet includes the following components:

- **Fixed Header**: A required 2-byte field that includes the 4-bit Control Packet type (based on the Quality of Service level), 4-bit flags specific to the packet type, and an 8-bit Remaining Length field that indicates the size of the subsequent headers and payload. The maximum packet size is 256 MB.

- **Variable Header**: An optional field that carries extra information depending on the packet type. A common element in many packet types is the *Packet Identifier*, which helps track message acknowledgments.
- **Payload**: Another optional field containing application-specific content. This includes the *Client Identifier (Client ID)*—a unique string between 1 and 23 characters long used to identify each MQTT client.

The smallest possible MQTT packet, such as an acknowledgment (ACK) message, is only 2 bytes. This minimalism underscores MQTT's efficiency, making it ideal for low-power devices with limited memory and processing capabilities.
Security has become a significant concern as MQTT continues to gain popularity in IoT infrastructure. While the protocol was initially designed for simplicity and efficiency, it lacked built-in security features. Its increasing use in transmitting sensitive or critical data across open networks highlights the need for enhanced security mechanisms [29]. This growing concern has led to a surge in research efforts focused on developing Intrusion Detection Systems (IDS) designed for MQTT-based IoT environments.

2.2 Previous Works

This chapter comprehensively reviews Intrusion Detection System (IDS) techniques used to secure the MQTT protocol, categorizing notable recent studies by methodology. Fig. 3 illustrates a taxonomy of MQTT IDS methods, showing the range of approaches researchers adopt. Identified limitations are discussed in the synthesis section. Our proposed system addresses these gaps by introducing a protocol-aware flow analyzer and an interpretable detection model designed for MQTT.

2.2.1 Signature-Based Detection Models

The signature-based detection approach for MQTT protocol vulnerabilities identifies known malicious patterns within MQTT messages or traffic, including specific payload formats or header information indicative of known attacks. [30] developed an MQTT parsing engine integrated with the Suricata IDS from [31] to accurately identify and verify MQTT packets against known signatures. This allows forwarding incoming packets to a signature-matching module that checks them against predefined rules to detect anomalies or threats, effectively blocking malformed packets. So, with this method, known attacks are efficiently detected. Researchers have proposed various approaches for signature-based detection; however, only the rule-based approach has been examined in MQTT-enabled environments as our research has progressed. This method employs distinct sets of rules to identify and characterize attacks. These rule sets are compared with the monitored data to detect intrusions. A notable study proposed a Bayesian Rule Learning-based Intrusion Detection System (IDS) as a countermeasure against various MQTT-specific cyberattacks [1]. Unlike many simulation-based works, the authors implemented a realistic smart home environment with physically deployed IoT devices to generate a comprehensive dataset including benign and malicious MQTT traffic. The system employs a Bayesian network, learned from training data and domain knowledge, which is subsequently translated into an interpretable set of IF-THEN rules for intrusion classification. As illustrated in

Figure 2, the proposed pipeline consists of network traffic monitoring, feature engineering, discretization of continuous variables, Bayesian network learning, and final rule inference. The inherently interpretable rule set, derived from the Markov blanket of the target node, enables transparent decision-making and classifies attacks such as DoS, PDoS, FDIA, botnet, replay, and sniffing. This approach achieves high accuracy up to 98% while maintaining interpretability, offering a practical rule-based alternative to more opaque machine learning models.

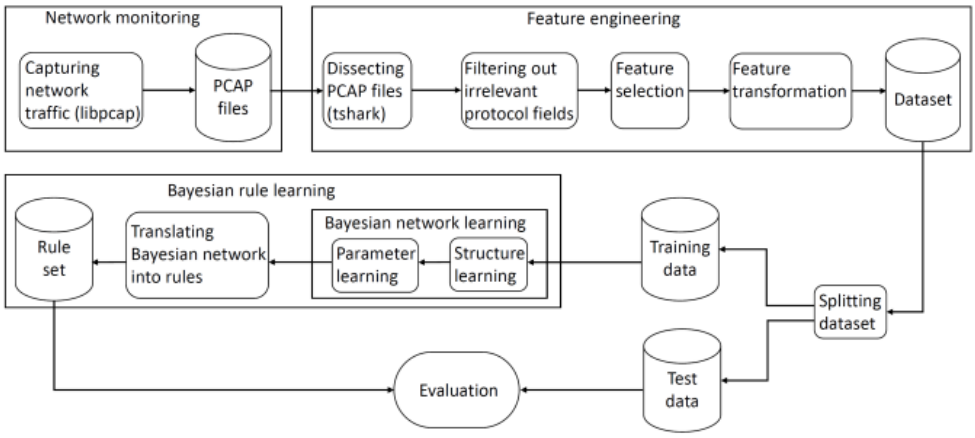

Figure 2: Proposed Bayesian Rule Learning-based IDS by [1]

A rule-based system using a Host Intrusion Detection System (HIDS) was also designed, employing a threshold-based packet discarding policy for various topics defined through MQTT [32]. This technique involves analyzing MQTT messages for specific sequences of characters, patterns, or wildcards that match known malicious signatures, thus enabling the detection of particular threats detailed in the IDS's signature database. However, the signature-based detection method has significant limitations. It fails to identify unknown or new attacks, and creating signatures and rules for every possible attack is not feasible. Furthermore, maintaining an up-to-date signature database requires considerable time and resources. Additionally, minor variations of known attacks can compromise the accuracy of the analysis if the detection system is not configured correctly.

2.2.2 Classical Anomaly Detection Models

Classical anomaly-based detection models are designed to identify unusual or unexpected behavior in network traffic without needing labeled training data or computationally intensive learning algorithms. These techniques rely on heuristics, statistical thresholds, or symbolic representations of behavior, making them lightweight and well-suited for environments with limited resources. Within MQTT-based IoT systems, two widely explored classical approaches include fuzzy logic systems and behavioral analysis models.

- **Fuzzy Logic model:** [2] addresses the inherent vulnerabilities of the MQTT protocol to DoS attacks caused by the expansive and complex nature of IoT networks. The proposed solution, Secure-MQTT, utilizes fuzzy logic techniques, including fuzzy inference and rule interpolation, to effectively analyze network traffic and identify anomalies. Specifically, the system employs a fuzzy rule interpolation

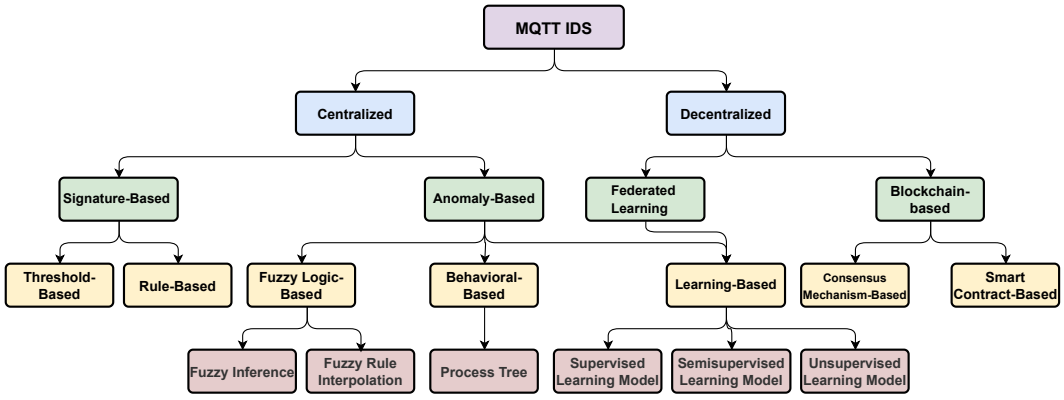

Figure 3: Taxonomy of MQTT IDS methods

mechanism to detect malicious behavior in nodes, thereby eliminating the need for a dense rule base by dynamically generating rules. Although fuzzy logic has demonstrated its efficacy in various systems, including sensor device intrusion detection in the Internet of Things (IoT), its significant difficulty with increasing input dimensions limits its ability to detect attacks on IoT platforms where large amounts of data are continuously transferred. Also, the study lacks details on how the input parameters CMR (Connection Message Ratio) and CAMR (Connection and Activity Message Ratio) were chosen to detect DDoS attacks. There is no information on how the crisp value for anomaly output was obtained. Despite these limitations, Secure-MQTT demonstrates superior accuracy and a lower rate of false positives than traditional security methods, making it particularly suitable for resource-constrained environments prevalent in the IoT. However, the research is primarily validated through simulated scenarios, indicating the need for further real-world applications and expansion to cover additional MQTT messages and attack vectors. As illustrated in Figure 4, the proposed architecture includes a fuzzy module, a rule base, fuzzy rule interpolation, and a network analyzer to support lightweight and adaptive intrusion detection.

- **Behavioral Analysis:**

  Behavioral analysis is a crucial component of anomaly-based detection within the MQTT protocol, designed to identify irregular communication patterns that may indicate potential security threats. The study by [3] proposes a detection technique that models normal and observed network behavior using process trees. These trees capture the structural and sequential relationships of MQTT messages, such as CONNECT, SUBSCRIBE, and PUBLISH, and help compare them against a reference model derived from benign traffic.

  The detection relies on evaluating the partial similarity between subtrees of the observed process tree and those of known abnormal behavior. By calculating Levenshtein-based edit distances, the system identifies sub-sequences in logs that match suspicious patterns, even if these patterns are not in a consecutive order, something that traditional sequence-based methods struggle to detect.

  This method effectively identifies subtle and distributed anomalies, such as repeated disconnection cycles or publish/subscribe inconsistencies, as shown in their MQTT case study. However, the model's

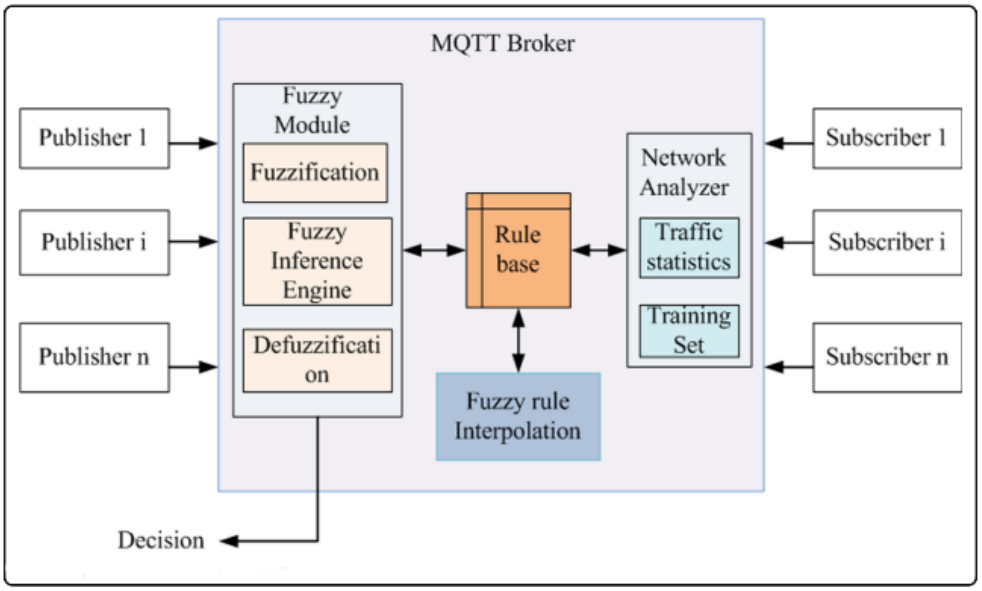

Figure 4: Proposed Secure-MQTT fuzzy logic-based IDS by [2]

efficiency is limited when anomalies lack distinct message flag characteristics. Moreover, it may not generalize well to newer adversarial attack patterns.

The architecture of the proposed detection framework is illustrated in Fig. 5, which outlines the process of constructing normal and observed process trees, comparing them through process mining, and extracting deviating subtrees based on a defined similarity threshold.

In summary, while classical anomaly detection methods—such as fuzzy logic and behavioral analysis—provide lightweight and interpretable solutions for MQTT-based intrusion detection, they face critical limitations. These include limited adaptability, reduced scalability, and weak performance in real-time or complex attack scenarios. As a result, recent research has shifted toward Artificial Intelligence (AI)-driven approaches, including machine learning (ML) and deep learning (DL), to enhance detection accuracy and adaptability.
For example, [33] proposed an AI-based IDS that employs various ML techniques to detect and classify attacks in MQTT communication. Their system, trained on the MQTTset dataset [34], significantly improves detection accuracy. However, it also introduces computational overhead that may limit real-time performance in resource-constrained environments.
To explore these advanced methods, the following subsections—**2.2.3 Supervised Anomaly Detection Models**, **2.2.4 Unsupervised Anomaly Detection Models**, and **2.2.5 Semi-Supervised Anomaly Detection Models**, examine learning-based intrusion detection strategies. Additionally, **2.2.6 Federated Learning-Based IDS Models** highlights approaches that enable collaborative training across distributed IoT nodes while preserving data privacy, and **2.2.7 Blockchain-Based Security Solutions** explores decentralized architectures that enhance trust, integrity, and resilience in MQTT communication. Each category offers distinct strengths and limitations, making them suitable for different aspects of MQTT intrusion detection.

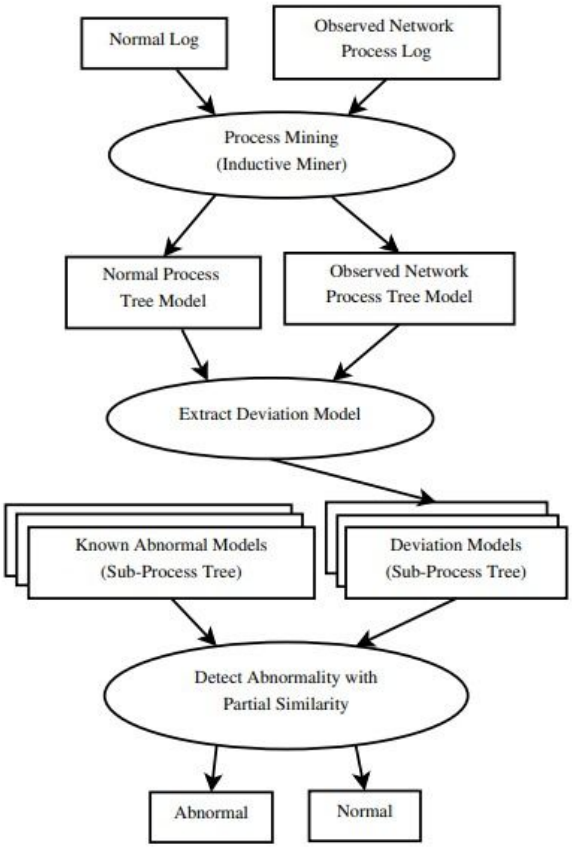

Figure 5: Process tree-based anomaly detection approach adapted from [3].

2.2.3 Supervised Anomaly Detection Models:

In [35], different machine learning (ML) models were used to improve the security of MQTT in IoT environments. These models included eXtreme Gradient Boosting (XGBoost), GRUs, and LSTMs, which were tested on threats such as intrusions, Denial of Service (DoS), and Man-in-the-Middle (MitM) attacks. Other ML methods like Naive Bayes (NB), Random Forest (RF), Decision Trees (DT), Logistic Regression (LR), K-Nearest Neighbors (KNN), and Support Vector Machines (SVM) were also tested using the MQTT-IoT-IDS2020 dataset [36]. The dataset includes several types of attacks, and the results showed that these ML methods could detect MQTT-related intrusions with good performance.

In another study, [37] proposed ARTEMIS, an anomaly-based intrusion detection system for IoT networks using MQTT. Although the paper did not list the specific attacks it tested, the system used different ML techniques like Autoencoders, SO GAAL, Random Forest, Isolation Forest, One-Class SVM, and K-means clustering. These models helped find unusual network behavior and send alerts when something seemed wrong.

In 2020, [36] used the MQTT-IoT-IDS2020 dataset to check how well different ML algorithms can detect MQTT-based attacks. Their study showed that ML can improve the performance of intrusion detection systems in MQTT-based IoT systems. Around the same time, [34] introduced the MQTTset dataset, which was created for supervised learning models in IoT security. They tested it with several models like Neural Networks (NN), Random Forest (RF), Naive Bayes (NB), Decision Trees (DT), Gradient Boosting (GB), and Multilayer Perceptrons (MLP). They pointed out that having a balanced dataset is crucial for fair and accurate results. Their findings confirmed that MQTTset helps train IDS models and provides high accuracy and F1 scores.

[4] proposed a deep learning-based Intrusion Detection System (IDS) tailored for MQTT-enabled IoT environments. Their model leverages a Deep Neural Network (DNN) with two hidden layers and uses either sigmoid or softmax activation at the output layer, depending on the classification task (binary or multi-class). The model was trained and tested on the MQTT-IoT-IDS2020 dataset, which includes three abstraction-level flow features: Uni-flow, Bi-flow, and Packet-flow. The study reported high classification performance across these feature sets, achieving 99.92

In addition to comparing their DNN against classical machine learning models (e.g., NB, RF, KNN, DT), the authors also benchmarked it against LSTM and GRU architectures. The DNN outperformed these alternatives in both accuracy and F1-score while maintaining lower training time on average. However, despite its effectiveness, the model did not include explicit feature selection or dimensionality reduction steps, which could impact scalability and efficiency in larger-scale deployments.

Figure 6 illustrates the proposed DNN pipeline, including dataset preprocessing, model training, and final classification of MQTT traffic as usual or attack.

To address the challenges of intrusion detection in MQTT-enabled IoT networks, Ullah et al. [5] proposed TNN-IDS, a Transformer Neural Network-based intrusion detection system tailored for sequence modeling of MQTT traffic. The authors identified key limitations in prior ML- and DL-based IDS approaches, such as slow learning, poor handling of imbalanced datasets, and limited capability to model temporal dependencies in packet sequences. To overcome these, they leveraged the self-attention mechanism of Transformers, which

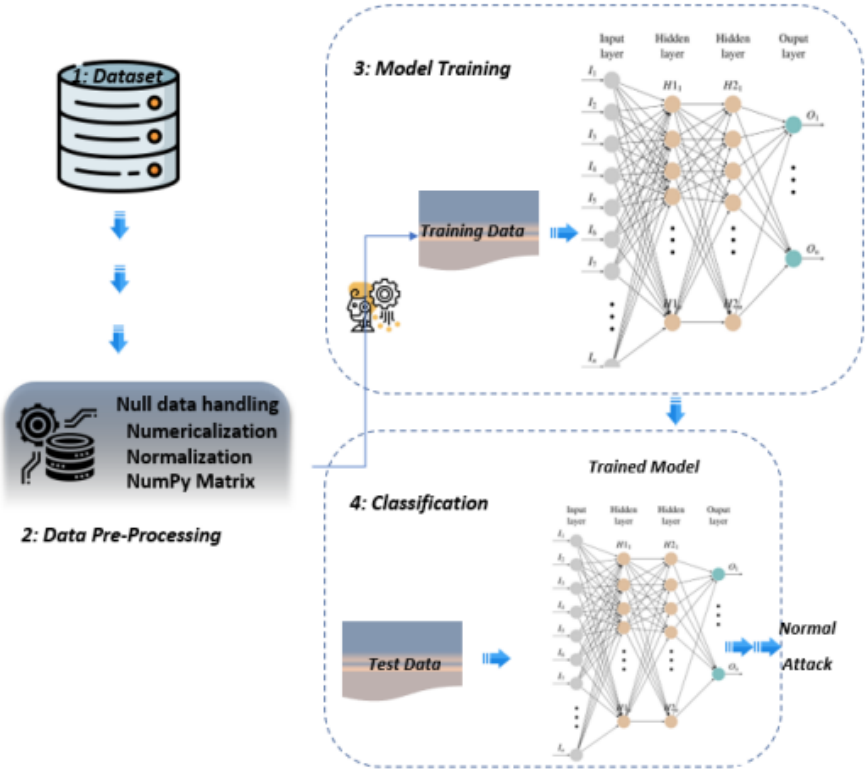

Figure 6: DNN-based IDS framework for MQTT intrusion detection, adapted from [4].

enables modeling long-range dependencies and parallelizing computation across multiple heads.

The model architecture includes a multi-head self-attention encoder, followed by one-dimensional convolutional, normalization, and dense layers, as illustrated in Fig. 7. This architecture was designed to process flow-based features—namely Uni-Flow, Bi-Flow, and Packet-Flow—from the MQTT-IoT-IDS2020 dataset. Feature selection was performed using an Extra Trees Classifier (ETC) to reduce dimensionality and improve generalization. The dataset was preprocessed rigorously, including normalization and handling of missing values, ensuring balanced input for effective learning.

Experiments were conducted using multiple optimizers (SGD, RMSprop, Adam) and batch sizes, revealing that the Adam optimizer with a batch size 32 consistently delivered optimal performance. The TNN-IDS achieved 99.99% accuracy, precision, recall, and F1-score, significantly outperforming classical models such as CNNs, LSTMs, GRUs, and ensemble classifiers, especially on imbalanced Packet-Flow features.

A key advantage of TNN-IDS lies in its robustness to imbalanced classes, scalability across flow types, and minimal need for domain-specific feature engineering. Its effectiveness across Uni-Flow, Bi-Flow, and Packet-Flow features suggests strong generalizability. The authors conclude their transformer-based approach is a promising alternative to conventional DL models for sequence-based intrusion detection in resource-constrained IoT environments.

Traditional intrusion detection systems often suffer from high dimensionality due to irrelevant or redundant features, leading to increased computational cost and reduced classification accuracy. To overcome these challenges, [6] proposed a lightweight and practical IDS framework for MQTT-based IoT networks by com-

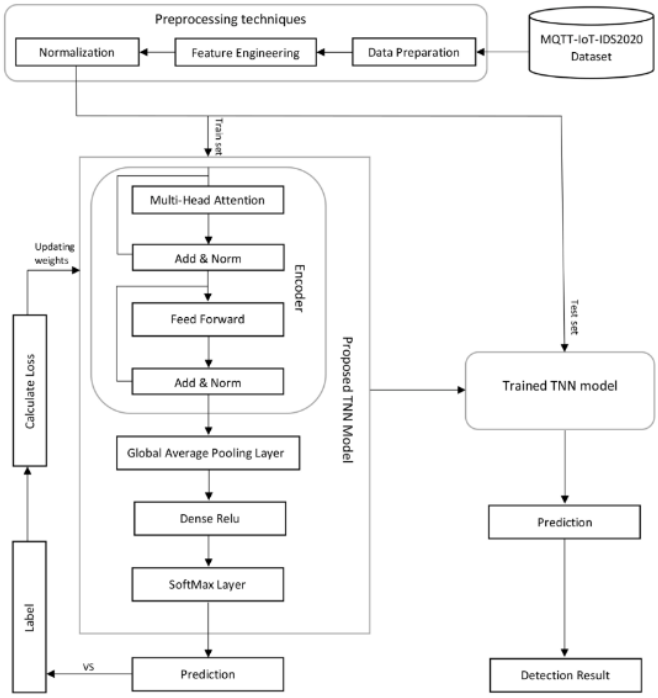

Figure 7: Proposed TNN-IDS model showing the multi-head attention, convolutional, and dense layers, adapted from [5].

bining the Enhanced Chaotic Salp Swarm Algorithm (ECSSA) for feature selection with the LightGBM classifier for detection. ECSSA is a nature-inspired metaheuristic that dynamically selects the most relevant subset of features, improving detection speed and accuracy. The authors also incorporated the SMOTE oversampling technique before training to address class imbalance in intrusion datasets.

The final model, named ECSSA-LightGBM NIDS, was tested on multiple MQTT datasets (MQTT-IoT-IDS2020 and MQTTset) and consistently outperformed baseline ML models such as Decision Trees, Random Forest, and XGBoost in terms of accuracy, precision, and detection time. As illustrated in Figure 8, the proposed architecture includes modules for data preprocessing, ECSSA-based feature selection, SMOTE-based balancing, and LightGBM classification, culminating in a binary decision output indicating regular or attack traffic.

2.2.4 Unsupervised Anomaly Detection Models:

Most of the existing work on IDS for MQTT networks has used supervised models ([4, 36, 35, 34, 38, 6]). These models effectively classify known MQTT attacks but struggle to detect and correctly classify unseen data as normal or anomalous.

To address this limitation, [7] proposed an unsupervised intrusion detection system tailored for resource-constrained IoT environments. Their system, EdgeIDS, leverages the Skip-GANomaly architecture—a generative adversarial network (GAN) framework with skip connections—to detect anomalies in MQTT traffic. The model is designed for deployment at the edge node level, enabling local anomaly detection on IoT devices such as cameras, sensors, or smart home appliances. EdgeIDS includes an encoder-decoder structure

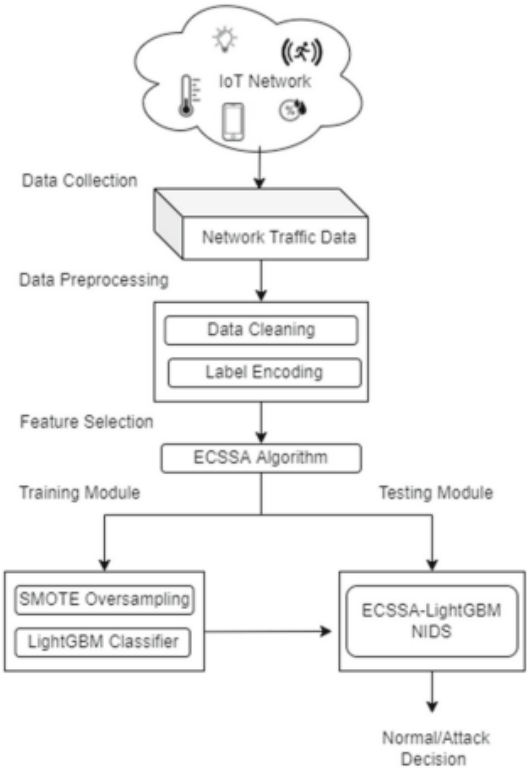

Figure 8: Architecture of the proposed ECSSA-LightGBM-based IDS [6]

in the generator, where the encoder compresses input data into a latent vector and the decoder reconstructs it. A discriminator then distinguishes between real and reconstructed traffic while extracting latent features. Anomalies are detected by measuring reconstruction errors and discrepancies in latent space representations. To build the model, input data is normalized and transformed into image-like representations using pixel mapping techniques. The generator learns to reconstruct standard traffic patterns, and deviations during testing indicate potential threats. Though the architecture performs well across multiple MQTT-specific attacks, such as brute force, malformed data, and DoS, the reliance on image conversion and the computational complexity of GANs make EdgeIDS less suitable for real-time inference on constrained devices.

As shown in Figure 9, the architecture comprises three main tiers: the Edge node for local detection using EdgeIDS, the Fog node for intermediate processing (FogIDS), and the Cloud node for broader contextual analysis and dataset aggregation. This tiered approach allows security measures to be distributed across different layers of the IoT network, enhancing scalability and robustness.

Additionally, [8] introduced a GAN-based autoencoder model, called GAN-AE, to detect unknown intrusions in MQTT-based IoT networks. This unsupervised approach combines a standard autoencoder with adversarial training from a Generative Adversarial Network (GAN), enhancing the model's ability to generalize to previously unseen attacks. Unlike traditional supervised models, GAN-AE learns only from regular traffic, making it suitable for detecting zero-day anomalies.

The architecture consists of two training phases. In Phase 1, the model learns to reconstruct input traffic data using an encoder-decoder structure, minimizing the reconstruction loss between the input and its recon-

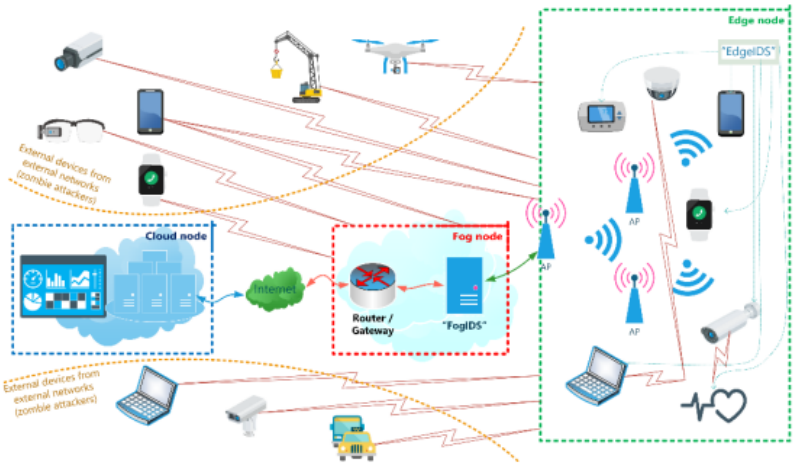

Figure 9: Proposed architecture of EdgeIDS for anomaly detection in IoT environments [7]

struction. In Phase 2, adversarial training is introduced—treating the autoencoder as a generator, it competes against a discriminator that attempts to distinguish real input flows from reconstructed ones. This competition encourages the generator to approximate normal data distribution better, making anomalies more distinguishable during inference.

During evaluation, an input is flagged as anomalous if its reconstruction error exceeds a certain threshold, determined based on the distribution of errors in the validation set. The GAN-AE model outperformed baseline models such as the standard autoencoder, One-Class SVM, and Isolation Forest, achieving a detection accuracy of 0.97 on both public (MQTT-IoT-IDS2020) and custom flow-based MQTT datasets.

The two-stage training pipeline of GAN-AE is illustrated in Figure 10, showing the integration of reconstruction-based and adversarial loss updates.

2.2.5 Semi-Supervised Anomaly Detection Models:

[9] addresses the challenge of detecting MQTT-based attacks in real time through an online-semisupervised anomaly detection system. The proposed model leverages a regularized Single-hidden Feedforward Neural Network (S-hFFNN) trained using a small amount of labeled data and a larger pool of unlabeled data. This approach significantly reduces the burden of manual labeling, which is often expensive and time-consuming in real-world IoT deployments.

The system operates in an iterative manner. The initial model is trained on available labeled data and then continuously updated by selecting the most informative samples from the unlabeled pool using a heuristic active learning strategy. These selected samples are labeled and added to the training set, refining the model over time. The method incorporates regularization to avoid overfitting and ensures robust performance even in noise and class imbalance.

The core workflow of this adaptive learning strategy is outlined in Algorithm 11, showing how labeled and unlabeled data are combined to enhance the anomaly detector iteratively.

This approach proved effective against multiple MQTT-based attacks, including Flooding DoS, Brute Force, and Malformed Packet attacks, using the MQTTset dataset. While the method provides excellent adapt-

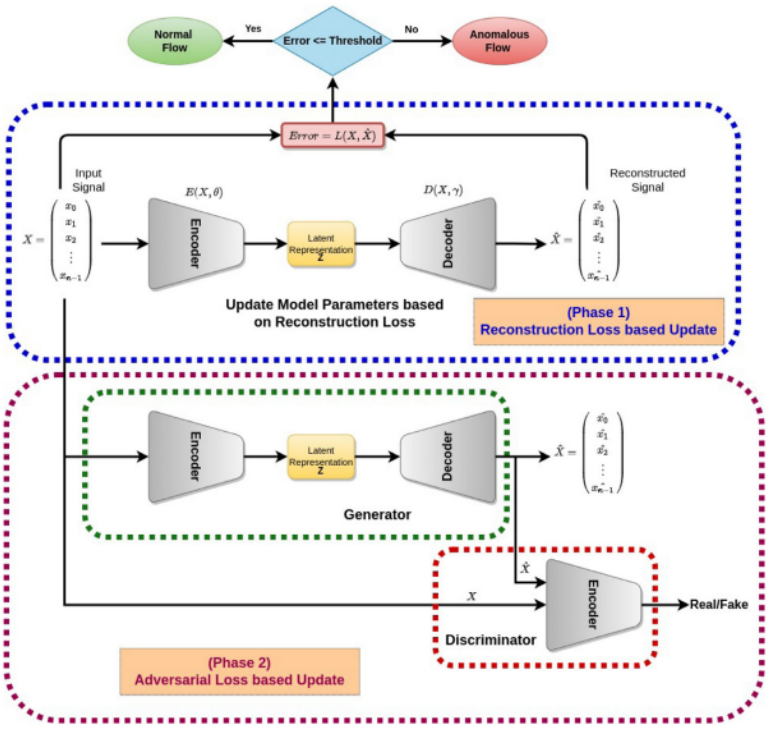

Figure 10: Architecture of the proposed GAN-AE model for unsupervised anomaly detection [8]

ability and detection performance, potential drawbacks include the need for some initial labeled data and computational complexity during iterative updates.

[10] proposed a novel semi-supervised deep learning model named SS-Deep-ID for intrusion detection in IoT networks. This architecture's core is a Multiscale Residual Temporal Convolutional (MS-Res) module, which enhances the model's ability to learn spatiotemporal representations from sequential traffic data. The MS-Res block combines multiple dilated causal convolutional (DC-Conv) layers at different receptive fields to capture short- and long-range dependencies in the traffic sequence. These are connected via residual links to improve training stability and prevent vanishing gradients. To further refine learning, the output of each DC-Conv passes through ReLU activation, batch normalization, and dropout layers, as shown in Figure 12. In parallel, the architecture integrates a traffic attention (TA) mechanism that estimates the importance of each traffic feature, allowing the network to focus on the most discriminative information while suppressing irrelevant noise. This dual-path structure (MS-Res + TA) enables more accurate and robust detection of complex intrusion patterns, even when trained with limited labeled data. Experimental results on CIC-IDS2017 and CIC-IDS2018 demonstrate the model's strong generalization ability, with SS-Deep-ID outperforming both supervised and unsupervised baselines across multiple performance metrics, while maintaining computational efficiency.

[11] addresses the class imbalance issue in IoT anomaly detection by designing a flexible and semi-supervised **framework based on conditional Generative Adversarial Networks (cGANs)**. This framework introduces two key models: the **one-class cGAN (ocGAN)** and the **binary-class cGAN (bcGAN)**. The ocGAN is trained solely on minority class data (typically anomalies), generating realistic synthetic samples to bal-

(1) Input L–labeled data, U–unlabeled data, C–regularized S-hFFNN, q–active learning strategy, m–new labeled data, *maxIter*–max iterations
(2) $h \longleftarrow C(L_0)$
(3) for i from 1 to maxIter
(4) choose m from $x \in U$ based on q strategy
(5) $\omega \longleftarrow f(x)$
(6) $L \longleftarrow L \bigcup (x, \omega)$
(7) $U \longleftarrow U - (x, \omega)$
(8) $h \longleftarrow h(L_0)$
(9) end for

Figure 11: Online-semisupervised anomaly detection algorithm proposed by [9]. The model iteratively selects and labels the most informative samples from the unlabeled pool to improve detection performance.

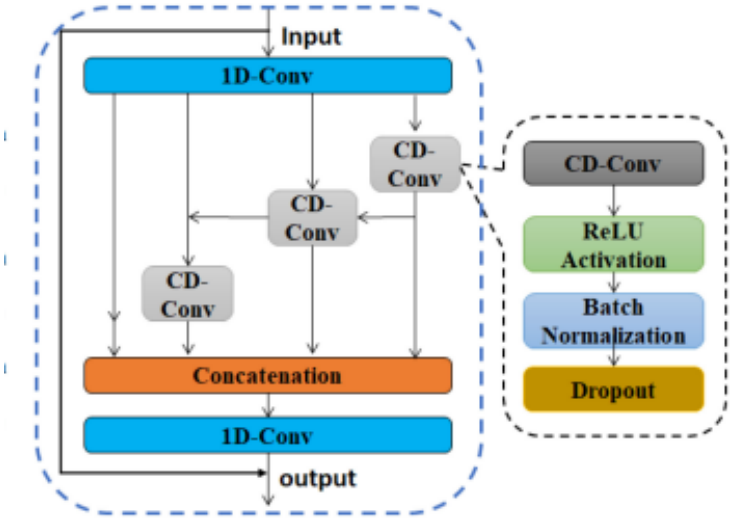

Figure 12: MS-Res module structure within the SS-Deep-ID model [10]. The module captures multiscale temporal dependencies via residual connections and dilated convolutions.

ance the dataset. The bcGAN, the other hand, is trained on balanced binary datasets (Benign vs. anomaly) to produce synthetic data for both classes, enabling enhanced classification performance.
The framework is evaluated under five different configurations, as illustrated in Figure 13:

- **(a)** and **(b)**: ocGAN generates minority class samples and the detector is trained with either real majority + synthetic minority or real + synthetic minority combined with real majority.
- **(c)**: bcGAN is directly trained on a balanced dataset for binary classification.
- **(d)**: ocGAN is used to balance the dataset, followed by bcGAN for data augmentation.
- **(e)**: Multiple bcGANs generate synthetic samples for each attack category, which are merged into a multiclass training dataset.

These architectures significantly improve anomaly detection performance on highly imbalanced IoT datasets like MQTT-IoT-IDS2020 and MQTTset, achieving detection rates of 98.66% and 99.10%, respectively. Despite the impressive results, the framework demands high computational resources and careful training to avoid overfitting, especially in cases with limited minority-class samples.

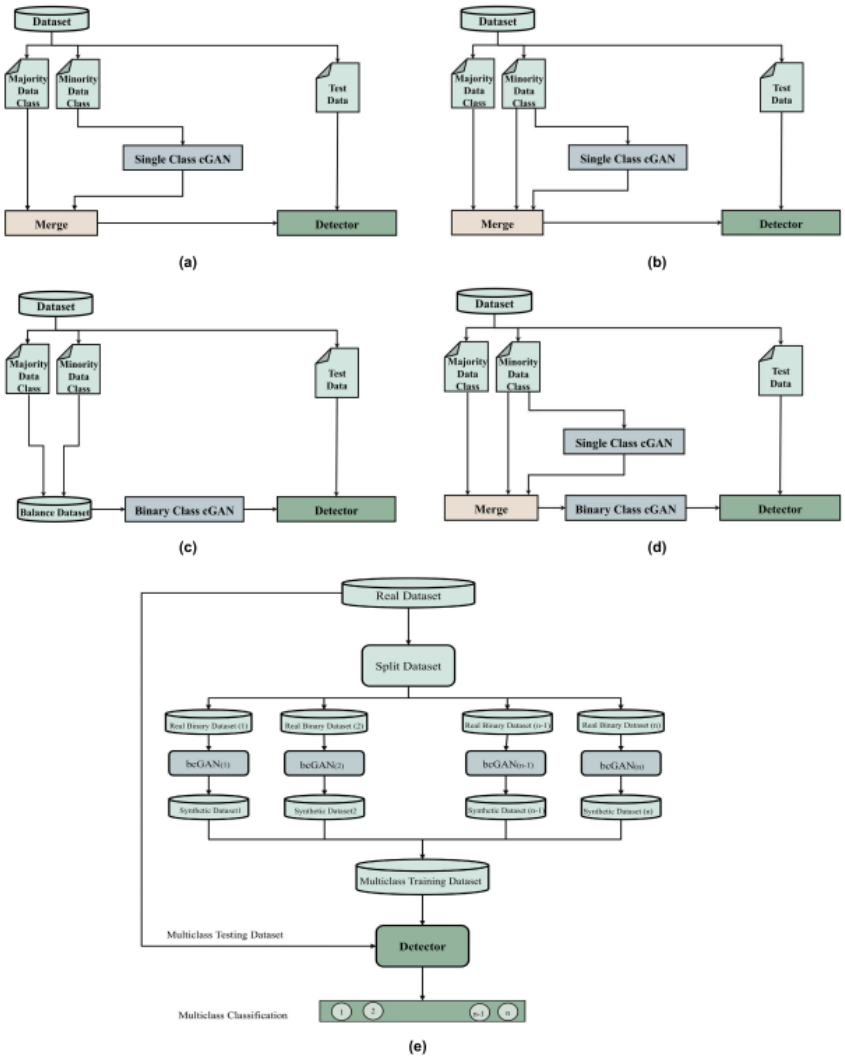

Figure 13: Anomaly detection framework configurations using ocGAN and bcGAN models [11]. (a)–(d): Binary-class settings; (e): Multiclass generation strategy.

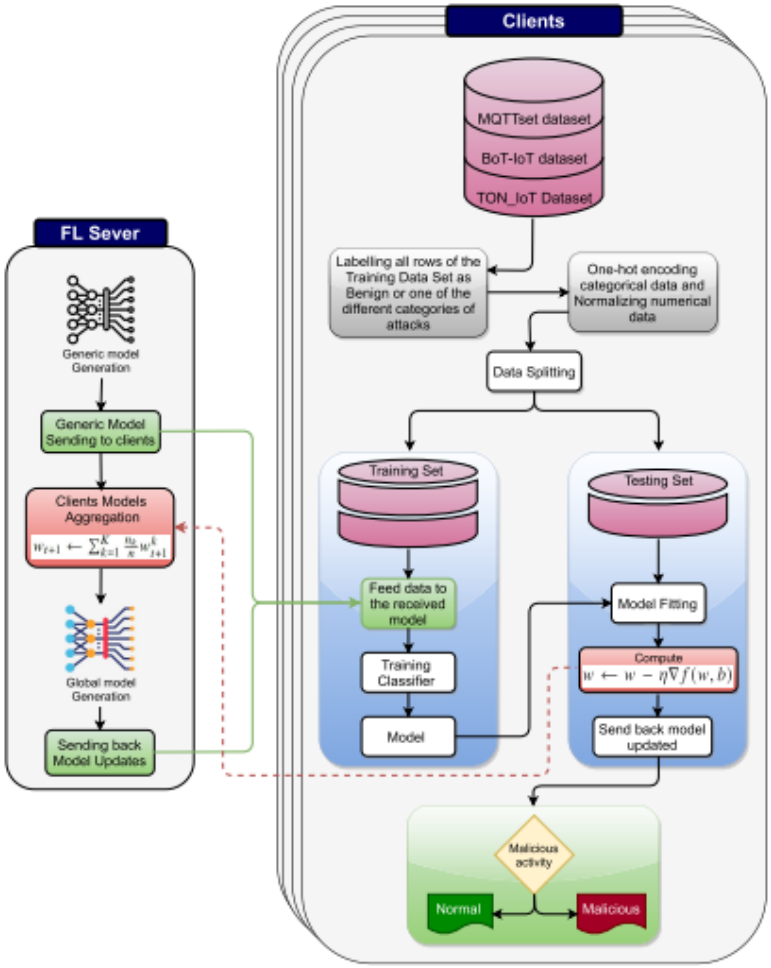

Figure 14: The architecture of the federated deep learning-based IoT intrusion detection system [12].

2.2.6 Federated Learning-Based IDS Models:

[12] presents a comprehensive federated deep learning-based intrusion detection system (IDS) designed specifically for IoT networks. The system addresses key challenges in centralized machine learning approaches, such as privacy leakage, high communication cost, and lack of scalability, by distributing the learning process across clients while keeping the raw data local. The proposed architecture supports multiple deep learning classifiers, including CNN, RNN, and DNN, and was tested on three major IoT security datasets: MQTTset, BoT-IoT, and TON_IoT.

The central server (FL server) initializes a generic model and sends it to the participating clients in this federated learning setup. Each client trains the model locally using its private dataset and then returns the updated model weights to the server. The server aggregates these updates to form a new global model, which is redistributed in the following training round. This process repeats until convergence. The training and testing pipeline at the client side includes one-hot encoding of categorical features, normalization of numerical data, and model evaluation using reconstructed local datasets. As shown in Figure 14, the system maintains privacy while achieving high classification accuracy for detecting malicious MQTT traffic.

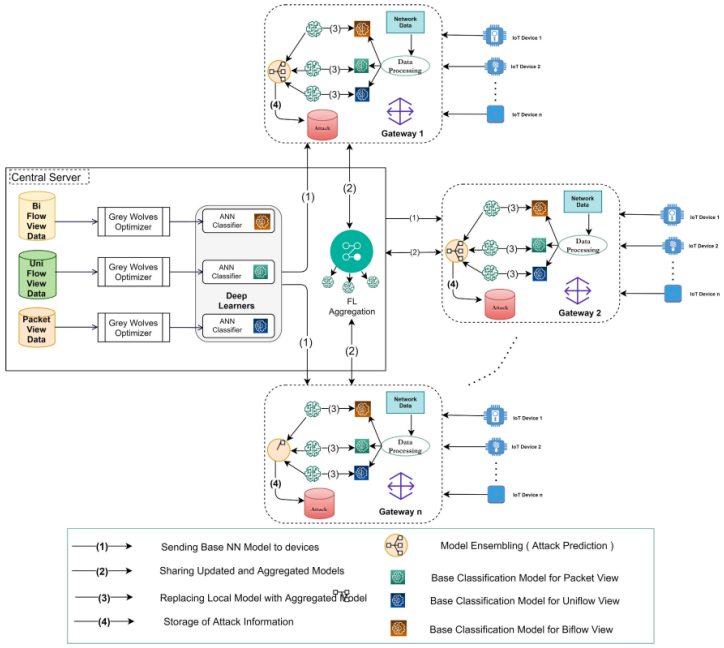

Figure 15: MV-FLID architecture for federated multi-view intrusion detection using Grey Wolves Optimizer and ensemble learning [13].

[13] introduced a novel federated learning-based intrusion detection framework named **Multi-View Federated Learning-Based Intrusion Detection (MV-FLID)**. This system trains machine learning models on decentralized MQTT-based IoT network traffic without exposing raw data, thereby preserving user privacy. Unlike traditional intrusion detection approaches that rely on centralized learning from a single feature set, MV-FLID uses a multi-view ensemble learning strategy that segments network data into three distinct views: -flow, Uni-flow, and Packet view. Each view is processed independently using a separate base classifier, enabling the system to capture complementary patterns of network behavior across views.

Each gateway (representing a federated client) trains view-specific neural network models locally using optimized feature subsets selected via the Grey Wolves Optimizer (GWO). These local models are periodically sent to a central server, where a federated aggregation process combines them into a global model for each view. The updated models are then redistributed back to the gateways. An ensemble model aggregates predictions from all three view-specific models to enhance accuracy to decide whether a traffic flow is normal or malicious.

As depicted in Figure 15, the architecture enables collaboration across gateways without sharing sensitive device data, improving intrusion detection performance while maintaining privacy. However, MV-FLID is not immune to federated poisoning attacks—if a malicious gateway uploads manipulated models, it could degrade the performance of the global model. The authors acknowledge this vulnerability and suggest future enhancements through outlier filtering mechanisms.

In addition, the GEMLIDS-MIoT framework proposed in [14] introduces an energy-efficient, federated learning-based intrusion detection system (IDS) tailored for Medical IoT (MIoT) networks. This architecture eliminates the need for centralized data aggregation by leveraging federated learning, allowing distributed

MIoT gateways to collaboratively improve the model without sharing raw data.

The framework integrates an anomaly detector based on One-Class SVM to identify previously unseen threats and an attack classifier built on an Enhanced Random Forest (ERF) to accurately recognize known attacks such as DDoS, MitM, and brute force with minimal resource usage. A federated learning module coordinates model training across gateways by exchanging only encrypted model updates. These updates are validated and aggregated on a central threat intelligence server using secure protocols like FedSecAgg, which ensures robustness against poisoning attacks and maintains synchronization across devices.

Each gateway performs local real-time detection and contributes to a shared global model, enabling continuous adaptation to new threats while keeping energy consumption and latency low. This design makes the system well-suited for constrained environments like healthcare IoT, while offering methodological insights applicable to MQTT-based and broader IoT intrusion detection.

As shown in Figure 16, the architecture supports a fully distributed, secure, and power-efficient detection process across MIoT nodes.

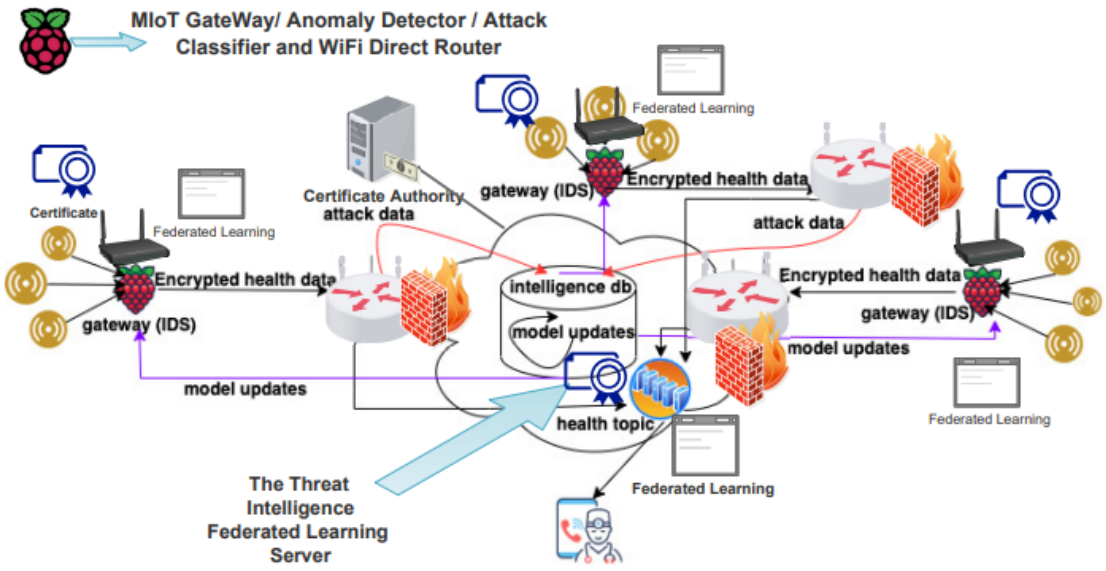

Figure 16: Architecture of the GEMLIDS-MIoT federated learning-based intrusion detection system [14]. It enables secure and energy-efficient anomaly detection across distributed medical IoT gateways.

2.2.7 Blockchain-Based Security Solutions

The decentralized nature of blockchain eliminates the need for a central authority, reducing the risk of single points of failure and enhancing system resilience against various attacks by distributing control across multiple nodes. This distribution ensures no single entity has overarching control, creating a more robust and secure MQTT communication environment.

To further enhance data security in MQTT-based IoT systems, [15] proposed a hybrid model that integrates MQTT communication with blockchain technology, aiming to create a tamper-proof and traceable ledger of IoT telemetry data. The central idea is to leverage blockchain's decentralized and immutable nature to

overcome MQTT's traditional vulnerabilities, such as a lack of strong authentication and potential single point of failure due to reliance on a central broker.

In the proposed architecture, each IoT sensor transmits telemetry data to a gateway device, which publishes this data through an MQTT broker. Instead of the broker simply forwarding data to subscribers, it is integrated with a blockchain validator node. This validator parses the MQTT messages into blockchain-compatible transactions and verifies them using a Proof-of-Authority (PoA) consensus mechanism. Once verified, the transaction is broadcast to the blockchain network, which is appended to the distributed ledger, ensuring immutability and traceability of the data.

The use of validator nodes maintains decentralization and enhances scalability and performance due to the lightweight nature of PoA. The system is particularly effective for private IoT networks, where trust is based on identity rather than financial incentives. It suits industrial applications such as supply chain management or smart healthcare systems. The sequence of operations—data collection, publishing, validation, and ledger updates—is shown in Figure 17.

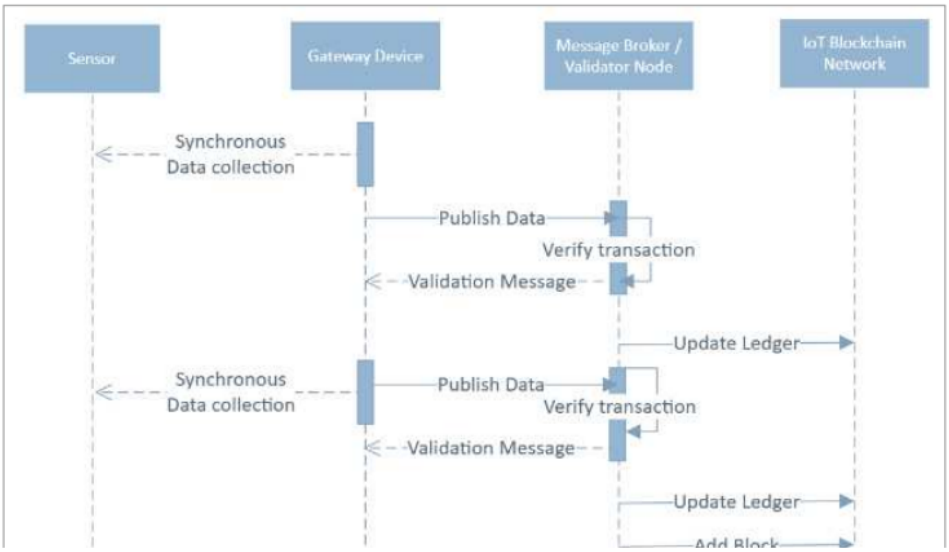

Figure 17: Sequence diagram of the proposed MQTT-Blockchain integration model [15]. It shows how sensor data is collected, validated, and recorded onto a blockchain ledger via MQTT message brokers acting as validator nodes.

In addition, [16] introduced a novel blockchain-based architecture that employs blockchain sharding and Ethereum smart contracts to enhance the security and scalability of MQTT communications. In this system, publishers encrypt and hash MQTT messages before transmission. These messages are then sent to an MQTT broker, which forwards them to a broker node responsible for managing data access and interaction with the blockchain network.

The broker node plays a pivotal role by assigning the encrypted messages to one of several shards based on a randomized sharding logic. Each shard operates independently and stores a subset of the blockchain, enabling parallel transaction processing, significantly reducing computation, and improving scalability. The broker node maintains an access control list and notifies authorized subscribers about new data availability. When subscribers receive a notification, they retrieve the data from the corresponding shard and perform

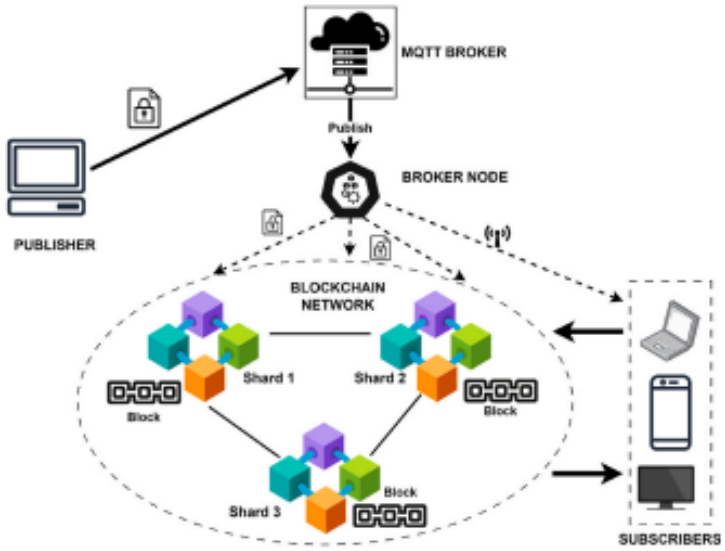

Figure 18: Proposed blockchain-sharding-based MQTT security architecture, adapted from [16].

dual verification using both the message hash and a nonce to ensure data integrity and authenticity. A smart contract governs access control and enforces signature verification rules, ensuring only authorized users can decrypt and process messages.

The complete architectural workflow, including the interaction between MQTT clients, the broker node, and the blockchain shards, is illustrated in Fig. 18. This model demonstrates how integrating sharding and cryptographic techniques can improve the' efficiency, privacy, and reliability of MQTT-based IoT systems.

While the proposed architecture provides improved data integrity, low latency, and efficient resource utilization, it does come with specific challenges. These include increased system complexity, potential latency from blockchain operations, and higher initial deployment costs. Nevertheless, this decentralized and user-controlled approach presents a scalable and robust solution for securing MQTT communication in resource-constrained IoT environments.

2.3 Synthesis

Several shortcomings have been identified in the existing methods for intrusion detection in MQTT protocols within IoT environments, including:

1. **Limited Scope on Specific Malicious Behavior**: Many existing methods, like rule-based IDS, focus on known signatures or specific attack patterns, limiting their ability to detect novel threats [30, 1, 32].

2. **Lack of Comprehensive Feature Sets**: Accurate detection requires comprehensive feature sets. However, most studies used datasets with limited features, which can compromise detection accuracy [2, 3].

3. **Lack of Baseline Comparisons with Signature-Based IDS**: Many machine-learning-based IDS studies do not compare their results with traditional signature-based IDS, making it difficult to determine the actual improvement over traditional methods. [39]

4. **Inadequacy for Zero-Day Attacks**: The ability to profile and detect zero-day attacks is crucial, but most existing systems are not equipped to handle these emerging threats [8, 32, 33]

5. **Lack of Explainability in Deep Learning-Based Approaches:** Many previous works employing deep learning techniques lack explainability, making it difficult to understand and trust the decision-making process of these models [4, 38, 5, 40].

6. **Lack of Defined Profiles for Attacks**: Many profiling works lack clear profiles for both benign and malicious activities, which reduces the effectiveness of detection systems [35],

7. **Lack of Comprehensive Dataset Requirements**: Effective IDS training and evaluation require comprehensive datasets often lacking in prior research [34, 12]. Also, paper [14] relied mainly on their own synthetic/curated MIoT dataset. The model's evaluation didn't involve larger, more general public datasets. Section 4.1 highlighted available datasets' main issues and shortcomings.

8. **Vulnerability to slight variations of known attacks**: Even minor changes in known attack patterns or rules can undermine the detection system's accuracy if it is not properly configured. [1, 32, 30, 41].

9. **Inapplicability for Online Detection**: Real-time detection is essential for IoT security, yet many profiling systems are unsuitable for online detection scenarios [9, 4].

10. **Limited Scalability**: Scalability is a critical requirement for handling new applications, protocols, and features, but many existing systems lack this capability [6], [2].

11. **High Complexity and Performance Challenges**: The complexity in terms of time, preparation, real-time detection, maintaining an up-to-date signature database, and overall performance is often overlooked, impacting the usability of IDS [16, 38, 7, 30, 1, 32, 15, 13].

12. **Limited focus on real attack traffic**: The research or study emphasizes simulated attack scenarios rather than real-world data. In [2], the attacking scenario is simulated in a network where the number of malicious nodes ranges from 10 to 50% of the total number of nodes deployed.

This study effectively addressed and resolved issues 2, 5, 6, and 11 through a comprehensive review and integration of advanced intrusion detection techniques. By incorporating explainable learning-based detection methods, we have enhanced detection accuracy, scalability, and adaptability, ensuring robust security solutions for MQTT-based IoT networks.

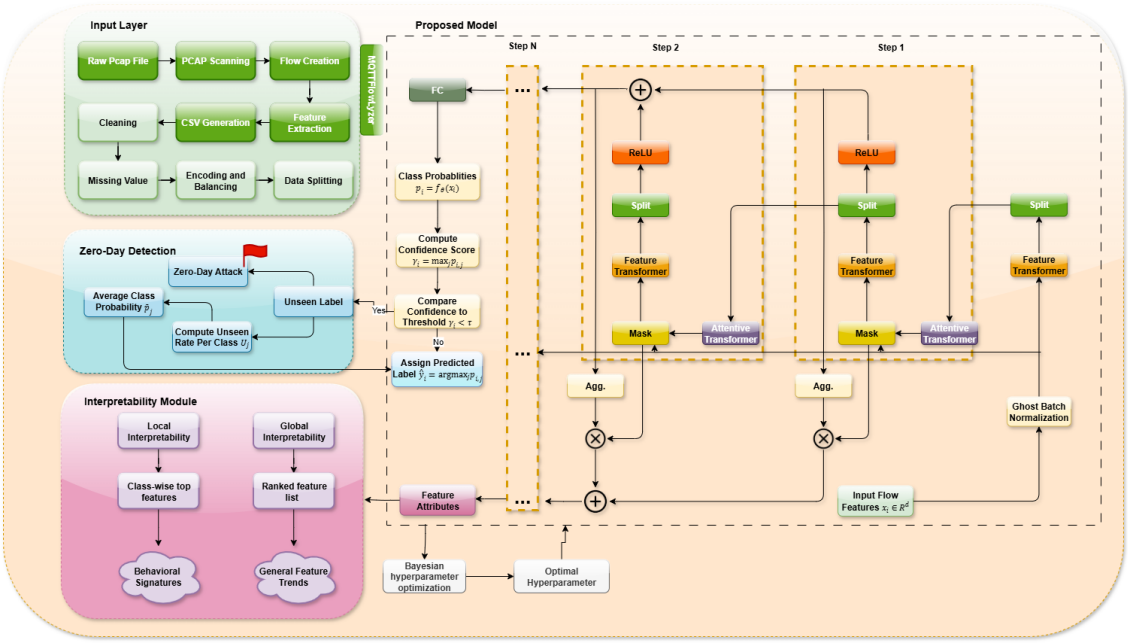

Figure 19: End-to-end architecture of the proposed TabNet-based MQTT intrusion detection framework.

3 Proposed Model

This section introduces the proposed deep learning-based intrusion detection framework for MQTT-enabled IoT environments. The model is developed in response to several key limitations outlined in Section 2.4, including the use of limited feature sets (Limitation 2), lack of interpretability in deep learning models (Limitation 5), absence of clearly defined behavioral profiles for network traffic (Limitation 6), and challenges related to performance and complexity in practical deployment (Limitation 11).

Unlike prior approaches that rely heavily on traditional machine learning techniques such as Random Forest or XGBoost, the proposed method employs TabNet, an architecture built explicitly for structured data. It combines feature selection and classification in a unified, interpretable framework. By leveraging flow-level representations of MQTT traffic and integrating a confidence-based mechanism for identifying zero-day threats, the model aims to deliver improved accuracy, transparency, and adaptability in real-world IoT settings.

The diagram of the proposed model architecture is presented in Figure 19, which illustrates the entire pipeline from raw PCAP input to final zero-day detection and interpretability output.

The following subsections describe each framework component, including data preprocessing, feature transformation, model training, and evaluation procedures.

3.1 MQTTFlowLyzer: MQTT Layer Traffic Feature Extractor

The MQTTFlowLyzer is a purpose-built tool designed to analyze MQTT protocol traffic and extract meaningful features for intrusion detection in IoT networks. It processes raw packet capture (PCAP) files, aggregates packets into MQTT flows, and generates structured datasets compatible with machine learning models.

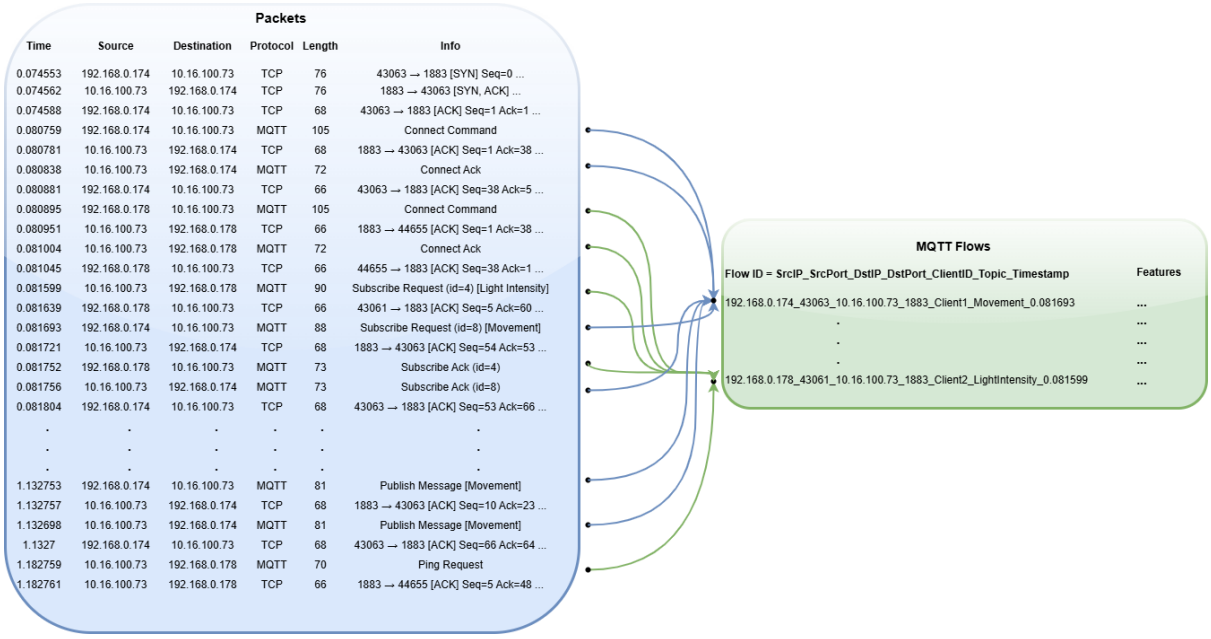

Figure 20: MQTTFlowLyzer constructs flows by grouping packets with the same source and destination IPs, ports, ClientID, and Topic. This hybrid flow definition combines transport-level identifiers with MQTT-specific semantics to enable accurate application-layer session reconstruction and feature extraction.

This is the first work to apply flow-based analysis to the MQTT protocol at the application layer. By focusing on MQTT-specific characteristics, the tool enables detailed analysis of communication patterns, which can improve the performance of intrusion detection systems.

3.1.1 Flow Creation

A core innovation of our methodology lies in redefining the concept of a flow for application-layer protocols, designed explicitly for MQTT. Traditional intrusion detection systems typically define a flow using the 5-tuple of the transport layer: source IP, destination IP, source port, destination port, and protocol. While effective for TCP/IP-level traffic [42], this definition lacks the granularity required to capture the semantic structure of MQTT's publish-subscribe architecture.

We introduce MQTTFlowLyzer, a protocol-aware tool that constructs session-level flows by incorporating MQTT-specific metadata to address this limitation. Each flow is defined using the following attributes:

- Client Identifier (ClientID) – a unique identifier representing the initiating MQTT client
- Topic(s) – one or more hierarchical topic strings used during the session
- Session Timestamps – timestamps marking the beginning and end of the communication session

Figure 20 illustrates the mapping process from individual packets to MQTT-aware flows, demonstrating how multiple MQTT control packets are aggregated into semantically coherent flows based on client identifiers and topic associations.

Flows are initiated when a CONNECT control packet is detected or when a new combination of ClientID and Topic is observed. A session is terminated under any of the following conditions:

- A DISCONNECT control packet is received, explicitly signaling the end of the session.
- The session exceeds a predefined maximum duration, as specified by protocol-aware configurations.
- The session becomes inactive. Specifically, if no packets are observed within a period exceeding twice the keepAlive interval advertised by the client, the session is considered stale and is marked for termination. Although the MQTT 3.1.1 specification requires a timeout threshold of 1.5 times keepAlive, our implementation opts for a slightly more conservative 2 times threshold to tolerate minor transmission delays while still mitigating denial-of-service risks such as those exposed in CVE-2020-13849 [43]. This CVE highlights how slow-rate attacks (e.g., SlowITe) can abuse insufficient timeout enforcement to monopolize server resources.

This redefinition offers two main advantages for downstream analysis:

1. Semantic alignment with the MQTT protocol, which ensures flow boundaries reflect true session semantics and allow more accurate behavioral modeling
2. Improved feature richness, as the use of MQTT-specific indicators such as QoS levels, topic structures, and control packet patterns allows for more effective anomaly detection

The flow construction process also accounts for packets containing multiple topics by extracting and treating each topic as an independent subflow, preserving analytical granularity. Both unidirectional and bidirectional interactions are supported, ensuring that various communication patterns commonly found in IoT environments are accurately captured.
This flow abstraction forms the foundation for behaviorally informed intrusion detection, enhancing the proposed learning framework's interpretability and performance in MQTT-driven IoT networks.

3.1.2 Feature Extraction

MQTTFlowLyzer extracts many features from MQTT traffic flows, enabling detailed analysis and anomaly detection. These features capture key aspects of MQTT communication at both the packet and flow levels. The extraction process begins by analyzing the types of MQTT control packets, including CONNECT, PUBLISH, SUBSCRIBE, and PINGREQ, as well as their corresponding acknowledgments. The total number of these packets within a flow and their proportions provide insight into communication patterns. For example, frequent PINGREQ packets may indicate an effort to maintain a connection, while many PUBLISH packets may signify a data-intensive session.
Flow-level attributes such as the total duration of communication, the volume of data exchanged in both forward and backward directions, and the time intervals between packets are calculated to represent the overall behavior of the flow. These attributes include advanced metrics, such as the mean, variance, and skewness of inter-arrival times, which help identify unusual delays or irregular traffic patterns. MQTTFlowLyzer also

examines features specific to the protocol, such as the Quality of Service (QoS) levels for published and subscribed messages to enhance profiling. Including session-level details like packet and message ratios enables additional behavioral profiling. This is particularly useful in identifying flows with suspicious authentication patterns.

Statistical analysis is performed on payload sizes and message lengths to capture variations and anomalies. Metrics such as the mean, median, standard deviation, and kurtosis of message lengths are derived to provide a complete picture of the flow data characteristics. These extracted features ensure that MQTTFlowLyzer delivers a comprehensive and subtle representation of MQTT traffic. This level of detail enables accurate profiling of regular communication and effective detection of anomalies or malicious behavior.

After generating flows using MQTTFlowLyzer, we extracted a total of 404 features to characterize MQTT communication behavior comprehensively for each flow. These features were categorized into distinct groups based on statistical, temporal, and protocol-specific characteristics, as shown in Figure 21. Each category captures a particular aspect of MQTT network behavior:

- **Delta-Time Based**: Measures variations in packet timing, including publish packet time, forward packet time, and backward packet time.
- **Delta-Length Based**: Captures packet, message, and MQTT length variations across flows.
- **Abs Delta-Length Based**: Focuses on absolute variations in packet, message, and MQTT length across flows.
- **Time Based**: Includes start time, end time, session duration, and directional duration for analyzing flow characteristics.
- **Count Based**: Tracks MQTT control packet occurrences (e.g., connect, publish, subscribe, pingreq) and their respective percentages.
- **Ratio-Based**: Evaluates packet and message ratios, including time rate, packet rate, and length rate, to detect imbalances in MQTT communication.
- **Header-Based**: Extracts fields from MQTT headers, including source/destination IP and port, client ID, topic, QoS, and protocol-specific flags.
- **Size-Based**: Measures packet length, message length, and MQTT message size, including statistical moments such as mean, variance, skewness, and kurtosis.

A detailed breakdown of these feature categories, along with their corresponding attributes and feature identifiers, is provided in Table 2. This tabular representation complements the hierarchical classification in 21 by offering a granular view of each extracted feature. The table lists all features generated by MQTTFlowLyzer, organized by category, and highlights the comprehensive scope of the feature set used to characterize MQTT communication flows.

Table 2: MQTTFlowLyzer Features

Category	Num	Feature Name	Num	Feature Name
Delta-Length based	F1	fwdPacketDeltaLengthMean	F46	bwdMsgDeltaLengthMedian
	F2	fwdPacketDeltaLengthStd	F47	bwdMsgDeltaLengthMode
	F3	fwdPacketDeltaLengthMax	F48	bwdMsgDeltaLengthSkewness
	F4	fwdPacketDeltaLengthMin	F49	bwdMsgDeltaLengthKurtosis
	F5	fwdPacketDeltaLengthVariance	F50	totalbwdMsgDeltaLength
	F6	fwdPacketDeltaLengthMedian	F51	msgDeltaLengthMean
	F7	fwdPacketDeltaLengthMode	F52	msgDeltaLengthStd
	F8	fwdPacketDeltaLengthSkewness	F53	msgDeltaLengthMax
	F9	fwdPacketDeltaLengthKurtosis	F54	msgDeltaLengthMin
	F10	totalfwdPacketDeltaLength	F55	msgDeltaLengthVariance
	F11	bwdPacketDeltaLengthMean	F56	msgDeltaLengthMedian
	F12	bwdPacketDeltaLengthStd	F57	msgDeltaLengthMode
	F13	bwdPacketDeltaLengthMax	F58	msgDeltaLengthSkewness
	F14	bwdPacketDeltaLengthMin	F59	msgDeltaLengthKurtosis
	F15	bwdPacketDeltaLengthVariance	F60	totalMsgDeltaLength
	F16	bwdPacketDeltaLengthMedian	F61	fwdMQTTDeltaLengthMean
	F17	bwdPacketDeltaLengthMode	F62	fwdMQTTDeltaLengthStd
	F18	bwdPacketDeltaLengthSkewness	F63	fwdMQTTDeltaLengthMax
	F19	bwdPacketDeltaLengthKurtosis	F64	fwdMQTTDeltaLengthMin
	F20	totalbwdPacketDeltaLength	F65	fwdMQTTDeltaLengthVariance
	F21	PacketDeltaLengthMean	F66	fwdMQTTDeltaLengthMedian
	F22	PacketDeltaLengthStd	F67	fwdMQTTDeltaLengthMode
	F23	PacketDeltaLengthMax	F68	fwdMQTTDeltaLengthSkewness
	F24	PacketDeltaLengthMin	F69	fwdMQTTDeltaLengthKurtosis
	F25	PacketDeltaLengthVariance	F70	totalfwdMQTTDeltaLength
	F26	PacketDeltaLengthMedian	F71	bwdMQTTDeltaLengthMean
	F27	PacketDeltaLengthMode	F72	bwdMQTTDeltaLengthStd
	F28	PacketDeltaLengthSkewness	F73	bwdMQTTDeltaLengthMax
	F29	PacketDeltaLengthKurtosis	F74	bwdMQTTDeltaLengthMin
	F30	totalPacketDeltaLength	F75	bwdMQTTDeltaLengthVariance
	F31	fwdMsgDeltaLengthMean	F76	bwdMQTTDeltaLengthMedian
	F32	fwdMsgDeltaLengthStd	F77	bwdMQTTDeltaLengthMode
	F33	fwdMsgDeltaLengthMax	F78	bwdMQTTDeltaLengthSkewness
	F34	fwdMsgDeltaLengthMin	F79	bwdMQTTDeltaLengthKurtosis
	F35	fwdMsgDeltaLengthVariance	F80	totalbwdMQTTDeltaLength
	F36	fwdMsgDeltaLengthMedian	F81	MQTTDeltaLengthMean
	F37	fwdMsgDeltaLengthMode	F82	MQTTDeltaLengthStd
	F38	fwdMsgDeltaLengthSkewness	F83	MQTTDeltaLengthMax
	F39	fwdMsgDeltaLengthKurtosis	F84	MQTTDeltaLengthMin
	F40	totalfwdMsgDeltaLength	F85	MQTTDeltaLengthVariance
	F41	bwdMsgDeltaLengthMean	F86	MQTTDeltaLengthMedian
	F42	bwdMsgDeltaLengthStd	F87	MQTTDeltaLengthMode
	F43	bwdMsgDeltaLengthMax	F88	MQTTDeltaLengthSkewness
	F44	bwdMsgDeltaLengthMin	F89	MQTTDeltaLengthKurtosis
	F45	bwdMsgDeltaLengthVariance	F90	totalMQTTDeltaLength
Abs Delta-Length Based	F91	fwdPacketAbsDeltaLengthMean	F136	bwdMsgAbsDeltaLengthMedian
	F92	fwdPacketAbsDeltaLengthStd	F137	bwdMsgAbsDeltaLengthMode
	F93	fwdPacketAbsDeltaLengthMax	F138	bwdMsgAbsDeltaLengthSkewness
	F94	fwdPacketAbsDeltaLengthMin	F139	bwdMsgAbsDeltaLengthKurtosis
	F95	fwdPacketAbsDeltaLengthVariance	F140	totalbwdMsgAbsDeltaLength
	F96	fwdPacketAbsDeltaLengthMedian	F141	msgAbsDeltaLengthMean

	F97	fwdPacketAbsDeltaLengthMode	F142	msgAbsDeltaLengthStd
	F98	fwdPacketAbsDeltaLengthSkewness	F143	msgAbsDeltaLengthMax
	F99	fwdPacketAbsDeltaLengthKurtosis	F144	msgAbsDeltaLengthMin
	F100	totalfwdPacketAbsDeltaLength	F145	msgAbsDeltaLengthVariance
	F101	bwdPacketAbsDeltaLengthMean	F146	msgAbsDeltaLengthMedian
	F102	bwdPacketAbsDeltaLengthStd	F147	msgAbsDeltaLengthMode
	F103	bwdPacketAbsDeltaLengthMax	F148	msgAbsDeltaLengthSkewness
	F104	bwdPacketAbsDeltaLengthMin	F149	msgAbsDeltaLengthKurtosis
	F105	bwdPacketAbsDeltaLengthVariance	F150	totalMsgAbsDeltaLength
	F106	bwdPacketAbsDeltaLengthMedian	F151	fwdMQTTAbsDeltaLengthMean
	F107	bwdPacketAbsDeltaLengthMode	F152	fwdMQTTAbsDeltaLengthStd
	F108	bwdPacketAbsDeltaLengthSkewness	F153	fwdMQTTAbsDeltaLengthMax
	F109	bwdPacketAbsDeltaLengthKurtosis	F154	fwdMQTTAbsDeltaLengthMin
	F110	totalbwdPacketAbsDeltaLength	F155	fwdMQTTAbsDeltaLengthVariance
	F111	PacketAbsDeltaLengthMean	F156	fwdMQTTAbsDeltaLengthMedian
	F112	PacketAbsDeltaLengthStd	F157	fwdMQTTAbsDeltaLengthMode
	F113	PacketAbsDeltaLengthMax	F158	fwdMQTTAbsDeltaLengthSkewness
	F114	PacketAbsDeltaLengthMin	F159	fwdMQTTAbsDeltaLengthKurtosis
	F115	PacketAbsDeltaLengthVariance	F160	totalfwdMQTTAbsDeltaLength
	F116	PacketAbsDeltaLengthMedian	F161	bwdMQTTAbsDeltaLengthMean
	F117	PacketAbsDeltaLengthMode	F162	bwdMQTTAbsDeltaLengthStd
	F118	PacketAbsDeltaLengthSkewness	F163	bwdMQTTAbsDeltaLengthMax
	F119	PacketAbsDeltaLengthKurtosis	F164	bwdMQTTAbsDeltaLengthMin
	F120	totalPacketAbsDeltaLength	F165	bwdMQTTAbsDeltaLengthVariance
	F121	fwdMsgAbsDeltaLengthMean	F166	bwdMQTTAbsDeltaLengthMedian
	F122	fwdMsgAbsDeltaLengthStd	F167	bwdMQTTAbsDeltaLengthMode
	F123	fwdMsgAbsDeltaLengthMax	F168	bwdMQTTAbsDeltaLengthSkewness
	F124	fwdMsgAbsDeltaLengthMin	F169	bwdMQTTAbsDeltaLengthKurtosis
	F125	fwdMsgAbsDeltaLengthVariance	F170	totalbwdMQTTAbsDeltaLength
	F126	fwdMsgAbsDeltaLengthMedian	F171	MQTTAbsDeltaLengthMean
	F127	fwdMsgAbsDeltaLengthMode	F172	MQTTAbsDeltaLengthStd
	F128	fwdMsgAbsDeltaLengthSkewness	F173	MQTTAbsDeltaLengthMax
	F129	fwdMsgAbsDeltaLengthKurtosis	F174	MQTTAbsDeltaLengthMin
	F130	totalfwdMsgAbsDeltaLength	F175	MQTTAbsDeltaLengthVariance
	F131	bwdMsgAbsDeltaLengthMean	F176	MQTTAbsDeltaLengthMedian
	F132	bwdMsgAbsDeltaLengthStd	F177	MQTTAbsDeltaLengthMode
	F133	bwdMsgAbsDeltaLengthMax	F178	MQTTAbsDeltaLengthSkewness
	F134	bwdMsgAbsDeltaLengthMin	F179	MQTTAbsDeltaLengthKurtosis
	F135	bwdMsgAbsDeltaLengthVariance	F180	totalMQTTAbsDeltaLength
Header Based	F181	srcIP	F194	isComplete
	F182	srcPort	F195	connectionRefused
	F183	dstIP	F196	keepAlive
	F184	dstPort	F197	cleanSession
	F185	clientID	F198	willFlag
	F186	clientIDLen	F199	willQoS
	F187	topic	F200	willmsgLen
	F188	topicLen	F201	willtopicLen
	F189	QoS	F202	passwordFlag
	F190	username	F203	reserved
	F191	usernameLen	F204	protocolName
	F192	password	F205	protocolNameLength
	F193	passwordLen	F206	connect_id
Count Based	F207	fwdPacketsCount	F224	connectPercentage

	F208	bwdPacketsCount	F225	connackPercentage
	F209	packetsCount	F226	publishPercentage
	F210	connectCount	F227	pubackPercentage
	F211	connackCount	F228	pubrecPercentage
	F212	publishCount	F229	pubrelPercentage
	F213	pubackCount	F230	pubcompPercentage
	F214	pubrecCount	F231	subscribePercentage
	F215	pubrelCount	F232	subackPercentage
	F216	pubcompCount	F233	unsubscribePercentage
	F217	subscribeCount	F234	unsubackPercentage
	F218	subackCount	F235	pingreqPercentage
	F219	unsubscribeCount	F236	pingrespPercentage
	F220	unsubackCount	F237	disconnectPercentage
	F221	pingreqCount	F238	interPublishTimeCount
	F222	pingrespCount	F239	fwdPacketDeltaTimeCount
	F223	disconnectCount	F240	bwdPacketDeltaTimeCount
Ratio-Based	F241	interPublishTimeRate	F259	totalbwdPacketDeltaLengthRate
	F242	fwdPacketDeltaTimeRate	F260	totalPacketDeltaLengthRate
	F243	bwdPacketDeltaTimeRate	F261	totalfwdPacketAbsDeltaLengthRate
	F244	fwdpacketsRate	F262	totalbwdPacketAbsDeltaLengthRate
	F245	bwdpacketsRate	F263	totalPacketAbsDeltaLengthRate
	F246	packetsRate	F264	totalfwdMQTTDeltaLengthRate
	F247	fwdTobwdPacketRatio	F265	totalbwdMQTTDeltaLengthRate
	F248	bwdTofwdPacketRatio	F266	totalMQTTDeltaLengthRate
	F249	fwdPacketLengthRate	F267	totalfwdMQTTAbsDeltaLengthRate
	F250	bwdpacketsLengthRate	F268	totalbwdMQTTAbsDeltaLengthRate
	F251	PacketLengthRate	F269	totalMQTTAbsDeltaLengthRate
	F252	fwdMsgLengthRate	F270	totalfwdMsgDeltaLengthRate
	F253	bwdMsgLengthRate	F271	totalbwdMsgDeltaLengthRate
	F254	MsgLengthRate	F272	totalMsgDeltaLengthRate
	F255	totalfwdMQTTLengthRate	F273	totalfwdMsgAbsDeltaLengthRate
	F256	totalbwdMQTTLengthRate	F274	totalbwdMsgAbsDeltaLengthRate
	F257	totalMQTTLengthRate	F275	totalMsgAbsDeltaLengthRate
	F258	totalfwdPacketDeltaLengthRate		
Size Based	F276	fwdPacketLengthMean	F324	bwdMsgLengthKurtosis
	F277	fwdPacketLengthStd	F325	totalbwdMsgLength
	F278	fwdPacketLengthMax	F326	MsgLengthMean
	F279	fwdPacketLengthMin	F327	MsgLengthStd
	F280	fwdPacketLengthVariance	F328	MsgLengthMax
	F281	fwdPacketLengthMedian	F329	MsgLengthMean
	F282	fwdPacketLengthMode	F330	MsgLengthStd
	F283	fwdPacketLengthSkewness	F331	MsgLengthMax
	F284	fwdPacketLengthKurtosis	F332	MsgLengthMin
	F285	totalfwdPacketLength	F333	MsgLengthVariance
	F286	bwdPacketLengthMean	F334	MsgLengthMedian
	F287	bwdPacketLengthStd	F335	MsgLengthMode
	F288	bwdPacketLengthMax	F336	MsgLengthSkewness
	F289	bwdPacketLengthMin	F337	MsgLengthKurtosis
	F290	bwdPacketLengthVariance	F338	totalMsgLength
	F291	bwdPacketLengthMedian	F339	fwdMQTTLengthMean
	F292	bwdPacketLengthMode	F340	fwdMQTTLengthStd
	F293	bwdPacketLengthSkewness	F341	fwdMQTTLengthMax
	F294	bwdPacketLengthKurtosis	F342	fwdMQTTLengthMin

	F295	totalbwdPacketLength	F343	fwdMQTTLengthVariance
	F296	PacketLengthMean	F344	fwdMQTTLengthMedian
	F297	PacketLengthStd	F345	fwdMQTTLengthMode
	F298	PacketLengthMax	F346	fwdMQTTengthSkewness
	F299	PacketLengthMin	F347	fwdMQTTLengthKurtosis
	F300	PacketLengthVariance	F348	totalfwdMQTTLength
	F301	PacketLengthMedian	F349	bwdMQTTLengthMean
	F302	PacketLengthMode	F350	bwdMQTTLengthStd
	F303	PacketLengthSkewness	F351	bwdMQTTLengthMax
	F304	PacketLengthKurtosis	F352	bwdMQTTLengthMin
	F305	totalPacketLength	F353	bwdMQTTLengthVariance
	F306	fwdMsgLengthMean	F354	bwdMQTTLengthMedian
	F307	fwdMsgLengthStd	F355	bwdMQTTLengthMode
	F308	fwdMsgLengthMax	F356	bwdMQTTLengthSkewness
	F309	fwdMsgLengthMin	F357	bwdMQTTLengthKurtosis
	F310	fwdMsgLengthVariance	F358	totalbwdMQTTLength
	F311	fwdMsgLengthMedian	F359	MQTTLengthMean
	F312	fwdMsgLengthMode	F360	MQTTLengthStd
	F313	fwdMsgLengthSkewness	F361	MQTTLengthMax
	F314	fwdMsgLengthKurtosis	F362	MQTTLengthMean
	F315	totalfwdMsgLength	F363	MQTTLengthStd
	F316	bwdMsgLengthMean	F364	MQTTLengthMax
	F317	bwdMsgLengthStd	F365	MQTTLengthMin
	F318	bwdMsgLengthMax	F366	MQTTLengthVariance
	F319	bwdMsgLengthMin	F367	MQTTLengthMedian
	F320	bwdMsgLengthVariance	F368	MQTTLengthMode
	F321	bwdMsgLengthMedian	F369	MQTTLengthSkewness
	F322	bwdMsgLengthMode	F370	MQTTLengthKurtosis
	F323	bwdMsgLengthSkewness	F371	totalMQTTLength
Delta-Time Based	F372	interPublishTimeMean	F390	fwdPacketDeltaTimeMax
	F373	interPublishTimeStd	F391	fwdPacketDeltaTimeMin
	F374	interPublishTimeMax	F392	fwdPacketDeltaTimeMedian
	F375	interPublishTimeMin	F393	fwdPacketDeltaTimeMode
	F376	interPublishTimeMedian	F394	fwdPacketDeltaTimeSkewness
	F377	interPublishTimeMode	F395	fwdPacketDeltaTimeKurtosis
	F378	interPublishTimeSkewness	F396	fwdPacketDeltaTimeVariance
	F379	interPublishTimeKurtosis	F397	fwdPacketDeltaTimeTotal
	F380	interPublishTimeVariance	F398	bwdPacketDeltaTimeMean
	F381	interPublishTimeTotal	F399	bwdPacketDeltaTimeStd
	F382	interPublishTimeMax	F400	bwdPacketDeltaTimeMax
	F383	interPublishTimeMin	F401	bwdPacketDeltaTimeMin
	F384	interPublishTimeMean	F402	bwdPacketDeltaTimeMedian
	F385	interPublishTimeStd	F403	bwdPacketDeltaTimeMode
	F386	interPublishTimeMax	F404	bwdPacketDeltaTimeSkewness
	F387	interPublishTimeMin	F405	bwdPacketDeltaTimeKurtosis
	F388	fwdPacketDeltaTimeMean	F406	bwdPacketDeltaTimeVariance
	F389	fwdPacketDeltaTimeStd	F407	bwdPacketDeltaTimeTotal
Time Based	F408	startTime	F411	fwdDuration
	F409	endTime	F412	bwdDuration
	F410	duration		

The feature categories summarized in Table 3 provide an organized overview of the types of information captured from MQTT flows, each targeting a specific behavioral aspect such as timing, size, frequency, and

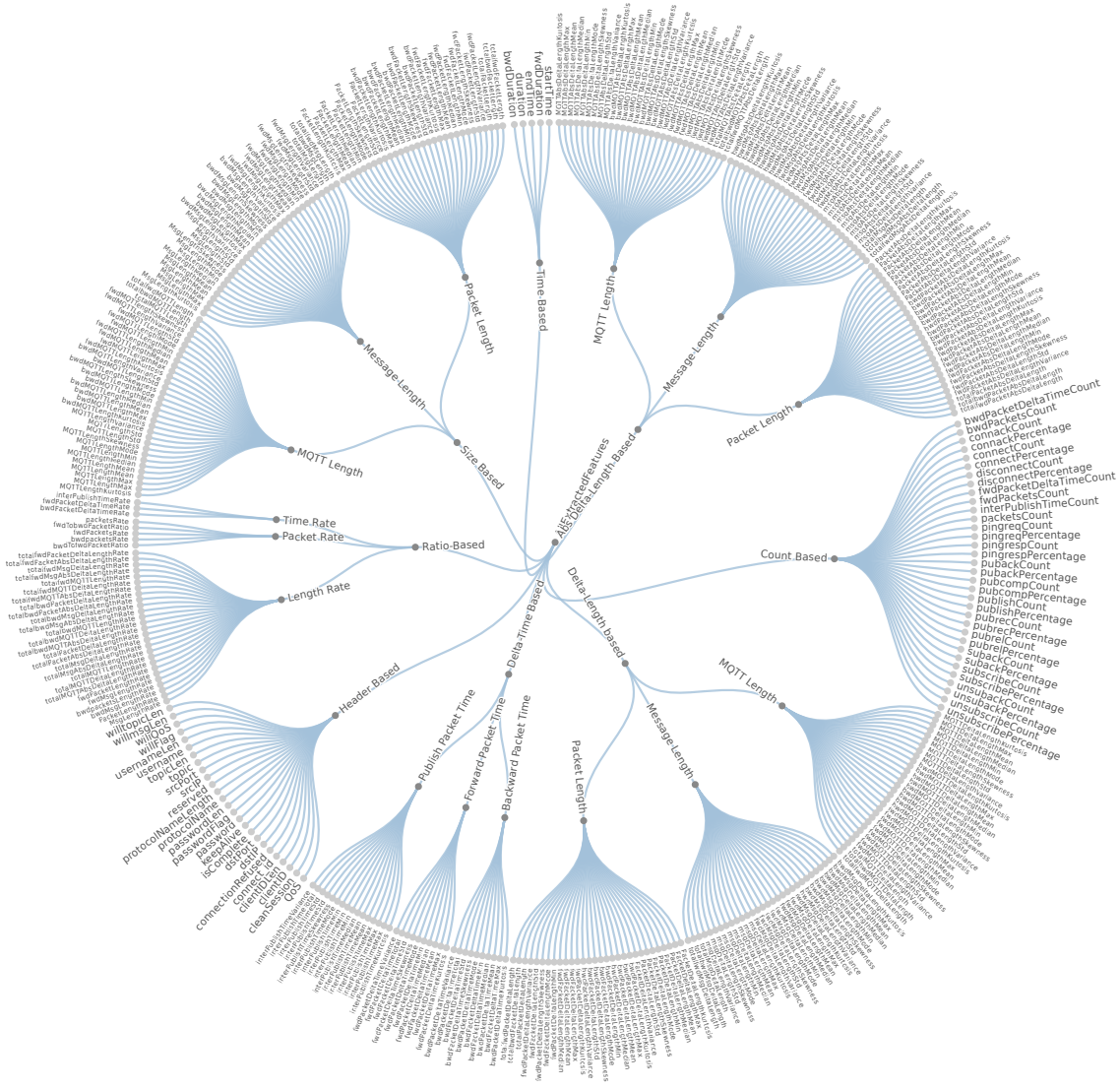

Figure 21: Hierarchical categorization of extracted features from MQTT traffic.

protocol structure.

Table 3: Summary of Feature Categories Extracted from MQTT Flows

Feature Category	Description
Delta-Time Based	Measures variations in time between packets, including inter-publish time, forward packet time, and backward packet time.
Delta-Length Based	Captures variations in packet length, message length, and MQTT message length across flows.
Abs Delta-Length Based	Focuses on absolute variations in packet, message, and MQTT length over time, useful for detecting abnormal payload fluctuations.
Time Based	Includes session-level timestamps such as start time, end time, and session duration for tracking the temporal behavior of MQTT flows.
Count Based	Tracks the frequency of MQTT control packets such as `CONNECT`, `PUBLISH`, `SUBSCRIBE`, `PINGREQ`, and `DISCONNECT`. Also includes the percentage of each control message type.
Ratio-Based	Evaluates MQTT traffic balance using metrics such as time rate, packet rate, and length rate to identify unusual traffic patterns.
Header-Based	Extracts MQTT protocol-specific fields such as source/destination IP and port, client ID, topic, QoS levels, authentication parameters, and protocol flags.
Size-Based	Measures packet size, message size, and MQTT payload length, including statistical properties such as mean, variance, skewness, and kurtosis.

3.2 Data Preprocessing

Data preprocessing is crucial in ensuring that the dataset is adequately prepared for machine learning tasks. In this study, we employed a series of preprocessing techniques to clean, balance, and structure the data, as outlined below:

3.2.1 Handling Missing Values

We addressed missing data by imputing values based on the data type. Categorical features with missing values were filled with the mode (the most frequent value) of the respective column, while numerical features were imputed using the median. This approach ensures that the dataset remains intact and prevents bias from being introduced during model training.

3.2.2 Addressing Class Imbalances

Class imbalances were identified as a potential issue, and to mitigate this, downsampling techniques were applied to ensure a more balanced distribution across the dataset. This helps prevent the model from being biased toward the majority class, enhancing its ability to generalize to all potential attack scenarios.

3.2.3 Categorical Feature Processing

Categorical features within the dataset were identified and processed for compatibility with machine learning algorithms. This allowed for the preservation of the inherent structure of the categorical variables, enabling more effective model training.

3.2.4 Feature Removal

Several features were removed from the dataset as part of the preprocessing pipeline. These included static or easily predictable fields such as IP addresses, ports, and identifiers, which are further discussed in Section **??**. These features could cause overfitting and may not contribute meaningfully to detecting sophisticated attack patterns. Removing them ensures the focus remains on more complex and dynamic features. After preprocessing, 378 features were retained for training and evaluating the TabNet model.

3.2.5 Feature-Label Separation and Label Encoding

The dataset was then split into features and labels. The target variable, `label`, was separated from the feature set, which was stored in `X`, while the `label` column was assigned to `y`. Labels were encoded using a `LabelEncoder`, transforming the categorical labels into a numerical format, making them suitable for machine learning algorithms. The encoder was also used to retrieve the class names for reference.

3.2.6 Data Splitting

The dataset was divided into training and testing sets, with 80% allocated for training and 20% for testing. Stratified splitting was applied to maintain the proportion of each class in both the training and testing sets. This ensures that the model is trained on a representative sample of all classes, preserving the distribution of labels.

Through these preprocessing steps, we ensured that the dataset was clean, balanced, and appropriately structured for machine learning tasks. Missing values were imputed, class imbalances were addressed, irrelevant features were removed, categorical data were encoded, and the dataset was split into training and testing sets. As a result, the dataset is now well-prepared for the subsequent analysis and model training.

3.3 TabNet-Based Learning Framework for IoT Intrusion Detection

The increasing complexity of network traffic in IoT environments demands robust and interpretable models that can handle high-dimensional data and provide reliable security insights. This section presents our TabNet-based end-to-end learning framework, a deep learning architecture that seamlessly integrates feature selection and classification for structured tabular data. We first provide a rationale for model selection, followed by an architectural overview and detailed explanation of TabNet's feature selection mechanism, interpretability, and application in classification.

3.3.1 Selection Strategy

Given the high dimensionality of our dataset, which comprises more than 400 flow-level features extracted from MQTT-based IoT traffic, effective feature selection is crucial to mitigate overfitting, reduce computational costs, and enhance model interpretability. Traditional machine learning pipelines typically decouple feature selection from classification by employing model-agnostic techniques, such as filter methods, wrapper methods, or post-hoc explainability tools like SHAP [44] and LIME [45]. Although these approaches

Table 4: Feature Selection Comparison

Method	*Feature Importance Basis*	*Strengths*	*Weaknesses*	*Best Use Cases*
Decision Trees	Gini impurity or information gain	Fast and interpretable	Biased towards high cardinality	Baseline models, small datasets
Random Forest	Averaged impurity reductions	Handles complex data well	Harder to interpret	High-dimensional data, structured datasets
Logistic Regression	Coefficient magnitudes of independent variables	Coefficient magnitudes of independent variables	Assumes linear relationships, less effective for complex patterns	Binary classification, healthcare, finance
Gradient Boosting	Gain or SHAP-based importance	High accuracy, handles feature interactions well	Computationally expensive, prone to overfitting	Tabular data, competitions, financial modeling
SHAP	Contribution of each feature to predictions	Highly interpretable	Computationally expensive	Explainability in critical applications
LIME	Locally approximated linear models	Explains individual predictions, model-agnostic	Not reliable for global interpretations	Model debugging, interpretability
Deep Learning	Weight contributions from hidden layers	Captures complex patterns and relationships	Captures complex patterns and relationships	Image, text, and sequential data analysis
TabNet	Attention-based feature selection across steps	Balances interpretability and performance, learns feature importance during training	Requires tuning, computationally expensive	Structured data with many irrelevant features

are widely used, they are less effective in scenarios with complex, nonlinear interactions and heterogeneous feature relevance across instances, conditions commonly found in intrusion detection tasks [46].

Initially, we explored explicit feature selection methods, including Decision Trees, Random Forests, and L1-regularized Logistic Regression [47], in conjunction with conventional classifiers. However, these approaches rely on global feature importance estimates that do not capture the instance-specific variability inherent in network intrusion patterns. Moreover, they lack the flexibility to adapt feature relevance dynamically at inference time.

Through empirical evaluation and review of recent advancements in tabular deep learning, we identified TabNet [48] as a more suitable alternative. TabNet employs a sequential attention mechanism that enables instance-wise, sparse feature selection as an integral part of its training process. Rather than assigning a fixed importance score to each feature, TabNet adaptively selects a relevant subset of features for each input sample at each decision step. This dynamic selection strategy enhances the model's ability to capture complex, context-dependent patterns in high-dimensional data.

Importantly, TabNet's attention masks are embedded within its internal representation learning and are computed jointly with classification objectives. These masks are not globally consistent or easily extractable, making reusing them as feature selectors for external models impractical. Attempting to do so would undermine the context-aware nature of the learned representations and likely result in performance degradation.

For these reasons, we adopt TabNet as our framework's feature selector and primary classifier. This unified approach enables us to fully leverage TabNet's architecture for end-to-end learning, resulting in a robust, interpretable, and scalable solution for IoT intrusion detection.

3.3.2 Architecture Overview

TabNet [48] is a novel deep learning architecture designed explicitly for tabular data, often containing numerical and categorical features. Traditional deep learning models, such as multi-layer perceptrons (MLPs), face difficulties when applied to tabular datasets due to the absence of inductive bias suited to decision trees, which are more naturally interpretable. TabNet addresses this issue by employing sequential attention to select a subset of features at each decision step, making the model effective and interpretable.

Mathematically, TabNet operates by first applying a decision block to the input data $X \in \mathbb{R}^{n \times d}$, where n is

the number of data points and d is the number of features. The decision block uses an attention mechanism to focus on the most relevant features for each input instance. Let the output of the decision block at the k-th step be denoted as:

$$\mathbf{z}^{(k)} = \text{DecisionBlock}\left(\mathbf{X}, \mathbf{a}^{(k-1)}, \mathbf{S}^{(k-1)}\right) \tag{1}$$

where $\mathbf{a}^{(k-1)}$ represents the attention from the previous step, and $\mathbf{S}^{(k-1)}$ is the feature selection mask that focuses the model's attention on the most relevant features.
The output of each decision block is then processed by a sparse attention mechanism, ensuring that only a subset of features is utilized at each decision point. This sparse attention mechanism is mathematically represented as:

$$\mathbf{S}^{(k)} = \text{SparseAttention}\left(\mathbf{z}^{(k)}\right) \tag{2}$$

where $\mathbf{S}^{(k)} \in \mathbb{R}^{n \times d}$ is a binary mask that indicates the active features of the k-th decision block. This allows the model to focus on the most important features and ignore irrelevant ones, improving both performance and interpretability.

3.3.3 Feature and Attentive Transformers

To better understand how TabNet processes tabular data, we refer to the overall system architecture shown in Figure 19. Although the feature transformer and attentive transformer are not explicitly separated in this figure, they are key internal components of the TabNet model. These two modules work together at each decision step to select and convert the most relevant features into meaningful representations. The feature transformer generates rich embeddings of the input data. In contrast, the attentive transformer dynamically selects which subset of features should be used in the next step based on learned attention scores. This interaction allows TabNet to perform sequential and interpretable decision-making throughout the intrusion detection.

3.3.3.1 Feature Transformer

As illustrated in Figure 22, the feature transformer is responsible for learning complex representations of the input features. It consists of several layers, each made up of a fully connected (FC) layer, batch normalization (BN), and a gated linear unit (GLU) activation function. These layers are organized into two parts:

- **Shared layers:** These are used in every decision step and help reduce the number of parameters by sharing the same weights across steps.
- **Decision step-specific layers:** These are unique to each decision step and allow the model to adapt the representation depending on the current step.

Residual connections between some layers, with a scaling factor of 0.5, stabilize training and make optimization easier. This module plays a key role in transforming raw input data into higher-level features later used by the attention mechanism.

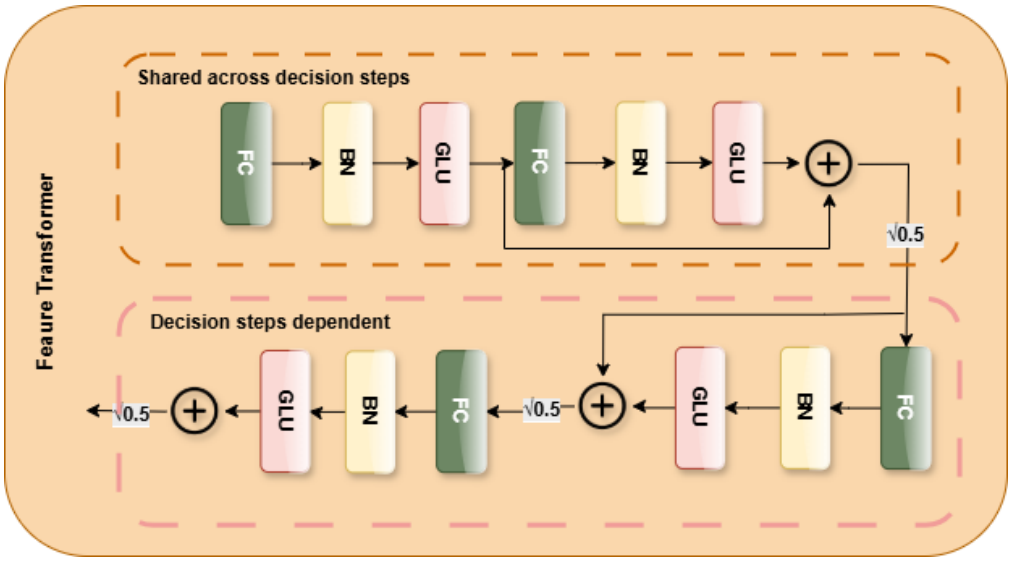

Figure 22: Feature transformer composed of shared and step-specific blocks using FC, BN, and GLU units.

3.3.3.2 Attentive Transformer

The attentive transformer, shown in Figure 23, is in charge of selecting which features to focus on at each decision step. It takes the output of the previous feature transformer and calculates attention scores to decide which features are most important. It includes the following components:

- A fully connected layer and batch normalization to prepare the input.
- A **prior scale** vector that tracks how much attention each feature has already received. This discourages the model from focusing on the same features repeatedly.
- A multiplication step that adjusts the feature scores based on the prior scale.
- A **sparsemax** activation function selects only a few essential features by setting the rest to zero. This makes the model more interpretable and helps prevent overfitting.

The result is a sparse feature mask that highlights only the most relevant features. This mask filters the input before passing it to the next decision step, allowing the model to attend to different feature subsets at each stage.
Together, these two modules enable TabNet to make powerful and interpretable decisions by dynamically transforming and selecting features during each decision step.

3.3.4 Feature Selection

The feature selection process in TabNet is based on an attention mechanism that dynamically selects the features most relevant to the current decision-making process. Unlike traditional models that use static feature selection, TabNet adapts the feature selection at each step, enabling it to focus on different subsets of features depending on the instance.

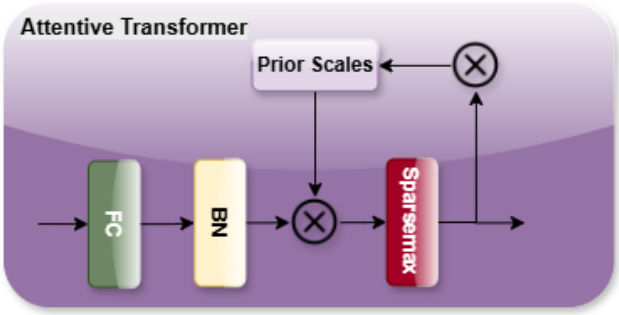

Figure 23: Attentive transformer using prior scales and sparsemax to generate sparse feature masks.

At each step of the model, a set of attention coefficients is calculated, representing the importance of each characteristic for the decision-making process. These attention coefficients generate a feature mask that controls which features are passed on to the next layer. Mathematically, the attention mechanism is computed as follows.

$$\mathbf{a}^{(k)} = \text{Softmax}\left(\frac{\mathbf{W}_a^{(k)}\mathbf{z}^{(k)}}{T}\right) \tag{3}$$

where $\mathbf{W}_a^{(k)}$ is the weight matrix for the attention mechanism, $\mathbf{z}^{(k)}$ is the output of the k-th decision block and T is the temperature parameter used to control the sharpness of the attention distribution. The attention mechanism produces a probability distribution over features, with higher values indicating that the feature is more critical for the current decision step.
The sparse attention mask is then obtained by applying a threshold to the attention coefficients.

$$\mathbf{S}^{(k)} = \mathbb{I}\left(\mathbf{a}^{(k)} > \varepsilon\right) \tag{4}$$

where $\mathbb{I}$ is the indicator function and ε is a threshold that controls the sparsity of the mask. This ensures that only the most relevant features are selected, resulting in better generalization and reduced overfitting.

3.3.5 Feature-Level Interpretability

One of the principal advantages of TabNet is its inherent interpretability, a rare characteristic among deep learning models applied to tabular data. While conventional neural networks such as multi-layer perceptrons (MLPs) often operate as opaque black-box systems, TabNet integrates a sequential attention mechanism that enables both *local* and *global interpretability*. This capability provides meaningful insights into which features influence predictions at the instance level and across the dataset.
TabNet's interpretability arises from its use of *sparse feature selection masks* and *attention coefficients* computed at each decision step. These components expose the model's internal decision-making process, making it suitable for security-critical applications such as IoT intrusion detection, where understanding the rationale behind predictions is vital for post-hoc analysis and real-time response.

3.3.5.1 Local Interpretability:

Local interpretability provides a fine-grained explanation of the model's behavior for individual samples. For each input instance $\mathbf{x}_n \in \mathbb{R}^d$, TabNet computes a sparse binary mask $\mathbf{S}_n^{(k)} \in \{0,1\}^d$ and a corresponding attention coefficient vector $\mathbf{a}_n^{(k)} \in \mathbb{R}^d$ at each of the T decision steps. These represent (i) which features were selected and (ii) the relative importance assigned to each selected feature.
The local feature importance for feature i in instance n is computed as:

$$I_{i,n} = \sum_{k=1}^{T} S_{i,n}^{(k)} \cdot a_{i,n}^{(k)} \tag{5}$$

To analyze class-specific patterns, we aggregate local feature importance values across all instances of a given class c. Let $\mathcal{C}_c$ denote the set of samples with label c, and $N_c = |\mathcal{C}_c|$ be the number of samples in class c. The average importance of feature i within class c is defined as:

$$\bar{I}_i^{(c)} = \frac{1}{N_c} \sum_{n \in \mathcal{C}_c} I_{i,n} \tag{6}$$

Ranking features by $\bar{I}_i^{(c)}$ allows us to identify which features are most discriminative for each class. For example, specific traffic patterns may consistently characterize a particular attack type, such as low-frequency connection attempts in scanning attacks or high throughput in flooding-based intrusions. The resulting class-wise feature profiles provide actionable insights into the model's decision boundaries. Further details on per-class top-ranked features are reported in Section 5.3.

3.3.5.2 Global Interpretability:

Global interpretability measures the average contribution of each feature across all predictions, offering a high-level overview of feature relevance. This is especially important for understanding which features consistently influence the dataset, thereby supporting model transparency and validation.
The global importance score for feature i is computed as the average of its local importance across all N samples:

$$I_i^{\text{global}} = \frac{1}{N} \sum_{n=1}^{N} I_{i,n} = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{T} S_{i,n}^{(k)} \cdot a_{i,n}^{(k)} \tag{7}$$

To facilitate comparison and visualization, these global importance values can be normalized as:

$$\tilde{I}_i^{\text{global}} = \frac{I_i^{\text{global}}}{\sum_{j=1}^{d} I_j^{\text{global}}} \tag{8}$$

The normalized score $\tilde{I}_i^{\text{global}} \in [0,1]$ indicates the relative contribution of each feature to the overall model behavior. Features with high global scores tend to play a consistent role in distinguishing between attack and benign traffic, while those with lower scores may be more situational or redundant.
Together, these interpretability analyses enhance trust in the model, support post-hoc forensics, and provide

a foundation for continuous feature refinement in evolving threat environments.

3.3.6 Classification Method

The final step in the learning framework involves classifying network traffic instances into predefined categories based on the representations learned during the sequential decision steps. After multiple rounds of attention-guided feature refinement, the latent representation produced by the final decision block, denoted as $\mathbf{z}^{(K)}$, captures high-level semantic information tailored to the task.

This representation is passed through a fully connected linear layer followed by a softmax activation to produce a class probability distribution:

$$\hat{y} = \text{Softmax}\left(\mathbf{W}_y \mathbf{z}^{(K)}\right) \tag{9}$$

where $\mathbf{W}_y \in \mathbb{R}^{C \times h}$ is the learned weight matrix for the output layer, C is the number of target classes, and h is the dimensionality of the final latent vector. The output $\hat{y} \in \mathbb{R}^C$ represents the predicted class probabilities for a given input sample.

The model is trained using categorical cross-entropy loss, which is minimized via mini-batch gradient descent. Early stopping based on validation performance prevents overfitting and ensures generalization. This end-to-end optimization strategy enables the model to simultaneously learn which features to focus on and how to classify instances, thereby streamlining the learning pipeline and enhancing robustness.

Crucially, the classifier benefits from the prior attention-based selection process, suppressing irrelevant or noisy features and enhancing the latent space's discriminative power. As a result, the classification module can make informed decisions even in the presence of redundant or high-dimensional inputs, such as the 404 flow-level features used in this study.

By integrating classification directly within the learning framework, the system avoids the risks associated with traditional modular designs, such as feature leakage or incompatible intermediate representations, while supporting deployment in real-time or resource-constrained environments typical of IoT infrastructures.

3.4 Confidence-Based Zero-Day Detection

While the classification model in this study is trained on known attack types, real-world IoT networks often exhibit novel or previously unseen intrusions, commonly referred to as zero-day attacks. These patterns are absent in the training data and may lead to erroneous or overconfident predictions if the model lacks a mechanism for detecting out-of-distribution behavior. To address this, we introduce a confidence-based inference strategy that enables the model to flag potentially novel inputs based on the uncertainty in its softmax output.

Given a trained model $f_\theta : \mathbb{R}^d \to [0,1]^C$, the output for any test input $\mathbf{x}_i \in \mathbb{R}^d$ is a class probability vector $\mathbf{p}_i = f_\theta(\mathbf{x}_i) = [p_{i,1}, p_{i,2}, \ldots, p_{i,C}]$, where each $p_{i,j}$ denotes the predicted probability of class j, and $\sum_{j=1}^{C} p_{i,j} = 1$. The index of the maximum probability, i.e., gives the predicted class label,

$$\hat{y}_i = \arg\max_j p_{i,j} \tag{10}$$

To quantify prediction confidence, we compute the maximum softmax probability:

$$\gamma_i = \max_j p_{i,j} \tag{11}$$

A threshold $\tau = 0.7$ is then defined to assess uncertainty. When the confidence score γ_i falls below this threshold, the prediction is considered unreliable and the corresponding input is flagged as potentially belonging to an unseen class. This is formalized as a binary decision rule:

$$\text{Unseen}(\mathbf{x}_i) = \begin{cases} 1 & \text{if } \gamma_i < \tau \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

We compute two auxiliary metrics to further evaluate the model's behavior under this detection regime. First, the unseen rate for each predicted class j, denoted as U_j, is defined as the proportion of samples classified as class j that were flagged as low-confidence:

$$U_j = \frac{\sum_{i=1}^{N} \mathbb{I}[\hat{y}_i = j \wedge \gamma_i < \tau]}{\sum_{i=1}^{N} \mathbb{I}[\hat{y}_i = j]} \tag{13}$$

Second, we measure the average predicted probability for each class j, given by:

$$\bar{p}_j = \frac{1}{N} \sum_{i=1}^{N} p_{i,j} \tag{14}$$

Together, these metrics provide a structured view of the model's sensitivity to distributional shifts and its ability to assess uncertainty. This confidence-based approach does not require retraining or external novelty detection modules and remains fully compatible with real-time deployment. Although conceptually simple, it enhances the system's robustness by enabling the identification of zero-day threats without labeled examples, positioning the model to act as both a supervised classifier and a lightweight anomaly detector in practice.

In summary, the proposed framework presents a comprehensive and scalable approach for IoT intrusion detection, leveraging flow-level feature engineering, interpretable deep learning, and confidence-based novelty assessment. Beginning with the design of MQTTFlowLyzer, we extract high-fidelity flow features that capture statistical, temporal, and protocol-specific characteristics of MQTT communication. These 404-dimensional representations are refined through a deep learning model that unifies sparse feature selection and classification, allowing the system to adaptively focus on the most informative aspects of each flow. The model's interpretability enables both localized and global understanding of decision behavior. At the same time, the inclusion of a softmax-based confidence evaluation mechanism extends the framework's capabilities to zero-day attack detection. This end-to-end design eliminates the reliance on separate feature selection modules or post hoc analysis, offering a principled, interpretable, and operationally viable solution for detecting known and previously unseen attacks in evolving IoT environments.

3.5 Hyperparameter Tuning

An extensive hyperparameter tuning process was conducted to ensure the TabNet model's optimal performance. Hyperparameters are settings defined before training begins and are critical in guiding how the model learns. Unlike internal parameters such as weights, which are learned during training, hyperparameters must be chosen manually or through automated search techniques. Selecting the right combination can significantly influence the model's accuracy, generalization ability, and training efficiency.
Key hyperparameters tuned for the TabNet model include:

- **n_steps:** Number of decision steps in the TabNet architecture. More steps allow deeper sequential attention, potentially capturing more complex feature interactions.
- **n_d and n_a:** Dimensionalities of the decision and attention layers, respectively. These control the model's learning capacity.
- **gamma:** Controls feature reuse by penalizing repeated selection of the same features across steps. Higher values encourage exploration of new features.
- **learning_rate:** Determines how quickly the model updates its weights. An appropriate learning rate is crucial to avoid underfitting or unstable training.
- **batch_size:** Number of training samples processed before the model updates its weights. Affects training stability and convergence speed.
- **optimizer and scheduler:** Define the weight update mechanism and learning rate decay strategy, respectively.

We utilized **Optuna**, an automated hyperparameter optimization framework that employs *Bayesian optimization* through Tree-structured Parzen Estimators (TPE). This approach models the performance of different hyperparameter combinations and guides the search toward more promising regions in the search space.
The tuning process followed these steps:

1. Optuna suggested an initial set of hyperparameter values randomly.
2. TabNet was trained using these values on the training dataset.
3. The model's performance was evaluated on a validation set using metrics such as macro F1-score.
4. Based on past results, Optuna refined its suggestions to focus on better-performing regions of the hyperparameter space.
5. This process was repeated over 50 trials to identify the most effective configuration.

Once the best hyperparameters were found, they were used to train the final TabNet model, which was evaluated on the test set. This automated tuning strategy reduced manual effort and ensured a more robust and efficient model configuration.
The final set of selected hyperparameters and their effects on performance are discussed in detail in the **Discussion** section under **Table 9**.

4 Experiments and Results

This chapter presents the results of applying the proposed intrusion detection framework to selected MQTT-based datasets. As there is no standard benchmark for MQTT intrusion detection, we first thoroughly reviewed available public datasets. Using a set of evaluation criteria, we identified and selected the most suitable datasets to ensure a realistic and comprehensive assessment of our model.
The experiments cover all system stages, including dataset selection, feature extraction, model training, and evaluation. Each of these steps is described in the following subsections. This section focuses on reporting the experimental results, while the analysis and interpretation of these results are presented in the subsequent section.
To prepare the data for training and testing, we applied downsampling to balance class distributions and used an 80–20 split to divide the data into training and test sets. The following subsections detail the datasets used, the criteria for their selection, the construction of the final dataset, and the performance of the proposed TabNet-based model.

4.1 Available Datasets

This section comprehensively reviews the available MQTT-based datasets, focusing on their strengths and limitations. We aim to identify the most suitable datasets for evaluating our proposed detection model by analyzing these datasets. Our review evaluates several key criteria, including the year of release, the inclusion of realistic traffic data, flow-based features, and the availability of benign data and PCAP files. Additionally, we will assess whether the datasets support multiclass classification and contain specific MQTT attacks such as DoS/DDoS, SlowITe, brute force, malformed data, flooding, and Man-In-The-Middle attacks. Each dataset will be weighted based on the number of attacks it includes, ensuring a fair and comprehensive comparison. This section aims to provide a detailed understanding of the datasets' capabilities and limitations, enabling us to select the most robust and comprehensive dataset for testing the resilience and effectiveness of our proposed MQTT intrusion detection model.

4.1.1 Datasets

This section provides an overview of the currently available MQTT-based datasets. Each dataset has unique characteristics and offers different data types and attack scenarios. By examining each dataset's strengths and weaknesses, we can better understand its applicability to our research.

- **NSL-KDD dataset:** In 2019,[49] employed the NSL-KDD dataset, an improved variant of the KDD Cup 1999 dataset. This dataset trains an artificial neural network (ANN) to detect network attacks on MQTT-enabled IoT devices. However, while the NSL-KDD dataset provides a comprehensive set of network traffic features, it does not include specific MQTT application data, concentrating instead on the TCP transport layer packets. As a result, it may be challenging to identify attacks that exploit MQTT's application layer protocol, indicating a potential vulnerability in defense against MQTT-specific threats.

- **ToN-IoT dataset:** After that, [50] at the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) released the ToN-IoT dataset, which aimed to address some of these gaps by providing a more detailed and diverse set of data from various IoT environments. However, even ToN-IoT fails to capture every MQTT packet, particularly the critical authentication and disconnection phases. It includes only a single TCP connection for all nodes, making it challenging to distinguish between different nodes at the transport layer. This highlights the need for more specialized datasets that fully capture MQTT-specific traffic and interactions to enhance the detection and mitigation of MQTT-specific threats.

- **BoT-IoT dataset:** The ACCS also created the BoT-IoT dataset, another valuable resource for IoT security research, which released citekoroniotis2019towards. The BoT-IoT dataset includes attack scenarios that utilize MQTT for communication with AWS services. However, the raw PCAP (packet capture) traffic data related to MQTT was not released. This means that while MQTT communication is part of the BoT-IoT dataset environment, these communications' specific details and data are not available for analysis. This lack of detailed MQTT traffic data limits its specificity for MQTT-focused research.

- **MQTT_AUD dataset:** The MQTT_AUD dataset is designed explicitly by [35], [51] to capture and analyze MQTT protocol traffic within IoT environments, and it is among the first MQTT-specific datasets. This dataset comprises 3 files containing normal and malicious traffic corresponding to specific vulnerabilities of the MQTT protocol. It evaluates the performance of intrusion detection systems and other security mechanisms tailored for MQTT traffic. However, the MQTT_AUD dataset is only available in pre-processed CSV format, and the raw PCAP (packet capture) data is not publicly released. This means researchers cannot manually process raw data or create custom CSV files from the raw traffic, which can limit the flexibility and depth of analysis for some specific research needs.

- **Artemis dataset:** In [37] paper, a DHT11 sensor connected to a Raspberry Pi is used to send temperature and humidity data. Although the PCAP file dataset is publicly released [52], it is not fully representative of a real network and is primarily beneficial for binary anomaly detection rather than multiclass classification.

- **MQTT-IoT-IDS2020 dataset:** [36] recorded the MQTT-IoT-IDS2020 dataset, which is publicly available [25] and simulates a realistic IoT network by using a camera feed, twelve MQTT sensors, and a broker. Several packet-based, unidirectional, and bidirectional flow features are used. This dataset provides pcap files but is not comprehensive in attack scenarios. In contrast, the dataset includes TCP-based attacks like the aggressive scan (Scan_A) and Sparta SSH brute-force (Sparta), as well as a UDP-based attack like the UDP scan (Scan_sU), the only MQTT-specific attack included is the MQTT brute-force (MQTT_BF).

- **MQTTset dataset:** Similarly, [34] presents the MQTTset dataset in . In the MQTTset dataset, the IoT-Flock tool [53] was utilized to simulate the realistic behavior of a home automation system. IoT-Flock, an open-source framework, allows the generation of diverse IoT traffic, including benign and

malicious patterns, by modeling various IoT devices and their interactions within a network. All the attacks included in the MQTTset dataset specifically target the MQTT protocol. These attacks are designed to exploit various vulnerabilities and weaknesses in the MQTT protocol, making the dataset valuable for developing and testing intrusion detection systems specifically for MQTT-based IoT environments. However, it doesn't capture abstraction levels of features, such as unidirectional flow and bidirectional flow features. The results of the previous work by [36] emphasize the importance of using flow-based features to differentiate MQTT-based attacks from benign traffic. In contrast, packet-based features are sufficient for traditional networking attacks. This limitation can impact the effectiveness of intrusion detection systems that rely on flow-based analysis to identify sophisticated attack patterns. Additionally, the MQTTset dataset captures traffic primarily from the broker's perspective, with the malicious node directly connected to the broker during attack phases. This focus on broker-side traffic means that attacks targeting publishers or subscribers are not captured, limiting the dataset's comprehensiveness in analyzing MQTT security vulnerabilities across all system components.

- **SENMQTT-SET dataset:** [54] proposed a novel IoT experimental setup created with heterogeneous sensors and real-time devices, resulting in the legitimate and attack dataset SENMQTT-SET. The SENMQTT-SET dataset includes three scenarios: No Attack, Attack on Subscriber, and Attack on Broker. However, it is constrained to basic connection flooding attacks and does not cover other MQTT-specific attacks. Moreover, the dataset is not publicly available, limiting its accessibility for further research.

- **DoS/DDoS-MQTT-IoT dataset:** Recently, [55] worked on the DoS/DDoS-MQTT-IoT dataset, which focuses on various types of denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks against the MQTT protocol. It provides a valuable resource for security researchers to develop and test countermeasures against DoS, and DDoS attacks in MQTT networks. The dataset includes five scenarios for each type of DoS and DDoS attack, but it is not comprehensive for other MQTT-based vulnerabilities and attacks.

- **CICIoMT2024:** [56] published CICIoMT2024, which includes extensive MQTT-based traffic, reflecting realistic conditions in healthcare IoMT environments. Various MQTT-specific attacks were executed. These attacks involve flooding the MQTT broker with connect or publish packets and sending malformed data to overwhelm the broker's processing capabilities. However, it is essential to note that the dataset only encompasses these five types of MQTT-based attacks and does not encompass the full spectrum of potential MQTT threats. In total, 18 attacks were executed against an IoMT topology comprising 40 IoMT devices, using a mix of real and simulated devices over MQTT.

4.1.2 Dataset Evaluation Criteria

Several key evaluation criteria were considered to ensure a comprehensive and practical analysis of MQTT datasets. Each criterion is assigned a specific weight based on its significance in assessing dataset applicability for intrusion detection and security research.

1. **Realistic Traffic Data**: The presence of realistic traffic data, generated from actual IoT devices and sensors, enhances the dataset's applicability for real-world security scenarios. The weight assigned to this criterion is **1**, as realistic traffic improves the dataset's relevance but does not solely determine its effectiveness.

2. **Flow-Based Features**: Datasets that include both packet-based and flow-based features allow for a more comprehensive network traffic analysis. Flow-based features capture the sequence of packet exchanges between endpoints, facilitating anomaly detection and behavioral analysis. Since these features significantly enhance network security assessments, the weight assigned to this criterion is **2**.

3. **Benign Data**: The availability of benign traffic is essential for establishing a baseline of normal network behavior and differentiating between legitimate and malicious activities. The weight assigned to this criterion is **2**, as it is a fundamental requirement for training effective intrusion detection models.

4. **PCAP File Availability**: Including PCAP files in a dataset allows researchers to conduct detailed packet analysis, extract additional features, and validate security mechanisms. Given its importance in forensic analysis and deep network inspection, this criterion is assigned a weight of **2**.

5. **Availability of MQTT Traffic Data**: Capturing MQTT-specific traffic within PCAP files ensures that the dataset accurately represents MQTT communication patterns. This is crucial for evaluating security mechanisms designed for MQTT-based IoT applications. Due to its direct relevance to MQTT security, the weight assigned to this criterion is **3**.

6. **Diverse Attack Types**: A dataset's ability to represent multiple attack types, including denial-of-service (DoS), brute force, and flooding attacks, makes it more suitable for testing generalizable intrusion detection models. Since diversity in attack representation is crucial for robust model evaluation, this criterion is assigned a weight of **3**.

7. **MQTT-Specific Attacks**: Some datasets explicitly include attacks that exploit MQTT protocol vulnerabilities, such as SlowITe attacks, publish flooding attacks, connect flooding attacks, and topic enumeration. The weight assigned to this criterion is **3**, as such attacks directly impact MQTT security assessments and provide valuable insights for protocol-specific defense mechanisms.

4.1.3 Dataset Scoring and Results

A weighted scoring methodology was applied based on the evaluation criteria outlined in the previous section to quantitatively assess the quality and applicability of the datasets. Each dataset was evaluated using a binary presence indicator ($\checkmark = 1$, $\times = 0$), where the presence of a criterion was multiplied by its respective weight. The final score for each dataset was computed using the following formula:

$$\text{Score} = \sum(\text{Weight}_i \times \text{Presence}_i)$$

where:

Table 5: Example Calculation of the Weighted Score for MQTT-IoT-IDS2020

Criterion	Weight	Presence (= 1, = 0)	Weighted Contribution
Realistic Traffic Data	1	0	$1 \times 0 = 0$
Flow-Based Features	2	1	$2 \times 1 = 2$
Benign Data	2	1	$2 \times 1 = 2$
PCAP File Availability	2	1	$2 \times 1 = 2$
Availability of MQTT Traffic Data	3	1	$3 \times 1 = 3$
Diverse Attack Types	3	1	$3 \times 1 = 3$
MQTT-Specific Attacks	3	1	$3 \times 1 = 3$
Total Weighted Score			**15**

Table 6: Evaluation of MQTT Attack Datasets Based on Key Criteria

Criteria	MQTT_UAD	Artemis	BOT-IOT	MQTT-IoT-IDS2020	TON-IoT	MQTTset	SENMQTT_SET	DoS/DDoS-MQTT-IoT	CICIOMT2024
Year	2019	2019	2019	2020	2020	2021	2021	2023	2024
Realistic Traffic Data	51	51	55	55	51	51	55	51	55
Flow-based features	55	51	51	51	51	51	55	51	51
Benign	51	51	51	51	51	51	51	51	55
Pcap file	55	51	55	51	51	51	55	51	51
MQTT Traffic Data	55	51	51	51	55	51	55	51	51
Diverse Attack Types	51	55	51	51	51	51	51	51	51
MQTT-Specific Attacks	51	55	55	51	55	51	51	51	51
Weighted Score	9	10	10	15	10	16	8	16	13

- Weight_i represents the assigned weight for criterion i.
- Presence_i is 1 if the dataset satisfies criterion i, otherwise 0.

Table 6 summarizes the computed scores for all datasets. To illustrate this methodology, we provide a step-by-step calculation of the weighted score for the `MQTT-IoT-IDS2020` dataset.
Table 5 details how each criterion contributes to the overall score by multiplying the respective weight by its binary presence indicator.
Applying the scoring formula:

$$\text{Weighted Score} = \sum(\text{Weight}_i \times \text{Presence}_i)$$

the total score for `MQTT-IoT-IDS2020` is computed as:

$$0+2+2+2+3+3+3 = 15$$

This calculation demonstrates that while `MQTT-IoT-IDS2020` meets several key evaluation criteria, including flow-based feature extraction, benign data availability, PCAP file inclusion, and diverse MQTT-specific attacks, it lacks realistic traffic data. The absence of real-world traffic samples may limit its applicability in specific intrusion detection scenarios where emulating practical network conditions is crucial. Despite this, the dataset remains a valuable resource for MQTT security research due to its diverse range of attack scenarios and comprehensive network traffic coverage. Future improvements could incorporate real sensor-generated traffic to enhance its realism and applicability in real-world IoT environments.
The results presented in Table 6 demonstrate considerable differences in dataset quality based on the defined evaluation criteria. The highest-scoring datasets, `MQTTset` and `DoS/DDoS-MQTT-IoT`, both achieved a score

of 16, followed closely by MQTT-IoT-IDS2020 with a score of 15. These datasets excel in multiple key areas, including the availability of pcap files, diverse MQTT-specific attacks, and capturing MQTT traffic, making them particularly well-suited for testing on MQTTFlowLyzer to detect intrusion and security analysis in MQTT environments. Conversely, datasets such as SENMQTT_SET (8) and Artemis (10) received lower scores due to their limited coverage of MQTT-specific attacks and absence of critical network traffic features such as flow-based analysis.

The findings emphasize the importance of datasets that incorporate realistic MQTT traffic and a diverse range of attack scenarios, as they provide a more robust foundation for developing effective security mechanisms. While datasets such as MQTTset and DoS/DDoS-MQTT-IoT offer significant research value, there remains room for improvement. Many existing datasets lack a comprehensive range of MQTT-specific attack types, including sophisticated evasion techniques and targeted protocol exploitation. Furthermore, some datasets do not include PCAP files, limiting researchers' ability to perform deep packet analysis.

Future research should enhance dataset quality by incorporating more attack types, ensuring realistic IoT-generated traffic, and maintaining high-resolution network traces in PCAP format. Additionally, establishing standardized evaluation benchmarks for MQTT security datasets could improve comparability and reproducibility across different studies. By adopting these improvements, future datasets can more effectively support the development of advanced intrusion detection systems designed explicitly for MQTT-based IoT networks.

4.2 New Augmented Dataset

Effective intrusion detection in MQTT-based IoT systems requires more than packet-level inspection. While most existing studies adopt packet-based IDS approaches, these often fail to capture the broader temporal and contextual behaviors that define modern attack patterns. As highlighted in the MQTT-IoT-IDS2020 study [25], a flow-based perspective is essential for identifying attacks that closely mimic normal protocol behavior. Although that work evaluated uniflow and biflow features, its analysis was limited to TCP-layer flows and did not incorporate MQTT-specific semantics, leaving a critical gap in application-layer intrusion modeling.

Given the increasing complexity of MQTT-based cyber threats, this limitation underscores the need for flow-based analysis that captures higher-level behavioral patterns within MQTT communication. We constructed a new augmented dataset to support this objective by processing raw PCAP files from selected MQTT-focused datasets and extracting protocol-aware flow features using our custom tool, MQTTFlowLyzer. As emphasized in recent work [26], developing robust and generalizable learning-based security frameworks critically depends on access to diverse and protocol-aware datasets. The goal was to generate a structured and comprehensive dataset suitable for training interpretable deep learning models, such as TabNet.

Three publicly available datasets were initially selected for inclusion: MQTTset, DoS/DDoS-MQTT-IoT, and MQTT-IoT-IDS2020. These were selected based on their high evaluation score in the previous section, availability of raw traffic captures, the inclusion of MQTT-specific attack scenarios, and their relevance to real-world IoT environments. Flow-level features were extracted from each using our proposed MQTTFlowLyzer tool. We then constructed a new, augmented dataset named BCCC-MQTT-IDS-2025. It covers

Table 7: BCCC-MQTT-IDS-2025 Dataset Overview

	Dataset	Category	Pcap Size	# Packets	# MQTT Flow
BCCC-MQTT-IDS-2025	**MQTTset**	Malformed	1,015 KB	10,924	2,925
		MalariaDoS	48,707 KB	130,223	188,957
		Capture_Flood	8,021 KB	613	1
		Bruteforce	1,119 KB	14,501	1,479
		Normal	1,068,044 KB	11,915,716	7,578,886
		Slowite	789 KB	9,202	1,024
	MQTT-IoT-IDS2020	Scan_A	15,837 KB	113,940	5,337
		Scan_sU	40,323 KB	255,058	15,921
		Normal	188,025 KB	1,070,577	79,770
		Sparta	3,311,657 KB	20,688,940	71,585
		Bruteforce	885,584 KB	10,010,556	1,000,166
	DoS-DDoS-MQTT-IoT	Normal	7,000 MB	66,800,000	5,224,765
		BF_DoS	3,000 MB	24,000,000	2,972,338
		BF_DDoS	5,000 MB	50,200,000	2,974,672
		Delay_DoS	3,000 MB	16,000,000	2,444,376
		Delay_DDoS	2,500 MB	25,100,000	3,023,806
		Sub_DoS	3,000 MB	33,500,000	1,124,873
		Sub_DDoS	3,000 MB	50,000,000	1,845,720
		WILL_DoS	3,000 MB	44,500,000	1,343,308
		WILL_DDoS	3,000 MB	50,000,000	1,845,720
		SYN_DoS	3,650 MB	31,500,000	2,867,338
		SYN_DDoS	3,000 MB	45,000,000	1,723,693

a diverse set of MQTT-specific threats, including brute-force authentication, malformed messages, SlowITe flooding, and various DoS/DDoS variants targeting MQTT's session and subscription mechanisms.

A detailed summary of the selected datasets, attack types, and generated flow statistics is provided in Table 7. This augmented dataset serves as the foundation for the experimental results discussed in the subsequent sections.

To ensure a balanced training process, we performed downsampling on the datasets, retaining 3,000 samples of the `normal` class from each dataset. This ensured that each class had an equal number of samples, addressing any class imbalance and providing a fair evaluation for the intrusion detection model.

4.3 Performance Results

In our analysis, we excluded the MQTT-IoT-IDS2020 dataset due to difficulties handling non-MQTT-specific attacks, which could have led to misleading results. The remaining datasets, MQTTset and DoS/DDoS-MQTT-IoT, were more compatible with our approach, ensuring the extraction of relevant and accurate features for model training.

Combining the attack categories from MQTTset and DoS/DDoS-MQTT-IoT, we formed a set of 9 attack categories, with regular traffic treated as the benign class. This approach enabled a comprehensive evaluation of MQTT-based intrusion detection systems. To address class imbalance, we downsampled by retaining 6,000 standard samples from the DoS/DDoS-MQTT-IoT dataset, combining attacks for each type. Addi-

Table 8: Label distribution, after cleaning and downsampling

Dataset	*Selected Category*	*# Flow*	*# Balanced Flow*
BCCC-MQTT-IDS-2025	Malformed	2,925	2925
	MalariaDoS	188,957	3000
	Bruteforce	1,479	1479
	Normal	12,803,651	3000
	Slowite	1,024	1024
	SYN_DoS/DDos	4,591,031	3000
	BF_DoS/DDos	5,947,010	3000
	Delay_DoS/DDos	5,468,182	3000
	Sub_DoS/DDos	2,970,593	3000
	WILL_DoS/DDos	3,189,028	3000

tionally, we kept 3,000 benign samples from MQTTset and downsampled the other attack categories in the DoS/DDoS-MQTT-IoT dataset. Furthermore, we removed the Capture_Flood category from the MQTTset dataset, as it contained only a single flow, which would not have contributed meaningful data to the analysis. After cleaning and downsampling, the final label distribution is in Table 8.

Following the preprocessing stage, the resulting balanced dataset was partitioned into distinct training and testing subsets to evaluate the TabNet model comprehensively. The training followed a structured methodology, as detailed in the preceding sections. To optimize model performance and ensure robustness, hyperparameter tuning was systematically conducted using the Optuna framework, which uses a Bayesian optimization strategy to identify the optimal set of hyperparameters that maximize classification accuracy.

The hyperparameters optimized included the number of sequential attention steps, the dimensions of the decision and attention embedding layers, regularization parameters such as the feature re-usage coefficient and sparsity regularization, the learning rate, batch size, and early stopping patience. Optimization was conducted through iterative training of the TabNet model across multiple epochs, assessing performance against validation data to mitigate the risk of overfitting. The optimal hyperparameter configuration identified by this rigorous search process is detailed in Table

The final model's effectiveness was evaluated using various standard performance metrics, including overall accuracy, a detailed classification report that encompassed precision, recall, and F1-score for each attack category, and a confusion matrix. These metrics provided a nuanced assessment of the model's ability to accurately distinguish between benign and malicious MQTT traffic.

The best-performing model achieved an overall accuracy of approximately 81.14%. Detailed metrics (Table 10) reveal excellent precision and recall for several attack categories, including "malformed", "brute force", and "malaria", indicating highly effective detection. However, categories such as "delay_dosddos" and "syn_dosddos" demonstrated relatively lower recall, highlighting areas for further model refinement and the importance of continued research into advanced detection methodologies.

Table 9: TabNet Hyperparameters - Default Ranges vs. Optimized Values (Optuna)

Hyperparameter	Description	Typical / Default	Best Value (Optuna)
n_d	Decision layer width	8–64 / 8	16
n_a	Attention layer width	8–64 / 8	16
n_steps	Attention steps	3–10 / 3	3
gamma	Feature reuse coefficient	1.0–2.0 / 1.5	1.4657
n_independent	Independent GLU layers	1–5 / 2	2
n_shared	Shared GLU layers	1–5 / 2	2
mask_type	Feature mask type	‘sparsemax’, ‘entmax’ / sparsemax	entmax
lambda_sparsity	Sparsity regularization	0–1.0 / 0.0001	0.00332
optimizer_fn	Optimizer used	Adam, SGD / Adam	torch.optim.Adam
optimizer_params	Learning rate settings	{’lr’: 2e-2, ’weight_decay’: 0}	{’lr’: 0.00098}
scheduler_fn	LR scheduler	StepLR / None	torch.optim.lr_scheduler.StepLR
scheduler_params	Scheduler config	None	{’step_size’: 10, ’gamma’: 0.9}
batch_size	Training batch size	256–2048 / 1024	512
virtual_batch_size	Ghost batch size	64–256 / 128	128
max_epochs	Max training epochs	50–200 / 200	128
patience	Early stopping threshold	10–30 / 15	20

Table 10: Classification Results of the Proposed Model

	precision	recall	f1-score	support
bf_dos/ddos	0.63	0.79	0.70	600
bruteforce	1.00	1.00	1.00	296
delay_dos/ddos	0.87	0.14	0.25	600
malaria_dos	1.00	0.99	0.99	600
malformed	0.99	1.00	1.00	585
normal	0.72	1.00	0.84	2400
slowite	1.00	1.00	1.00	205
sub_dos/ddos	0.99	0.78	0.87	600
syn_dos/ddos	0.69	0.36	0.48	600
will_dos/ddos	1.00	0.71	0.83	600
accuracy			0.81	7086
macro avg	0.89	0.78	0.80	7086
weighted avg	0.83	0.81	0.79	7086

4.4 Top-Ranked Features

After training the TabNet model on MQTT traffic data, we extracted feature importance scores to understand global and local decision behavior. Global importance scores highlight features that consistently influenced predictions across the dataset, offering insights into which traffic characteristics are most helpful in distinguishing between benign and malicious MQTT flows.

Table 11 presents the top 30 features ranked by global importance derived from TabNet's attention masks over the test set. These include MQTT protocol-specific attributes and statistical descriptors computed by MQTTFlowLyzer.

Table 11: Top 30 Most Important Features Identified by the TabNet Model

Rank	*Feature Name*	*Importance Score*
1	mqttdeltalengthmean	0.058185
2	bwdpacketlengthmode	0.057776
3	iscomplete	0.051889
4	topiclen	0.037335
5	msgdeltalengthmode	0.036577
6	fwdmqttdeltalengthvariance	0.033023
7	totalbwdmqttlength	0.030457
8	fwdpacketlengthvariance	0.030132
9	mqttlengthmax	0.025368
10	bwdmsglengthmedian	0.024256
11	totalfwdmqttdeltalength	0.023234
12	mqttdeltalengthmax	0.023079
13	fwdmqttlengthmean	0.021262
14	fwdmqttdeltalengthmean	0.020152
15	fwdpacketabsdeltalengthmean	0.019470
16	msglengthskewness	0.019133
17	packetlengthmax	0.017663
18	fwdmqttabsdeltalengthmode	0.017393
19	bwdmsglengthmin	0.016400
20	bwdmsglengthskewness	0.016226
21	qos	0.014693
22	msgdeltalengthmedian	0.014624
23	mqttabsdeltalengthkurtosis	0.013766
24	fwdpacketdeltatimekurtosis	0.013514
25	packetabsdeltalengthmedian	0.012221
26	pubrelpercentage	0.011252
27	bwdmqttlengthvariance	0.011066
28	publishcount	0.010695
29	fwdpacketdeltatimerate	0.010027
30	pubcomppercentage	0.009686

These findings highlight the critical role of MQTT-specific behavioral indicators in intrusion detection. Notably, features such as message length variability (`mqttdeltalengthmean`, `mqttdeltalengthmax`), session completeness (`iscomplete`), and topic hierarchy complexity (`topiclen`) demonstrate strong discriminative

power. Their statistical relevance underscores the necessity of incorporating protocol-aware characteristics for accurate threat modeling in MQTT-based environments. . Additionally, the statistical characteristics of forward and backward traffic flows—including packet length variations and inter-arrival timing—further contributed to the accuracy of intrusion detection. The combination of protocol-specific and statistical features proves effective for identifying known attacks and laying the groundwork for generalizing to zero-day threats through behavioral modeling.

4.5 Class-wise Behavioral Profiling

In addition to global interpretability, we used TabNet attention masks to perform local feature attribution, identifying which features influenced individual predictions. To summarize these insights at the class level, we computed the average local feature attribution across all samples belonging to a given class label.

Let A_{ij} represent the importance of feature i for sample j, as extracted from TabNet's attention explanation matrix. For each attack class c, we define the class-wise feature importance as:

$$I_i^{(c)} = \frac{1}{|D_c|} \sum_{\mathbf{x}_j \in D_c} A_{ij} \tag{15}$$

where D_c is the set of test samples labeled with class c. This formulation yields a class-specific feature profile vector $\mathbf{I}^{(c)}$ for each class, enabling the discovery of distinctive behavioral signatures.

The aggregated importance vectors for all classes were visualized using a heatmap, shown in Figure 24, where rows correspond to features and columns to attack classes. Each cell's intensity reflects a feature's average importance in predicting samples from that class.

This visualization reveals distinct behavioral signatures across MQTT attack categories, underscoring the relevance of tailored statistical features in classifying malicious activity. For the brute-force attack, two features emerge as highly discriminative: `mqttabsdeltalengthmedian`, which captures abrupt shifts in payload size, and `connectionrefused`, which directly signals failed connection attempts due to invalid credentials. These indicators align with the repetitive trial-and-error nature of credential-based intrusions, consistent with the findings of [34] in the MQTTset dataset.

In contrast, the `sub_dos/ddos` attack, which abuses valid credentials to flood the broker with invalid subscription or publish requests [55], shows a dominant association with `mqttlengthvariance`, reflecting significant fluctuations in message lengths—an indicative pattern of protocol-level exploitation aimed at overwhelming system resources. Notably, `interpublishtimestd`, defined as the standard deviation of time differences between consecutive `PUBLISH` packets within a flow, captures the irregular timing patterns typical of such flooding behavior. Additional features such as `fwdmqttabsdeltalengthmin` further emphasize abrupt size changes in forward traffic, echoing the behavior observed in the DoS/DDoS-MQTT-IoT dataset. These insights validate the effectiveness of distributional and temporal MQTT-specific features in enabling interpretable, class-aware intrusion detection.

Using TabNet's inherently interpretable structure, our approach supports high classification performance and the generation of human-understandable insights, which are critical for practical security deployment and zero-day threat investigation.

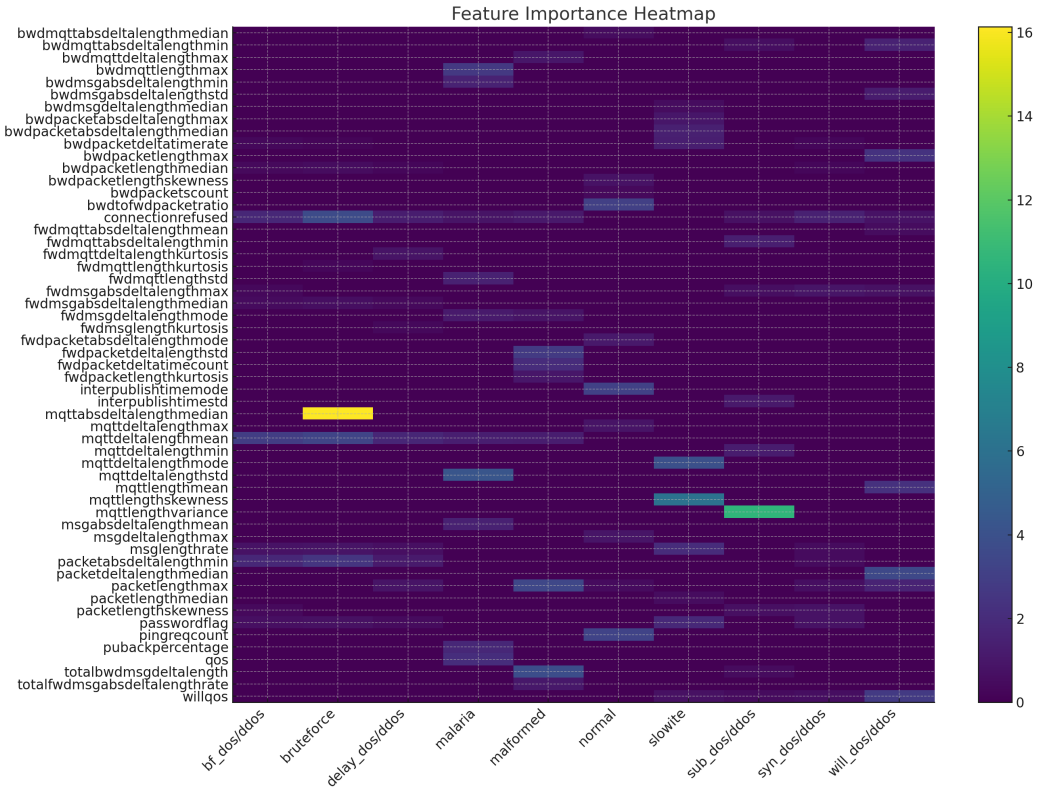

Figure 24: Heatmap of class-wise local feature importance based on TabNet's attention mechanism. Brighter cells denote a higher influence of the corresponding feature for that class.

4.6 Zero-Day Detection Scenario

To assess the generalization capacity and robustness of the proposed TabNet-based intrusion detection framework, we conducted a zero-day evaluation using the MQTT-IoT-IDS2020 dataset [36]. This dataset includes an MQTT-specific brute-force attack (MQTT-BF) that was deliberately excluded from the training phase. By evaluating the model on this unseen attack, we aimed to measure its ability to recognize novel threats solely based on previously learned behavioral characteristics.

As noted in the original study [36], MQTT-based attacks often exhibit high syntactic similarity to benign traffic due to their reliance on legitimate protocol commands such as `PUBLISH` and `SUBSCRIBE`:

> "MQTT-based attacks have similar characteristics to benign MQTT communication. Since MQTT-based attacks rely on the available MQTT communication commands (i.e., publish, subscribe, etc), it is challenging to discriminate attacks from normal operations where the same commands are used." [36]

Despite this inherent challenge, our model successfully detected the unseen MQTT-BF traffic with a classification accuracy of 99%. This strong result is attributed to the flow-level features extracted by *MQTTFlowLyzer*, which capture behavioral patterns such as connection frequency, authentication failure rates, and session dynamics, offering a richer representation than solely packet-level attributes.

The MQTT-BF samples were generated using MQTT PWN [57], which emulates credential-guessing attacks during the authentication phase. This contrasts with the brute-force attack in the training set, which was

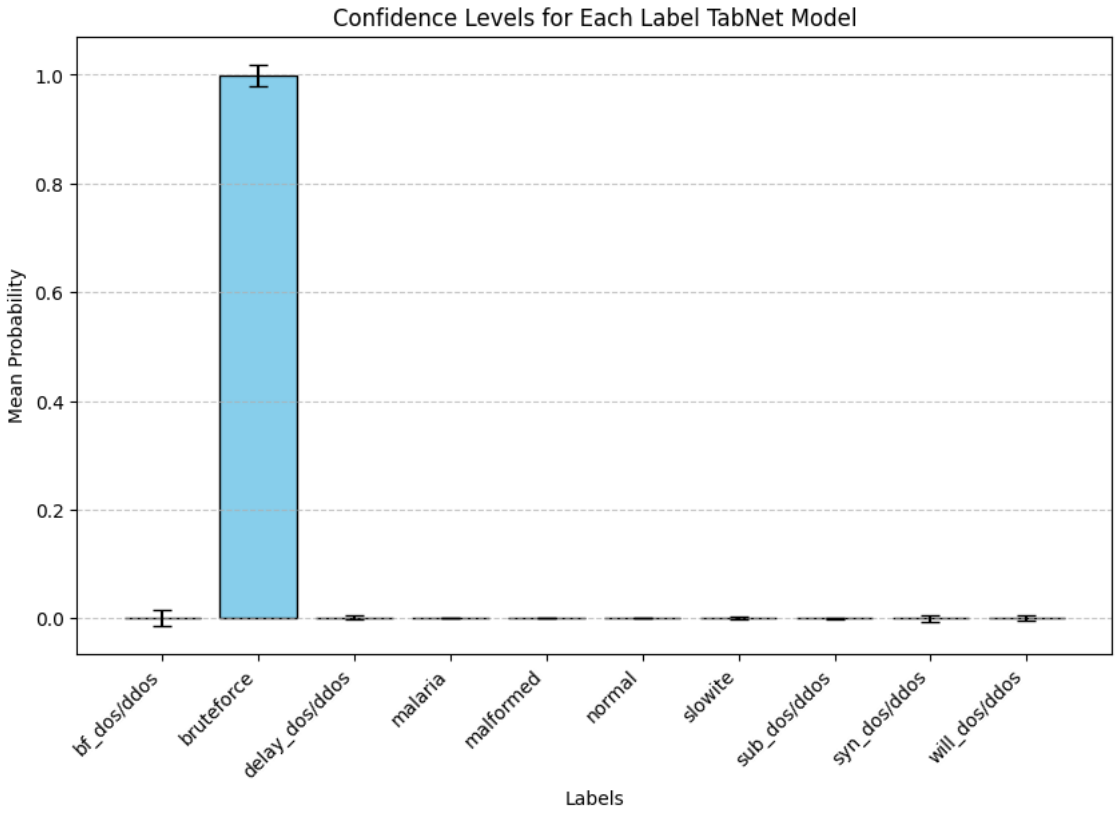

Figure 25: Confidence level of the TabNet model on the MQTT-BF zero-day attack.

conducted using MQTTSA [58] and the `rockyou.txt` password list [59]. While the tools and strategies differed, both attack types targeted the MQTT authentication mechanism and induced similar session-level anomalies. These shared characteristics enabled the model to map the unfamiliar MQTT-BF traffic to the learned brute-force class without direct exposure.

Additionally, connect-flooding attacks bf_dosddos in the training set contributed to the model's understanding of relevant flow anomalies, such as abnormal initiation rates and frequent failed connections. These parallels enhanced TabNet's ability to generalize from related behavioral patterns.

We applied the confidence-based zero-day framework described in Section 3.4 to validate the detection reliability further. This approach flags predictions with a softmax confidence below 0.7 as potentially novel. TabNet consistently produced high-confidence predictions across most MQTT-BF samples, supporting its reliability in handling distributional shifts.

These findings confirm that TabNet and flow-level behavioral profiling can detect novel attacks by learning abstract threat patterns rather than relying on exact signature matches.

5 Analysis and Discussion

This section presents a detailed evaluation of the proposed intrusion detection framework, focusing on behavioral patterns captured from MQTT session-level traffic. By systematically analyzing these flows, we aim to uncover insights into MQTT-based IoT communications' operational dynamics and potential vulnerabilities.

The following analysis integrates quantitative performance metrics and qualitative interpretability insights to demonstrate the efficacy of the TabNet-based classification pipeline. By leveraging enriched flow representations and confidence-aware inference, our system provides accurate and interpretable intrusion detection tailored to the unique characteristics of MQTT communication. These findings substantiate the framework's applicability to real-world deployments, where maintaining detection robustness and computational efficiency is paramount.

5.1 Interpretation of the Most Important Features

The MQTT flow-based and statistical features identified by the TabNet model as highly influential are logically significant. They provide valuable insights into the subtle behavioral differences that distinguish malicious from benign MQTT traffic. This subsection provides an in-depth interpretation of why these specific features are essential and how they contribute to effective intrusion detection.

MQTT, a lightweight protocol designed for IoT applications, typically produces uniform and indistinguishable traffic at first glance. Therefore, nuanced feature analysis becomes essential for detecting diverse attack categories, such as brute force attempts, malformed packets, flooding, and various denial-of-service scenarios. MQTTFlowLyzer's comprehensive approach to defining MQTT flows—based on attributes such as source and destination IPs and ports, client identifiers, topics, and accurate timestamps—enables the detection of minor but meaningful variations in traffic patterns.

Specifically, the following insights justify the selection of key features:

- **MQTT Delta-Length Mean and Max (*mqttdeltalengthmean*, *mqttdeltalengthmax*):** Changes in MQTT message sizes between consecutive packets effectively identify anomalies related to flooding or malformed packet attacks. Such attacks often manifest as irregular and frequent fluctuations in packet lengths, making these features critical.

- **Backward Packet Length Mode (*bwdpacketlengthmode*):** Consistent packet lengths in broker-to-client communications typically indicate automated or scripted attack patterns, significantly contrasting with the natural variability of legitimate traffic.

- **Session Completeness (*iscomplete*):** This feature flags whether MQTT sessions properly conclude using standard protocol commands (e.g., DISCONNECT). Malicious traffic often lacks these normal terminations, particularly in abrupt or flooding attack scenarios.

- **Topic Length (*topiclen*):** Extreme variations in topic length, either unusually short, randomized, or excessively long, can indicate scanning, enumeration attempts, or other malicious behaviors, differentiating them clearly from typical MQTT use cases.

- **Message Delta-Length Metrics (*msgdeltalengthmode*, *msgdeltalengthmedian*):** Systematic patterns in message length variations typically reflect brute-force or denial-of-service attacks, characterized by repeated or deliberately structured payloads.

- **Forward MQTT Delta-Length Variance (*fwdmqttdeltalengthvariance*, *totalfwdmqttdeltalength*):** High variance in forward-direction MQTT message lengths indicates random or erratic communication, commonly seen in flooding attacks, while low variance can suggest scripted brute-force attempts.
- **Statistical Moments (Skewness, Kurtosis, Variance) of Lengths:** Statistical descriptors (*msglengthskewness*, *mqttabsdeltalengthkurtosis*) capture subtle irregularities in the distribution of MQTT payload sizes and inter-packet timing, differentiating maliciously crafted traffic from typical network behavior.
- **Quality of Service (*qos*):** Attackers often manipulate QoS settings to maximize throughput during flooding, resulting in distinct QoS patterns compared to legitimate sessions.
- **Control Packet Metrics (*publishcount*, *pubrelpercentage*, *pubcomppercentage*):** These features quantify the distribution and frequency of MQTT control messages. Abnormal usage patterns of these messages often signal brute-force or flooding activities.
- **Timing-Based Features (*fwdpacketdeltatimekurtosis*, *packetabsdeltalengthmedian*):** These features effectively capture temporal irregularities, which identify attacks that exhibit distinct timing patterns, including high-frequency flooding or slow, prolonged attempts.

The multi-dimensional characterization provided by these features is essential for accurately differentiating MQTT traffic categories. The richness and specificity of the extracted features, validated through TabNet's interpretability-driven feature selection, demonstrate the method's efficacy in identifying MQTT-specific attacks, reinforcing its practical value in IoT cybersecurity frameworks.

5.2 Zero-Day Attack Analysis Across Baseline Models

To better interpret the performance of the proposed TabNet model, we conducted a comparative analysis of several traditional machine learning and deep learning classifiers under a zero-day detection scenario. Each model was evaluated using the unseen MQTT-BF attack samples from the MQTT-IoT-IDS2020 dataset. The goal was to assess how confidently and accurately these models classify novel attacks based on behavioral generalization. The results, summarized in Figure 26, illustrate each model's mean confidence and variability in the prediction across all classes.

The Random Forest classifier tended to misclassify the MQTT-BF attack as a slowite attack, assigning it the highest average probability of 0.4629 among all predicted labels. This misclassification indicates that, while Random Forest could detect anomalous behavior, it lacked the contextual understanding required to associate the pattern with the correct brute-force category. The relatively high variance (0.1466) further suggests a degree of uncertainty in its predictions.

XGBoost and LightGBM demonstrated better alignment with the accurate brute-force label, with mean confidence values of 0.7840 and 0.6786, respectively. However, these models also exhibited high standard deviations (0.2993 for XGBoost and 0.2620 for LightGBM), indicating inconsistent performance and less reliable predictions across samples. Their ability to generalize was present but unstable, reflecting their sensitivity to variations within the unseen attack flows.

While Logistic Regression offers limited effectiveness in assigning part of its prediction probability to the correct class (brute-force, 0.3322), it also distributes a more significant proportion to the bf_dos/ddos label (0.5824). This behavior illustrates confusion between flooding and brute-force behaviors, likely due to shared flow characteristics such as increased connection attempts, yet the model failed to distinguish between them clearly.

Gradient Boosting exhibited nearly identical behavior to LightGBM, with moderate emphasis on the correct label but a similarly dispersed confidence pattern. This may reflect their shared tree-based architecture, which tends to prioritize shallow pattern matching over deeper behavioral abstraction.

Deep learning baselines, including the Multilayer Perceptron (MLP) and standard Neural Network, showed the weakest generalization. Both models predominantly misclassified the MQTT-BF attack as will_dos/ddos, with extremely high average confidence scores (0.9960 and 0.9976, respectively). Despite high certainty, this misclassification suggests overfitting to dominant traffic patterns in the training data and a failure to infer nuanced session-level behavioral similarities.

Collectively, these results highlight a significant limitation of conventional ML and DL models in zero-day scenarios. While some models can partially capture behavioral patterns, they generally struggle to make confident and accurate classifications when presented with novel attacks. This limitation becomes particularly evident in protocol-level similarity between benign and malicious MQTT traffic, where reliance on packet-level features alone cannot distinguish between them. These findings further justify using models integrating flow-level behavior and interpretable attention mechanisms for robust intrusion detection in IoT environments.

5.3 Class-wise Feature Attribution and Behavioral Signatures

An essential advantage of TabNet is its ability to provide per-instance and per-class explanations through sparse attention masks. Unlike global feature importance metrics, which offer an overall ranking of influential features but often obscure class-specific or context-dependent patterns, TabNet enables localized interpretability. In this study, we deliberately focused on local feature attributions to better capture the diverse behaviors exhibited by different types of attacks. By aggregating these local importance scores across samples within each class, we identified distinguishing feature sets that characterize each attack category.

We extracted the top 10 most influential features for each class, as illustrated in Figure 27. These class-wise profiles reflect the model's internal reasoning and provide concrete behavioral signatures that differentiate between attack types, making them valuable for forensic analysis and real-time detection pipelines.

A cross-class comparison of the most influential feature for each attack category reveals how the MQTTFlowLyzer-TabNet model captures the unique behavioral patterns underlying each type of MQTT activity. In the case of `bruteforce` attacks, the model assigns the highest importance to `connectionrefused`, directly indicating repeated failed connection attempts due to invalid login credentials. For the `bf_dos/ddos` class, `mqttdeltalengthmean` stands out, capturing the regularity—or disruption—of publish message timing as attackers attempt to overload the broker with frequent requests. Meanwhile, the `malformed` class is most influenced by `totalbwdmsgdeltalength`, highlighting inconsistencies in message timing and size

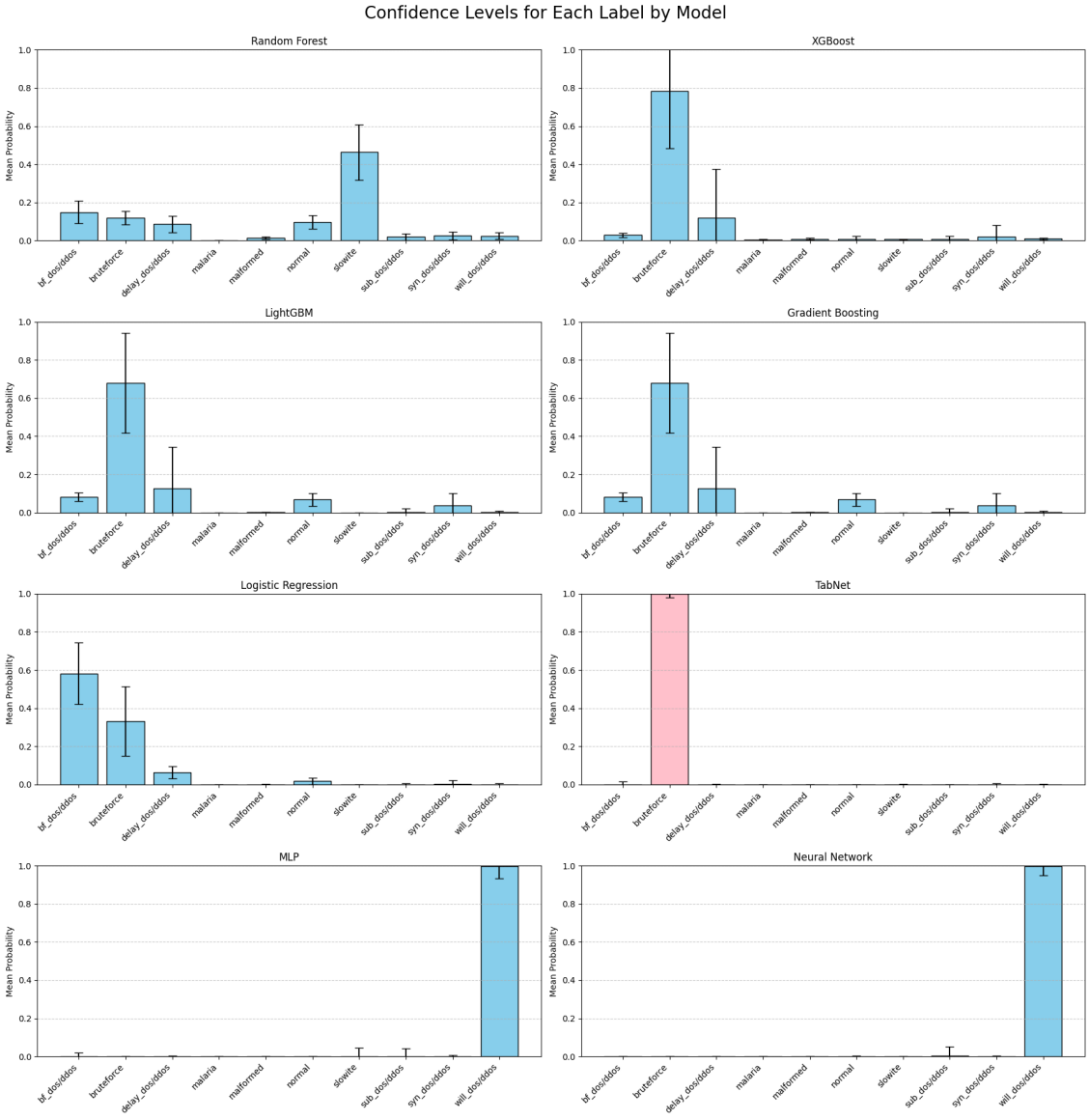

Figure 26: Confidence levels of various ML and DL models on the zero-day MQTT-BF attack.

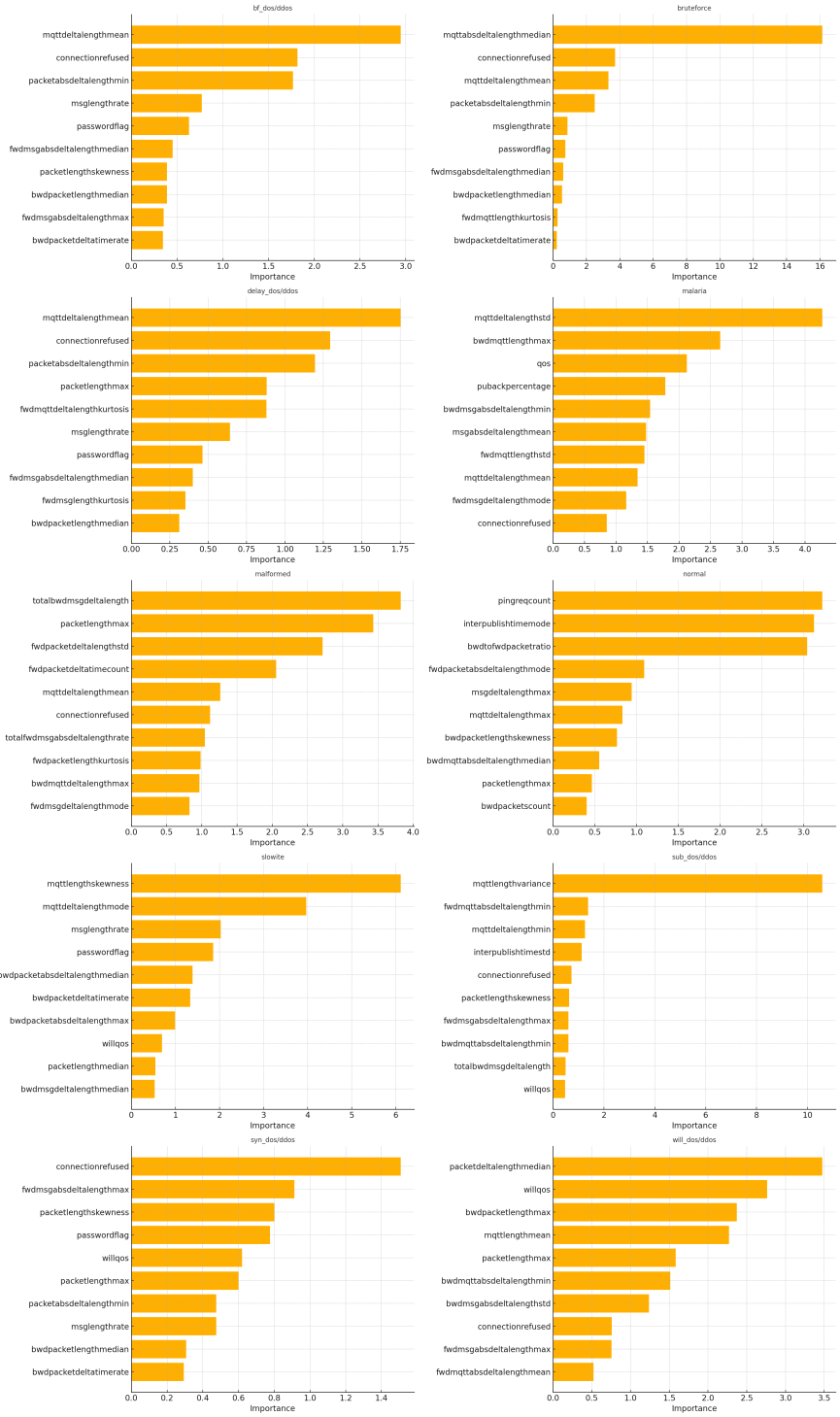

Figure 27: Feature Importance bar chart by Class.

in the backward direction, often caused by incorrectly formatted packets. Finally, for slowite traffic, mqttlengthskewness is the most essential feature, capturing subtle but consistent deviations in message size distributions that signal low-rate or stealthy attacks. These feature-level insights confirm that each attack type leaves a distinct statistical and protocol-level footprint, which the model successfully learns to distinguish through targeted feature attributions.

In detail, for the normal behaviour, the feature attribution plot emphasizes characteristics associated with protocol-compliant and stable communication patterns. The most influential feature, pingreqcount, represents the frequency of MQTT keep-alive requests routinely exchanged in legitimate sessions to maintain broker connectivity. This behavior is typically observed in idle or periodically active clients, thus strongly indicative of benign operation. Similarly, interpublishtimemode captures the modal time interval between consecutive publish packets, a feature that reflects the regularity and periodicity inherent to sensor-driven data transmission in non-malicious environments. The significance of bwdtofwdpacketratio highlights the importance of balanced bidirectional traffic, commonly observed in standard MQTT interactions. Additionally, features such as fwdpacketabsdeltalengthmode and msgdeltalengthmax capture payload and timing consistency across flows, which are more common in regular traffic than in the unpredictable and inconsistent patterns often seen during attacks. These features describe the normal class as having regular timing, balanced communication, and behavior that aligns with standard MQTT usage. This makes it easier for the model to separate regular sessions from suspicious or potentially harmful ones.

Also, the behavioral pattern of the malariaDos attack is shaped by its goal of overloading the MQTT broker by sending many messages over many connections. This kind of traffic causes a lot of variation in message sizes and timing, which is captured by the top-ranked feature mqttdeltalengthstd, the standard deviation of MQTT message lengths in a flow. High values here reflect the inconsistent message sizes typical of tools like MQTT-malaria designed to stress the broker. The feature bwdmqttlengthmax shows the largest message size sent back by the broker, which can grow when the broker is under heavy load. The features qos and pubackpercentage highlight irregular use of quality of service levels and problems in acknowledging published messages, both of which suggest abnormal behavior. Other features like bwdmsgabsdeltalengthmin, msgabsdeltalengthmean, and fwdmqttlengthstd show inconsistent message timing and content in both directions. These features reflect the noisy and overloaded traffic generated during a malaria-based DoS attack, helping the model differentiate from everyday communication.

These behavioral insights are essential for improving explainability and designing protocol-aware IDS rules and mitigation strategies. Furthermore, such interpretability enhances trust in deep learning-based cybersecurity solutions, supporting their adoption in regulated and safety-critical IoT environments.

5.4 Computational Efficiency

The computational efficiency of the proposed MQTTFlowLyzer-TabNet framework is a critical determinant of its viability for real-time deployment in resource-constrained IoT environments. To evaluate its practical applicability, we quantified the average inference latency, defined as the time required by the TabNet model to classify a single MQTT network flow:

Table 12: Computational analysis of MQTTFlowLyzer feature extraction on the MQTTset dataset

Dataset	Attack Category	Size (KB)	#Packets	#MQTT Packets	#Flows	Flow Duration	Feature Extraction	CSV Write	Analyzer Time
	Malformed	1,015	10,924	3,656	2,925	66.50 s	7.76 s	0.25 s	2.85 s
	MalariaDoS	48,707	130,223	93,317	188,957	6.77 s	803.94 s	19.85 s	4510.46 s
MQTTset	Capture_Flood	8,021	613	303	1	205.76 s	0.44 s	0.00 s	9.53 s
	Bruteforce	1,119	14,501	2,921	1,479	66.23 s	0.81 s	0.16 s	1125.00 s
	Slowite	789	9,202	3,056	1,024	0.00 s	0.28 s	0.07 s	1.49 s

$$\text{Inference Latency} = \frac{\sum_{i=1}^{n} T_{\text{flow}_i}}{n} \tag{16}$$

Here, T_{flow_i} represents the processing time for each flow. The inference latency was empirically measured using the best-performing TabNet configuration obtained via Optuna hyperparameter optimization. The resulting average latency was approximately **0.0725 milliseconds per flow**, achieved in a CPU-only execution environment. This sub-millisecond latency underscores the model's suitability for real-time intrusion detection, satisfying the stringent responsiveness requirements of IoT-based environments.
From a theoretical standpoint, the computational complexity of the TabNet model can be expressed as:

$$O(n_{\text{steps}} \cdot (n_d + n_a) \cdot d) \tag{17}$$

where n_{steps} is the number of decision steps, n_d and n_a represent the dimensions of the decision and attention layers, and d denotes the input feature dimensionality. This linear complexity in both model depth and width ensures scalability to larger datasets while maintaining computational tractability.
To further optimize performance, the framework integrates the following strategies:

- **Bayesian hyperparameter optimization**[1] (refer to Table 4.3), which enables an automated, sample-efficient search for model configurations that jointly optimize accuracy and computational efficiency.
- **Early stopping** with a patience threshold of 20 epochs, which curtails unnecessary training iterations and reduces computational overhead without compromising model performance.

Overall, these design choices contribute to an efficient and scalable framework that supports real-time intrusion detection in practical MQTT-based IoT deployments.

5.5 Feature Dimensionality and Computational Cost Analysis

High-dimensional feature extraction, as employed by *MQTTFlowLyzer*, enhances intrusion detection accuracy by capturing detailed and protocol-specific characteristics of MQTT traffic flows. However, this granularity introduces non-negligible computational overhead, a critical concern in environments with constrained processing, memory, and energy resources.
To evaluate this overhead, we conducted a performance analysis of feature extraction on the *MQTTset* dataset, focusing on the cost of computing 404 flow-level features. Table 12 summarizes the per-category computational time for feature extraction, CSV writing, and per-flow analysis.

[1] Implemented using the Optuna framework [60]

The analysis reveals considerable variance in extraction time, primarily influenced by dataset size and flow volume. For instance, the *malariaDoS* category required over 800 seconds for feature extraction and over 4,500 seconds for flow analysis, while smaller subsets like *slowite* were processed within seconds. These results underscore the necessity for scalable and optimized data preprocessing strategies in large-scale IoT deployments.

Despite the computational cost, rich, flow-level features are justified by their substantial contribution to detection performance. Unlike traditional packet-based approaches, *MQTTFlowLyzer* captures higher-level protocol semantics and temporal behaviors critical for identifying complex and zero-day attacks. When combined with TabNet's sparse attention mechanism and interpretability, the resulting system achieves high detection accuracy and explainability, while remaining computationally feasible for practical deployment.

5.6 Detection Metrics

In our evaluation of MQTTFlowLyzer–TabNet, we employed a comprehensive suite of detection metrics to assess overall and class-specific performance. These metrics were computed in a multiclass classification setting, including accuracy, precision, recall, and F1-score.

We adopted macro-averaging as our aggregation strategy to maintain consistency and comparability with existing research. This approach calculates the unweighted mean of the per-class metrics. It is particularly suitable for intrusion detection in IoT networks, where the severity of an attack does not necessarily correlate with the volume of flows or packets it generates. Macro-averaging ensures that each class contributes equally to the overall performance assessment regardless of its size or frequency.

In line with best practices in the field, the normal (benign) class was treated as one among many labels in the multiclass setting. Therefore, true negatives (TNs) were not computed in this context, as is standard in multiclass classification paradigms.

The detection metrics for multiclass classification with L classes and n samples are defined as follows:

$$\text{Accuracy}_{MC} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\hat{y}_i = y_i) \tag{18}$$

$$\text{Precision}_{MC} = \frac{1}{|L|}\sum_{i=1}^{|L|} \frac{TP_i}{TP_i + FP_i} \tag{19}$$

$$\text{Recall}_{MC} = \frac{1}{|L|}\sum_{i=1}^{|L|} \frac{TP_i}{TP_i + FN_i} \tag{20}$$

$$\text{F1 Score}_{MC} = \frac{1}{|L|}\sum_{i=1}^{|L|} \frac{2TP_i}{2TP_i + FP_i + FN_i} \tag{21}$$

Where TP_i, FP_i, and FN_i are the true positives, false positives, and false negatives for class i, respectively. These definitions provide a rigorous mathematical foundation for evaluating model performance in scenarios involving multiple attack types and varying class distributions.

We also employed early stopping with a patience parameter to prevent overfitting and to ensure optimal

generalization. This evaluation framework ensures statistical robustness and reliability of the reported results, making our findings suitable for reproducibility and cross-study comparisons in the IDS research community.

5.7 Comparative Performance Evaluation

The experimental results demonstrate the effectiveness of the proposed *MQTTFlowLyzer-TabNet* framework in securing MQTT-based IoT environments. Key performance metrics highlight the practical benefits of our approach:

- **Attack Detection Accuracy**: Our model achieved impressive recall rates exceeding 99% for critical attacks such as *malformed* and *brute-force* (Table 10), indicating its robustness in identifying malicious flows with minimal false negatives. Such high recall is vital for IoT systems, where even a few undetected intrusions could lead to significant disruptions or cascading failures.
- **False Positive Mitigation**: Precision scores exceeded 90% for the majority of attack classes (e.g., *malaria*, *slowite*), demonstrating the effectiveness of our feature selection mechanism. However, the observed lower precision (63%) for the *bf_dos/ddos* category highlights the inherent challenges in distinguishing high-frequency attacks from legitimate traffic due to their similar traffic profiles.
- **Balanced Performance**: The macro-average F_1-score of 0.80 highlights our model's ability to maintain a good balance between precision and recall. This balance is primarily facilitated by the hierarchical and interpretative nature of TabNet's feature extraction process.

To enable a precise comparative evaluation, Figures 28 present the confusion matrices of two models: the Random Forest (RF) classifier from the original MQTTset study [34] and the proposed MQTTFlowLyzer–TabNet framework. The MQTTset paper provided only the confusion matrix without reporting per-class metrics such as precision, recall, or F1-score. Consequently, these metrics had to be derived manually, limiting their results' depth of interpretability and reproducibility.

The RF model demonstrates notable misclassifications across several classes. For instance, 5997 *DoS* samples and 212 *BruteForce* samples were incorrectly predicted as *Normal*, which can critically undermine an IDS by allowing malicious traffic to pass undetected. These errors are especially problematic in MQTT-based IoT environments where protocol-specific attacks can exhibit subtle behavior.

In contrast, the proposed MQTTFlowLyzer–TabNet model shows significantly improved classification performance, with clear diagonal dominance in its confusion matrix. Key classes such as *BruteForce*, *Malaria DoS*, and *Malformed* are classified with near-perfect accuracy, and overall misclassification is minimal. This demonstrates the model's ability to separate expected and malicious flows reliably.

This improvement is mainly due to two design elements: (1) the use of flow-level MQTT features engineered by MQTTFlowLyzer, and (2) the attention-based feature selection capability of TabNet, which enhances interpretability and supports more robust generalization.

Moreover, in other datasets such as the DoS-DDoS-MQTT-IoT [55], only overall accuracy was reported across multiple models, with no confusion matrix or class-wise breakdown provided. This lack of detail restricts meaningful evaluation. Similarly, in [39], the authors evaluated uni-flow and bi-flow models on

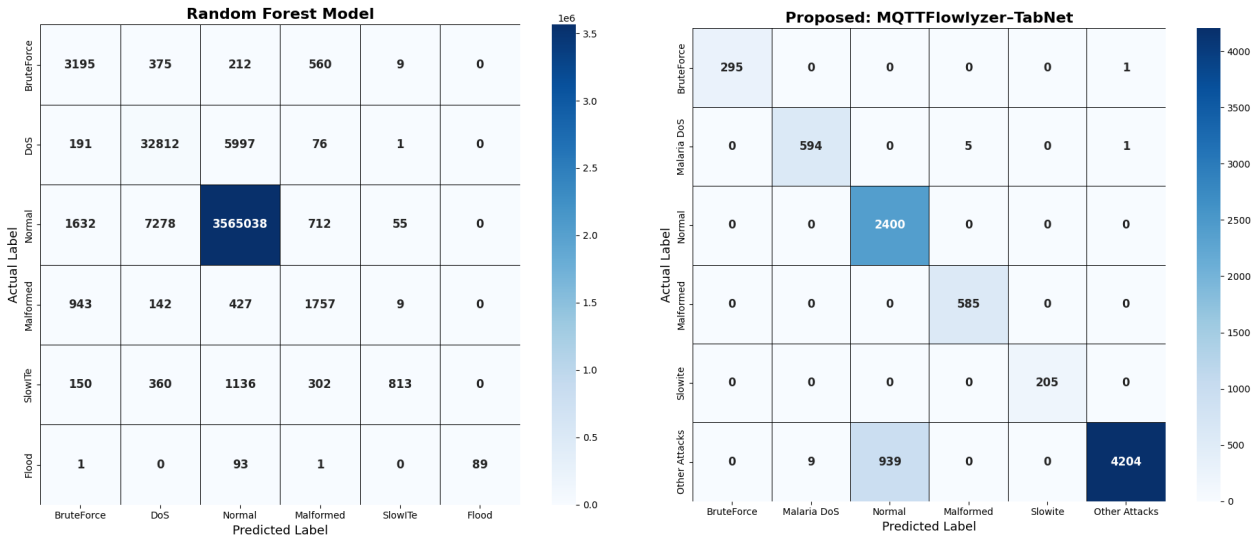

Figure 28: Comparison of Confusion Matrices between (a) RF model (baseline) and (b) proposed MQTTFlow-Lyzer–TabNet.

the MQTT-IoT-IDS2020 dataset and acknowledged that their models failed to detect brute-force attacks effectively in the updated MQTT-IoT-IDS2020-UP dataset. In contrast, our model achieved a 99% detection rate on the same dataset in a zero-day setting, highlighting its robustness against unseen attacks and improved transferability.

In summary, these results prove that MQTTFlowLyzer–TabNet outperforms traditional ML-based IDS models, offering greater accuracy, interpretability, and reliability. Its superior classification performance, especially for challenging attack classes, confirms its potential for real-world deployment in MQTT-driven IoT infrastructures.

6 Conclusion and Future Works

This paper presents a comprehensive and application-layer-centric intrusion detection framework for MQTT-enabled IoT ecosystems. Our end-to-end pipeline spans from raw PCAP acquisition to lightweight, interpretable threat detection, tackling the challenges of feature extraction, model architecture design, uncertainty-based inference, and interpretability. Central to our system is MQTTFlowLyzer, a protocol-aware flow generator that extracts 404 MQTT-specific statistical and behavioral features, forming the basis of the newly introduced BCCC-MQTT-IDS-2025 dataset.

A primary innovation of this work lies in the design of the TabNet-based learning model, designed specifically for flow-level MQTT traffic. This architecture integrates feature selection, classification, and confidence-based zero-day detection into a unified pipeline, addressing existing limitations in interpretability and adaptability. We refined our feature set to 378 dimensions through dynamic preprocessing to improve generalization. This enabled robust detection across a diverse set of attacks, achieving over 99% accuracy on brute-force classification and reliable detection of malformed and flooding-based intrusions.

The zero-day detection module, built upon uncertainty estimation, successfully identified previously unseen MQTT-BF traffic, underscoring the model's resilience under open-world conditions. Meanwhile, our interpretability module produced class-wise feature attributions, offering meaningful explanations that aid forensic traceability and auditability, often absent in traditional IDS frameworks.

Our evaluation methodology surpasses standard accuracy benchmarks by emphasizing generalization capacity, class-specific robustness, and interpretability. The proposed system demonstrated consistent performance across multiple experimental configurations, offering practical value for real-time IoT security deployment. Comparative studies against baseline models and prior literature further affirm the distinctiveness and efficacy of our framework.

Future work will prioritize developing and integrating more comprehensive MQTT-specific datasets that reflect a broader spectrum of realistic and emerging attack types. The current landscape's coverage of subtle protocol-layer behaviors and adversarial scenarios remains limited. Expanding the diversity and scale of available MQTT threat data is critical for enabling more rigorous benchmarking and advancing the robustness of intrusion detection frameworks. Additional research directions include ensemble learning for increased resilience, open-world generalization with unlabeled traffic, and deployment in real-time environments to validate system performance under production-level conditions. Another promising direction involves using class-specific behavioral signatures. These profiles can serve as templates for anomaly detection, session clustering, and automated threat response, enabling lightweight yet adaptive defenses in dynamic IoT settings.

Bibliography

[1] Q. Liu, H. B. Keller, and V. Hagenmeyer, "A bayesian rule learning based intrusion detection system for the mqtt communication protocol," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pp. 1–10, 2021.

[2] H. Ap and K. K, "Secure-mqtt: an efficient fuzzy logic-based approach to detect dos attack in mqtt protocol for internet of things," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 90, 2019.

[3] M. A. B. Ahmadon, N. Yamaguchi, and S. Yamaguchi, "Process-based intrusion detection method for iot system with mqtt protocol," in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, pp. 953–956, IEEE, 2019.

[4] M. A. Khan, M. A. Khan, S. U. Jan, J. Ahmad, S. S. Jamal, A. A. Shah, N. Pitropakis, and W. J. Buchanan, "A deep learning-based intrusion detection system for mqtt enabled iot," *Sensors*, vol. 21, no. 21, p. 7016, 2021.

[5] S. Ullah, J. Ahmad, M. A. Khan, M. S. Alshehri, W. Boulila, A. Koubaa, S. U. Jan, and M. M. I. Ch, "Tnn-ids: Transformer neural network-based intrusion detection system for mqtt-enabled iot networks," *Computer Networks*, vol. 237, p. 110072, 2023.

[6] C. Prajisha and A. Vasudevan, "An efficient intrusion detection system for mqtt-iot using enhanced chaotic salp swarm algorithm and lightgbm," *International Journal of Information Security*, vol. 21, no. 6, pp. 1263–1282, 2022.

[7] I. Idrissi, M. Azizi, and O. Moussaoui, "An unsupervised generative adversarial network based-host intrusion detection system for internet of things devices," *Indones. J. Electr. Eng. Comput. Sci*, vol. 25, no. 2, pp. 1140–1150, 2022.

[8] T. K. Boppana and P. Bagade, "Gan-ae: An unsupervised intrusion detection system for mqtt networks," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105805, 2023.

[9] Z. Gao, J. Cao, W. Wang, H. Zhang, and Z. Xu, "Online-semisupervised neural anomaly detector to identify mqtt-based attacks in real time," *Security and Communication Networks*, vol. 2021, pp. 1–11, 2021.

[10] M. Abdel-Basset, H. Hawash, R. K. Chakrabortty, and M. J. Ryan, "Semi-supervised spatiotemporal deep learning for intrusions detection in iot networks," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12251–12265, 2021.

[11] I. Ullah and Q. H. Mahmoud, "A framework for anomaly detection in iot networks using conditional generative adversarial networks," *IEEE Access*, vol. 9, pp. 165907–165931, 2021.

[12] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138509–138542, 2021.

[13] D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh, "An ensemble multi-view federated learning intrusion detection for iot," *IEEE Access*, vol. 9, pp. 117734–117745, 2021.

[14] I. Ioannou, P. Nagaradjane, P. Angin, P. Balasubramanian, K. J. Kavitha, P. Murugan, and V. Vassiliou, "Gemlids-miot: A green effective machine learning intrusion detection system based on federated learning for medical iot network security hardening," *Computer Communications*, vol. 218, pp. 209–239, 2024.

[15] M. M. Katende, "Combining mqtt and blockchain to improve iot data security," *IoTBDS*, 2020.

[16] P. Akshatha and S. D. Kumar, "Mqtt and blockchain sharding: An approach to user-controlled data access with improved security and efficiency," *Blockchain: Research and Applications*, vol. 4, no. 4, p. 100158, 2023.

[17] E. Al-Masri, K. R. Kalyanam, J. Batts, J. Kim, S. Singh, T. Vo, and C. Yan, "Investigating messaging protocols for the internet of things (iot)," *IEEE Access*, vol. 8, pp. 94880–94911, 2020.

[18] R. K. Kodali and S. Soratkal, "Mqtt based home automation system using esp8266," in *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, pp. 1–5, 2016.

[19] S. V. Mukherji, R. Sinha, S. Basak, and S. P. Kar, "Smart agriculture using internet of things and mqtt protocol," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 14–16, 2019.

[20] A. Cornel-Cristian, T. Gabriel, M. Arhip-Calin, and A. Zamfirescu, "Smart home automation with mqtt," in *2019 54th International Universities Power Engineering Conference (UPEC)*, pp. 1–5, 2019.

[21] R. K. Kodali and B. S. Sarjerao, "A low cost smart irrigation system using mqtt protocol," in *2017 IEEE Region 10 Symposium (TENSYMP)*, pp. 1–5, 2017.

[22] R. Atmoko and D. Yang, "Online monitoring & controlling industrial arm robot using mqtt protocol," in *2018 IEEE International Conference on Robotics, Biomimetics, and Intelligent Computational Systems (Robionetics)*, pp. 12–16, 2018.

[23] B. Safaei, A. Monazzah, M. Bafroei, and A. Ejlali, "Reliability side-effects in internet of things application layer protocols," in *2017 2nd International Conference on System Reliability and Safety (ICSRS)*, pp. 207–212, 2017.

[24] D. Soni and A. Makwana, "A survey on mqtt: A protocol of internet of things (iot)," in *International Conference on Telecommunication, Power Analysis and Computing Techniques (ICTPACT)*, 2017.

[25] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, "Mqtt-iot-ids2020: Mqtt internet of things intrusion detection dataset," *IEEE Dataport*, 2020.

[26] E. C. P. Neto, S. Dadkhah, S. Sadeghi, H. Molyneaux, and A. A. Ghorbani, "A review of machine learning (ml)-based iot security in healthcare: A dataset perspective," *Computer Communications*, vol. 213, pp. 61–77, 2024.

[27] G. Patti, L. Leonardi, G. Testa, and L. L. Bello, "Priomqtt: A prioritized version of the mqtt protocol," *Computer Communications*, vol. 220, pp. 43–51, 2024.

[28] OASIS Standard, "MQTT Version 5.0." `https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html`, 2019. Accessed: 2025-05-06.

[29] A. J. Hintaw, S. Manickam, S. Karuppayah, and M. F. Aboalmaaly, "A brief review on mqtt's security issues within the internet of things (iot).," *J. Commun.*, vol. 14, no. 6, pp. 463–469, 2019.

[30] M. Husnain, K. Hayat, E. Cambiaso, U. U. Fayyaz, M. Mongelli, H. Akram, S. Ghazanfar Abbas, and G. A. Shah, "Preventing mqtt vulnerabilities using iot-enabled intrusion detection system," *Sensors*, vol. 22, no. 2, p. 567, 2022.

[31] Open Information Security Foundation (OISF), "Suricata 4.1.8 released," *Suricata Blog*, 2020. Available: `https://suricata-ids.org/2020/04/28/suricata-4-1-8-released/` (Accessed: 2025-04-23).

[32] G. Potrino, F. De Rango, and A. F. Santamaria, "Modeling and evaluation of a new iot security system for mitigating dos attacks to the mqtt broker," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2019.

[33] A. Alzahrani and T. Aldhyani, "Artificial intelligence algorithms for detecting and classifying mqtt protocol internet of things attacks. electronics 2022, 11, 3837," 2022.

[34] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, "Mqttset, a new dataset for machine learning techniques on mqtt," *Sensors*, vol. 20, no. 22, p. 6578, 2020.

[35] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A. L. Muñoz-Castañeda, I. Garc´ia, C. Benavides, *et al.*, "Multiclass classification procedure for detecting attacks on mqtt-iot protocol," *Complexity*, vol. 2019, 2019.

[36] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine learning based iot intrusion detection system: An mqtt case study (mqtt-iot-ids2020 dataset)," in *International networking conference*, pp. 73–84, Springer, 2020.

[37] E. Ciklabakkal, A. Donmez, M. Erdemir, E. Suren, M. K. Yilmaz, and P. Angin, "Artemis: An intrusion detection system for mqtt attacks in internet of things," in *2019 38th Symposium on Reliable Distributed Systems (SRDS)*, pp. 369–3692, IEEE, 2019.

[38] F. Mosaiyebzadeh, L. G. A. Rodriguez, D. M. Batista, and R. Hirata, "A network intrusion detection system using deep learning against mqtt attacks in iot," in *2021 IEEE Latin-American Conference on Communications (LATINCOM)*, pp. 1–6, IEEE, 2021.

[39] M. Schrötter, A. Niemann, and B. Schnor, "A comparison of neural-network-based intrusion detection against signature-based detection in iot networks," *Information*, vol. 15, no. 3, p. 164, 2024.

[40] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, "A deep learning ensemble approach to detecting unknown network attacks," *Journal of Information Security and Applications*, vol. 67, p. 103196, 2022.

[41] C. N. Modi and K. Acha, "Virtualization layer security challenges and intrusion detection/prevention systems in cloud computing: a comprehensive review," *the Journal of Supercomputing*, vol. 73, no. 3, pp. 1192–1234, 2017.

[42] M. Shafi, A. H. Lashkari, and A. H. Roudsari, "Ntlflowlyzer: Towards generating an intrusion detection dataset and intruders behavior profiling through network and transport layers traffic analysis and pattern extraction," *Computers Security*, vol. 148, p. 104160, 2025.

[43] "CVE-2020-13849: MQTT 3.1.1 Timeout Misconfiguration Vulnerability." `https://nvd.nist.gov/vuln/detail/CVE-2020-13849`, 2020. Accessed: April 23, 2025.

[44] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[45] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

[46] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[47] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[48] S. O. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," *arXiv preprint arXiv:2008.08604*, 2020.

[49] A. Shalaginov, O. Semeniuta, and M. Alazab, "Meml: Resource-aware mqtt-based machine learning for network attacks detection on iot edge devices," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, pp. 123–128, 2019.

[50] N. Moustafa, "New generations of internet of things datasets for cybersecurity applications based machine learning: Ton_iot datasets," in *Proceedings of the eResearch Australasia Conference, Brisbane, Australia*, pp. 21–25, 2019.

[51] J. Aveleira, "MQTT_UAD: MQTT Under Attack Dataset. A Public Dataset for the Detection of Attacks in IoT Networks Using MQTT." https://joseaveleira.es/dataset/, 2023. Accessed on 20 November 2023.

[52] E. Ciklabakkal, A. Donmez, M. Erdemir, E. Suren, M. K. Yilmaz, and P. Angin, "Artemis: An intrusion detection system for mqtt attacks in internet of things," 2019. Accessed: 2024-06-12.

[53] S. Ghazanfar, F. Hussain, A. U. Rehman, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "Iot-flock: An open-source framework for iot traffic generation," in *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, pp. 1–6, IEEE, 2020.

[54] H. Siddharthan, T. Deepa, and P. Chandhar, "Senmqtt-set: An intelligent intrusion detection in iot-mqtt networks using ensemble multi cascade features," *IEEE Access*, vol. 10, pp. 33095–33110, 2022.

[55] A. Alatram, L. F. Sikos, M. Johnstone, P. Szewczyk, and J. J. Kang, "Dos/ddos-mqtt-iot: A dataset for evaluating intrusions in iot networks using the mqtt protocol," *Computer Networks*, vol. 231, p. 109809, 2023.

[56] S. Dadkhah, E. Carlos Pinto Neto, R. Ferreira, R. Chukwuka Molokwu, S. Sadeghi, and A. Ghorbani, "Ciciomt2024: Attack vectors in healthcare devices-a multi-protocol dataset for assessing iomt device security," *Raphael and Chukwuka Molokwu, Reginald and Sadeghi, Somayeh and Ghorbani, Ali, CiCIoMT2024: Attack Vectors in Healthcare Devices-A Multi-Protocol Dataset for Assessing IoMT Device Security*, 2024.

[57] D. Abeles and M. Zioni, "MQTT-PWN, IoT exploitation & recon framework." https://mqtt-pwn.readthedocs.io/en/latest/index.html, 2018. Accessed: 2020-02.

[58] A. Palmieri, P. Prem, S. Ranise, U. Morelli, and T. Ahmad, "MQTTSA: A tool for automatically assisting the secure deployments of MQTT brokers," in *Proceedings of the 2019 IEEE World Congress on Services (SERVICES)*, (Milan, Italy), pp. 47–53, IEEE, 2019.

[59] T. Kakarla, A. Mairaj, and A. Y. Javaid, "A Real-World Password Cracking Demonstration Using Open Source Tools for Instructional Use," in *Proceedings of the 2018 IEEE International Conference on Electro/Information Technology (EIT)*, (Rochester, MI, USA), pp. 0387–0391, IEEE, 2018.

[60] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

Vita

Vita

Candidate's full name: Arefeh Kouhironaghi

Bachelor of Electrical Engineering at University of Amirkabir, 2017-2022

Publications:
Arefeh KouhiRonaghi, Arash Habibi Lashkari, "MQTTFlowlyzer: An MQTT Application-Layer Traffic Analyzer for Interpretable Flow-Level Intrusion Detection and Zero-Day Threat Identification in IoT Environment Using TabNet", Computer Networks, 2025 (under review, Q1, IF=5.6) (Elsevier).