Master Thesis

Enhancing the Security of AI Systems In IoT Environment: Analysis and Defending against Poisioning Attacks

Submitted by:	Jenny Nadar
First examiner:	Markus Miettinen
Second examiner:	Peter Ebinger
Date of start:	09.12.2024
Date of submission:	13.05.2025

Statement

I confirm that I have written this thesis on my own. No other sources were used except those referenced. Content which is taken literally or analogously from published or unpublished sources is identified as such. The drawings or figures of this work have been created by myself or are provided with an appropriate reference. This work has not been submitted in the same or similar form or to any other examination board.

12.05.2025 ...

Date, Signature of the Student

Abstract

In the progressively interconnected world, IoT devices are integrated deeply in our day to day lives; yet, their exensive utilization also presents considerable security challenges. As Artificial Intelligence(AI) systems become integral to safeguarding IoT networks, it is imperative to ensure their resistance against hostile threats. This thesis examines the susceptibility of AI-driven traffic classifiers in IoT settings to poisoning attacks, wherein compromised inputs or parameters are employed to dimish model efficacy.

Upon assessing other datasets, we determined that the IoT-23 dataset is the best approriate owing to its extensive collection of authentic malware traffic recorded in pcap files. They were transformed into relevant flow-based and packet-based features for training of machine learning models. In this work proposes an XGBoost- based pipeline for traffic classification and an RNN-based deep learning model to behave like a transformer. Several attacking scenarios were performed to test the robustness of these models, e.g: mislabeling the training data, perturbating feature weights and changing the hyperparameters of the model.

These experimental results show that small-scale attacks can dramatically drop the classification accuracy and integrity of a global model in a federated learning environment. The results highlight the necessity for effective defence and early detection mechanisms in AI-enabled IoT systems to mitigate such stealthy adversarial approaches.

As a part of defence strategy, different defence mechanisms like early anomaly detection mechanisms, robust model aggregation techniques, and hyperparameter validation layers can help to mitigate such attacks. The results of the attack experiment emphasizes the need for including security checks in the training loop of AI models deployed which will help to bridge the gap between performance and resilience.

Contents

Table of Figures

List of Tables

1 Introduction

The Internet of Things (IoT) has quickly become an essential part of a modern digital infrastructure that connects billions of viable smart devices—from consumer appliances and wearable computing to industrial sensors and autonomous vehicles. Together, while the cognitive elements that make up this ecosystem allow for great convenience and automation of everyday life, they also create an exponential new attack surface for adversaries to leverage cyber threats. IoT devices are commonly in resource constrained, under-resourced and physically insecure environments, allowing adversaries the potential to deploy their own tools and exploit IoT devices for disruption of service, espionage, or hijacking legitimate behaviors of such devices.

To combat and become resilient to ongoing malicious actions and cyber threats, AI—through machine learning-based intrusion detection systems (IDS)—has emerged as a potentially powerful layer of defense. These models have shown promise in both learning complex patterns in the flow of network traffic are capable of identifying when affirmative behaviors and activities are not congruent in the real time. In this regard, this thesis focuses on online traffic classification and its role in assessment of malignant actors on an edge network. Regardless of the level of technology-based trust, AI-based intrusion detection models are themselves not immune from attacks.

One specific stealthy and dangerous threat is poisoning attacks. In a poisoning scenario, attackers introduce misleading training data to explicitly impact the training process of an AI-based or machine learning model or may manipulate model parameters or change the weights. In federated learning environments, where models are trained locally on edge devices and the updated weights are only shared, such attacks can effectively poison the global model by impacting one or a many clients. This poses a critical question: as we lean into more AI that is securing IoT networks, how do we ensure that the AI is secure too?

In this thesis, we explore the risks of poisoning AI-based traffic classification models in IoT environments. We first consider a range of publically available datasets, finally deciding that the IoT-23 dataset represents our best option for understanding the impacts of poisoning since it contains real-world malware covering the labeled pcap traffic. After converting the pcap files into flow-based features, we trained traffic classifiers using XGBoost and a RNN model optimized for sequential traffic behavior. Finally, we created several poisoning attack scenarios based on data mislabeling, feature scaling adjustments, and change in learning parameters to evaluate the effects of the attack for potential degradation of the model performance.

Our research further investigates how attackers can exploit the federated learning aggregation process, how that influence can be transmitted across the system, and how even small changes can result in a rippling impact on global accuracy. In addition to the simulated attack, we outline potential early-stage defensive techniques like anomaly-aware model aggregation, hyperparameter monitoring, and statistical consistency checks, in order to recognize incipient trends and influences that could be poisoning before it is passed into the system.

In the end, this thesis is intended not only to show the risks present, but also to help create more robust AI models for the community—models that are able to learn, not only from the data, but from the threats hidden in the data

2 Related Work

The deployment of Artificial Intelligence (AI) within Internet of Things (IoT) ecosystems has revolutionized the ability to detect and prevent security threats. In consideration of the growingly sophisticated method of cyber-attacks and the different variations of IoT deployments, it is important to reflect on the growth of security frameworks based upon AI in these distributed environments. In this chapter, we will review the existing literature on IoT security, the use of Federated Learning (FL) as a learning framework that preserves privacy, the emerging risks of poisoning attacks within decentralized learning systems, and the range of defences. Each section is designed to progressively build the foundation upon which the subsequent methodology and experiments are based.

2.1 IoT Security Challenges and AI Integration

The Internet of Things (IoT) domain has become an important enabler for smart environments, in areas such as healthcare, transportation, energy, and home automation, among others, and while IoT systems are a heterogeneous and resource-constrained environment, they also present unique security issues. Increasingly, those devices involving natural and artificial infrastructures are expanding the total threat surface for attack. Artificial Intelligence (AI) and Machine Learning (ML) have been flagged as appropriate solutions to improve security response through automation and can help organizations identify problems without human intervention. However, as previously mentioned, AI and ML are commonly constructed around a centralized architecture which introduces new restrictions in applying a more decentralized architecture such as Federated Learning (FL). The remainder of this chapter reviews IoT vulnerabilities, the support of AI and ML, and the other constraints from moving to privacy-preserving decentralized intelligence.

2.1.1 Inherent IoT Vulnerabilities.

IoT systems are fundamentally insecure due to the intersecting characteristics of their architecture and operation.

- **Diversity of Devices:** IoT ecosystems are composed of various types of devices - temperature sensors, smart cameras, GPS trackers, and industrial controllers in which each comes bundled with its own firmware, OS types, and communication protocols. This diversity complicates the implementation of standard security mechanisms and creates inconsistencies that can be exploited by motivated attackers [1] .
- **Limited Resources**: IoT devices are resource-constrained (limited memory, processing power, and energy supply) which inhibits traditional security mechanisms like cryptographic protocols and runtime monitoring from executing properly. Attackers regularly attempt to exploit these resource limits in DDoS attacks such as examples like the Mirai botnet [2]
- **Data Privacy Dangers**: Data is usually stored in large cloud exploitative systems in a related fashion for processing in traditional IoT systems. As such, sensitive data is stored together in a centralized notation, either behavioural, biometric, or actual location data. A breach or attack at the central point may represent a massive vulnerability for thousands of devices simultaneously, making them appealing targets for adversaries [3].

2.1.2 Role of AI/ML in IoT Security.

AI and ML methods are being applied more often in order to strengthen security in IoT applications via intelligent decision-making, anomaly detection, and pre-emptive action against security threats.

- **Anomaly Detection:** Supervised, unsupervised, and semi-supervised ML models are trained to recognize the normal operating states in network traffic or sensor data, along with behaviours that differ from expected or acceptable patterns. Therefore, these models can highlight new or zero-day attacks, and various abnormalities in communication behaviours that static rule-based models won't recognize. Kuzlu et al. [4], for example, suggested an AI-based system for detecting botnets in smart home networks based on real-time traffic analysis.

  The figure 1 gives a simple illustration of how anomaly detection works in an IDS. The image illustrates how incoming data is filtered, monitored, and compared in the deployed detection system against learned

behavioural baselines, to determine if someone is trying to commit some criminal act in the environment. The IDS will generate alerts when the data shows signs that deviance is present, suggestive of possible malicious intent and will typically do this without prior explicit knowledge of any attack signatures.

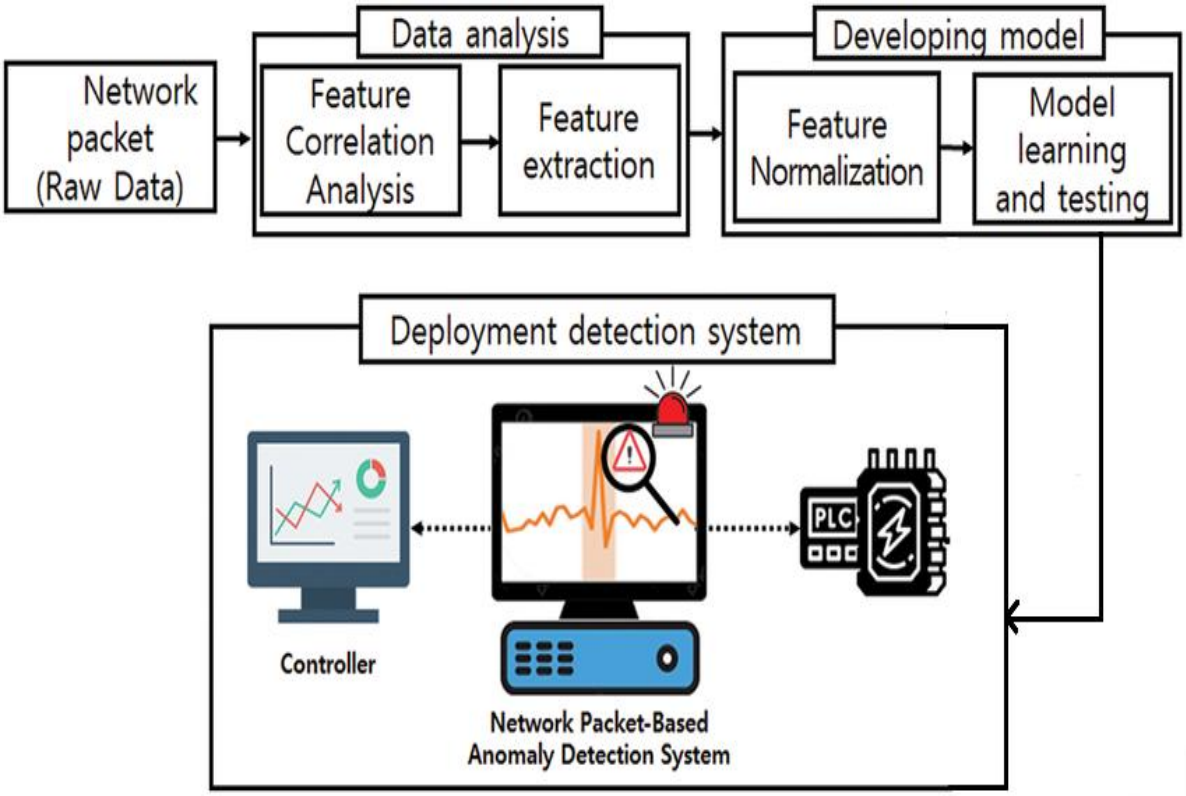

Figure 1: Working of IDS System (Adapted from [5])

- **Traffic Classification**: Deep learning models and the use of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks are used to classify traffic flows based on an analysis of packet headers and the sequencing of payloads. Traffic classification models are trained to model the distinctions between malicious and benign traffic with high accuracy, however, found platforms like SE-MAR's uses support vector machines (SVM) for real-time monitoring in smart environments that lack adversarial robustness against model manipulation or poisoning attacks [6].

  The below figure (refer figure 2) shows the general framework of Traffic classification in IoT Setting. It illustrates how traffic classification models differ from IDS detection by ingesting packet-level input like flow classification, extraction of statistical and time-based features, and classification to classify traffic using Artificial Intelligence (AI) algorithms. It captures the end-to-end pipeline of data acquisition to the classification of potential threats, to illustrate the layered decision-making process associated with that class of systems.

Figure 2: General working of a traffic classification in IoT Environment.

- **Predictive Maintenance**: AI-based optimization algorithms, and particularly RL, have been suggested for addressing problems of predicting device failure or patterned anomalies introduced by security breaches. They aim to bolster system reliability by propagating authorized maintenance or other proposed actions against compromised devices prior to a comprehensive attack. [7]

Together, these different forms of AI/ML, specificity anomaly detection and traffic classification, provide the core of intelligent IoT security systems. Figures 1 and 3 provide a visual demonstration of the real-world architecture and working elements of IDS and traffic categorization systems, respectively. AI/ML has allowed IoT networks to move the structures and processes of reactive defence systems towards intelligent security infrastructures to provide real-time decision-making analytics, from adaptive learning, that can identify anomalies and classify patterns of traffic.

2.1.3 Limitation of Centralized AI Models

Although centralized AI models have advantages, they encounter some systemic issues when being deployed into IoT environments.

- **Latency & Real-Time Constraints**: Centralized processing creates latency for both threat detection/mitigation as it takes time for the data to transmit remotely to a centralized server while also waiting for the model inferences to return back to the edge. This latency may be important for some scenarios that need real-time responses as seen in autonomous vehicles or industrial control systems. [8]
- **Scalability Limitation**: As the number of connected objects exponentially increases into the billions, training and managing the centralized models are no longer computationally feasible in terms of connected data. As the volume of data sent from edge devices to centralize clouds also begins to flow into gigabytes, it begins to create bandwidth congestion and throughput issues with inference pipelines [9]

- **Privacy and Compliance Risk**: Centralized architectures are by design counterintuitive with emerging privacy legislation such as the General Data Protection Regulation (GDPR). Centralizing user data in a database storage increases concerns about data exposure and increases inheritance of non-compliance with the privacy requirements [10].

2.1.4 Solution to Centralized AI Model Limitation: Decentralized AI

To mitigate the limitations associated with centralized models, researchers have designed decentralized learning methods like Federated Learning (FL). FL aims to jointly train AI models across multiple decentralized devices while not directly sharing the original data. It only shares model updates or gradients which reduces the overall data exchanged and mitigates privacy issues. [11]

This decentralized approach has merits that align with the privacy and scalability of IoT systems. However, it also leads to new security vulnerabilities. Among them, model poisoning attacks (where adversaries inject a malicious update) are a significant objection. Adversaries can harm the operational integrity of a model without ever explicitly accessing the raw data. In fields associated with our study such as IoT, with limited trust and oversight, and the challenges associated with IoT design and deployment, such attacks become more insidious and challenging. Our study discusses the challenges associated with FL systems that have IoT components. Sections four and five detail and break down the various attack vectors most prominently the threat model, the attack vectors, and finally the defence mechanisms or frameworks that may be useful in navigating a IoT-integrated FL system.

2.2 Federated Learning in IoT: Opportunities and Vulnerabilities.

Federated learning (FL) reflects a technical evolution of the Internet of Things (IoT) by enabling collaborative training of AI models on encounters of smart edge devices without requiring sensitive data to be consolidated. At the same time, this new paradigm provides distinct advantages such as privacy, less bandwidth needed and scalability — attributes that are particularly desirable for the distributed and data-sensitive features of IoT contexts.

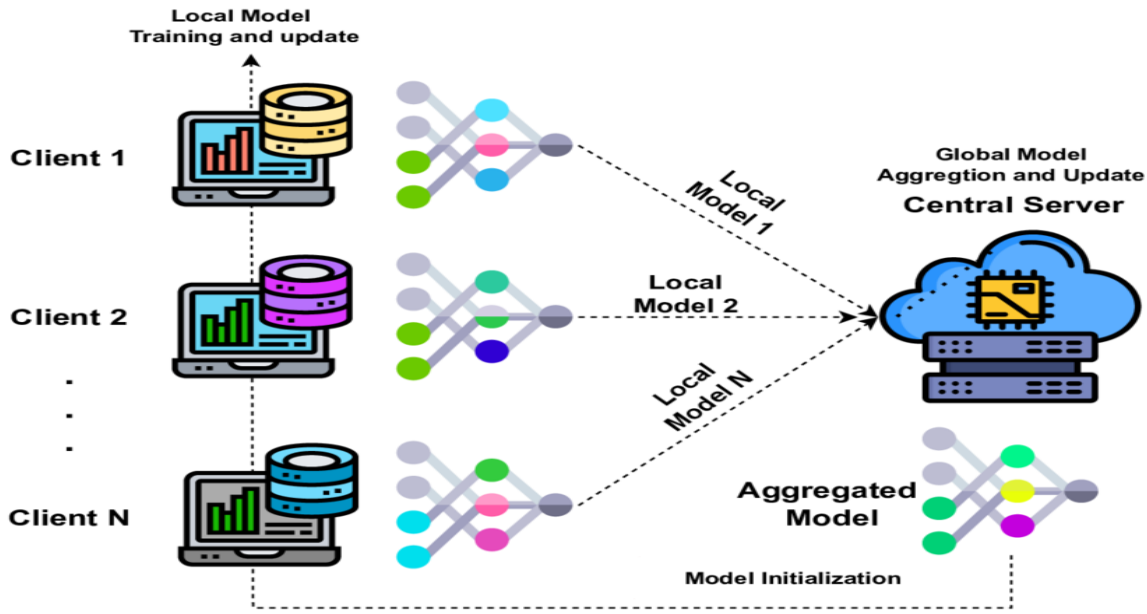

Figure 3: Federated Learning Architecture in IoT Systems.

The above figure 3. shows a standard federated learning (FL) architecture for IoT systems is comprised of multiple client which are heterogeneous IoT devices (e.g., smart home sensors, wearables, or industrial controllers) locally training machine learning models on device-specific data. Each device sends their model updates (e.g., gradients or weights) periodically to a central server where the updates are aggregated to produce a global model using techniques such as Federated Averaging (FedAvg). The central server sends back the global model to each device so that the devices can continue training each model. This iterative process develops the global model collaboratively on all clients while keeping raw data decentralized.

Nonetheless, this distributed scheme creates additional attack surface area and operational complexity in resource constrained, adversarial, and heterogeneous IoT environments. Devices may differ in hardware capabilities, data distribution, or reliable connectivity - which increases the risk of model drift, biased learning, or malicious contributions. Additionally, the central server may not have access to local training conditions, which adversaries can use to sneakily poison the IoT and/or inject malicious updates that could weaken the data and affect the global model negatively.

The possibilities and vulnerabilities of FL in IoT pave the way for the next discussion on attack vectors - specifically poisoning attacks, and how they exploit the absence of centralized authority and the distributed trust that FL is founded upon.

2.2.1 Benefits of FL in IoT

Federated Learning (FL) is a naturally aligned concept for use in Internet of Things (IoT) settings. The collaborative element of FL has significant market appeal for IoT devices in decentralized scenarios because it enables model training across edge devices while sustainable preserving privacy, conserving bandwidth, and scalability. This addresses many of the limitations of central data collection from traditional machine learning in resource constrained and changing IoT networks.

In Figure 4, the scheme of FL, which allows for a middle ground between centrally trained and fully distributed, relies on raw data remaining local to the devices while only sending model updates (such as weights or gradients) back to the centralized aggregator. Thus, there are three main advantages of FL for IOT:

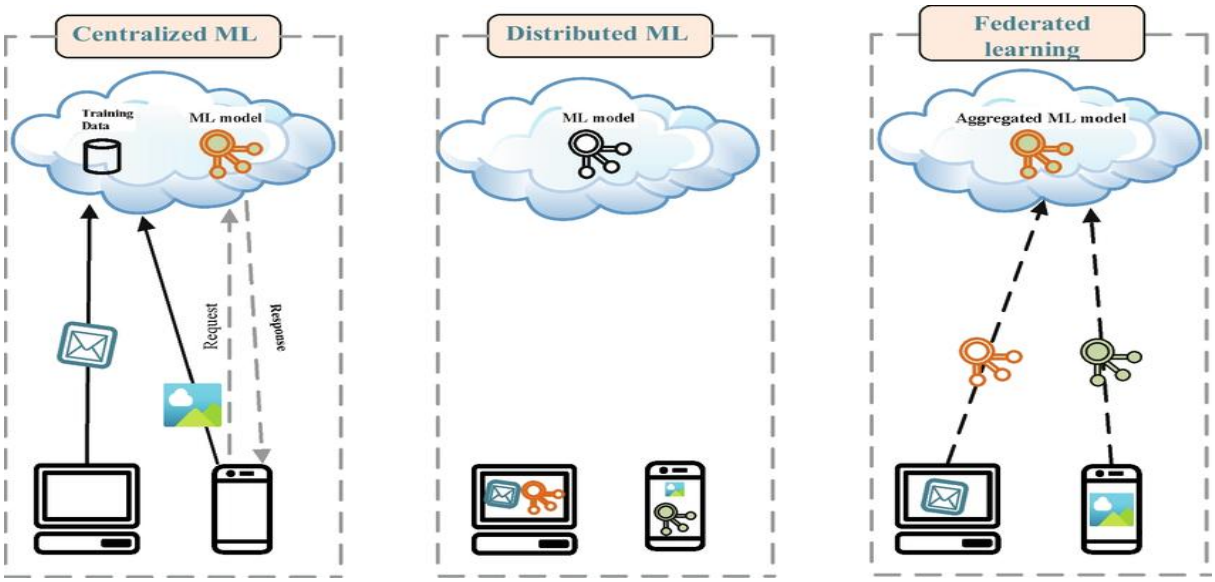

Figure 4: Comparative View of Centralized, Distributed and Federated Learning Training in IoT. [12].

- **Privacy Preservation:** In a centralized training scheme, data is collected and stored on a centralized server. FL stores sensitive data (e.g., health data, location traces, or logs of sensors) in the local data, reducing risk of data exposure significantly. As an example, smart homes could use FL to collaboratively train intrusion detection systems (IDS) using local traffic patterns observed on local gateways, however the packet-level data is stored on the local device and never passed through the IDS system / cloud. [13]. This reduces the safety threat of data loss, or misuse.
- **Bandwidth Efficiency:** In IoT environments where there may be limitations of shared network resources, FL can reduce the amount of communication overhead because only model updates are communicated rather than larger and more plentiful raw data. This makes it especially desirable for low-power wireless networks that have limited uplink capacity. Try to imagine in your mind if FL was NOT used, the communication costs vs. using FL could decrease communication costs on average by 90% over conventional centralized learning where data is sent to a cloud or data centre.
- **Scalability**: FL can be detached from direct limits because it allows devices to participate asynchronously and via heterogeneous devices in any application. A FL system uniquely scales to massive systems, such as industrial IoT systems with thousands of sensors. A key aggregation method i.e., FedAvg method also manages the communications cost of FL at convergence, thereby preserving user privacy and managing environmental impact by having fewer monitored devices.

2.2.2 FL's Vulnerability in IoT Settings.

While FL offers advantages, it also comes with several attack surface areas and contextual limitations - particularly in dynamic, resource-constrained IoT settings:

- **Unreliable Clients**: IoT devices often function in intermittent-connectivity situations using legacy firmware, causing stale updates or noisy updates. Example: A compromised smart thermostat could add instructor gradients to the FL process, resulting in incorrect global temperature optimization.
- **Lack of Centralized Authority**: As raw data remains on local nodes, the FL central server cannot perceive the distribution and quality of training data, leaving it open to invisibly execute poisoning attacks such as mislabelling and data corruption.
- **Data Heterogeneity (e.g., Non-IID Conditions)**: IoT devices generate data with skewed or personalized distributions. For instance, sensor patterns in a factory differ drastically across machines and usage contexts. Non-IID data can affect model convergence stability and performance.

2.2.3 Existing Studies on FL for IoT Tasks

The current literature reveals applicability of FL and undiscussed issues regarding the applicability of FL to IoT security efforts:

- **Intrusion Detection:** In the paper "Poisoning Attacks on FL-based IoT IDS, Nguyen et al." demonstrated backdoor attacks that allowed FL-trained intrusion detection systems (IDS) to misclassify as 30% of malware rate as benign with only 10% compromised clients [13].
- **Traffic Classification:** Two pre-trained model examples, NetMamba **[Wang et al.]** and ET-BERT **[Lin et al.]**, have been modified for FL, however, they were unable to be deployed on any low-power IoT edge device due to computational costs [11].
- **Environmental Monitoring:** The SEMAR platform is another system that uses FL for air quality prediction but assumes clients can be trusted and does not contemplate adversarial clients [6].

Table 1. show the traditional as well as modern approaches along with their pros and cons in the real-world scenarios.

Table 1: Real World Use Cases of FL in IoT Environment.

Use Case	FL Implementation	Strengths	Limitations
Intrusion Detection	FL-Based IDS trained across edge nodes	Reduces need to transmit raw data; privacy - preserving	Vulnerable to poisoning attacks. Eg, backdoors with minimal compromised clients
Traffic Classification	Pre- trained models like NetMamba and ET-Bert	High accuracy on encrypted/raw flows; useful for pattern recognition.	Not deployable on low- power IoT devices due to high computational costs
Environmental Monitoring	SEMAR platform using FL for air quality prediction	Collaborative, decentralized learning from sensor nodes.	Assumes trustworthy clients, does not handle adversarial behaviour or model manipulation.
Device Health Monitoring	Predictive maintenance with distributed FL agents	Early detection of faults; avoids single point failure	Still in early-stage deployment; requires reliable temporal data streams.
User Behaviour Analytics	FL models tracking usage patterns	Personalized security without compromising user data privacy	Difficult to handle non-IID data across users; personalization may affect global convergence.

Although the decentralized and obscured characteristics of FL in IoT promote data sovereignty the possibility for poisoning attacks is enticing. Since attackers can tamper with model updates and not be detected and cases involving devices that are sporadically connected and observe non-IID data feature unique challenges, the following section presents the taxonomy of these attacks, showing how adversaries exploit the weaknesses in FL in order to poison global models and disrupt IoT services.

The real-world applications in Table 1 demonstrate the promise federated learning holds in IoT systems in terms of privacy preservation, decentralized training, and early anomaly detection. However, they also highlight some major limitations-the models ET-BERT, and NetMamba work well in scenarios where computational resource and training time are not an issue but do not work well enough on the edge; most systems assume non-adversarial clients and frankly have no resources for dealing with adversarial behaviour; and quite simply, FL schemes are rather prone to data and model poisonings-and especially so in cases non-IID situations hold sway.

Herein lie the challenges, underscoring the major void-a FL-based IoT is never secure-by construction. Hence, the study at hand targets analyzing the extent of poisoning attacks in affecting AI-based IoT systems, along with finding ways to defend against these attacks that are both highly effective and pragmatically deployable in real life.

2.3 Taxonomy of Poisoning Attacks in Decentralized Systems.

Poisoning attacks take advantage of the collaborative nature of decentralized learning systems, such as Federated Learning (FL), by attacking the integrity of the model training workflow. The attacks differ primarily in their attack surface either data poisoning, which is an attack on the input training data, or model poisoning, which is an attack on the model update that clients share during aggregation. Poisoning attacks also differ with respect to the number of compromised clients—from small stealthy cases of manipulation to larger episodic systemic failures.

2.3.1 Attack Types

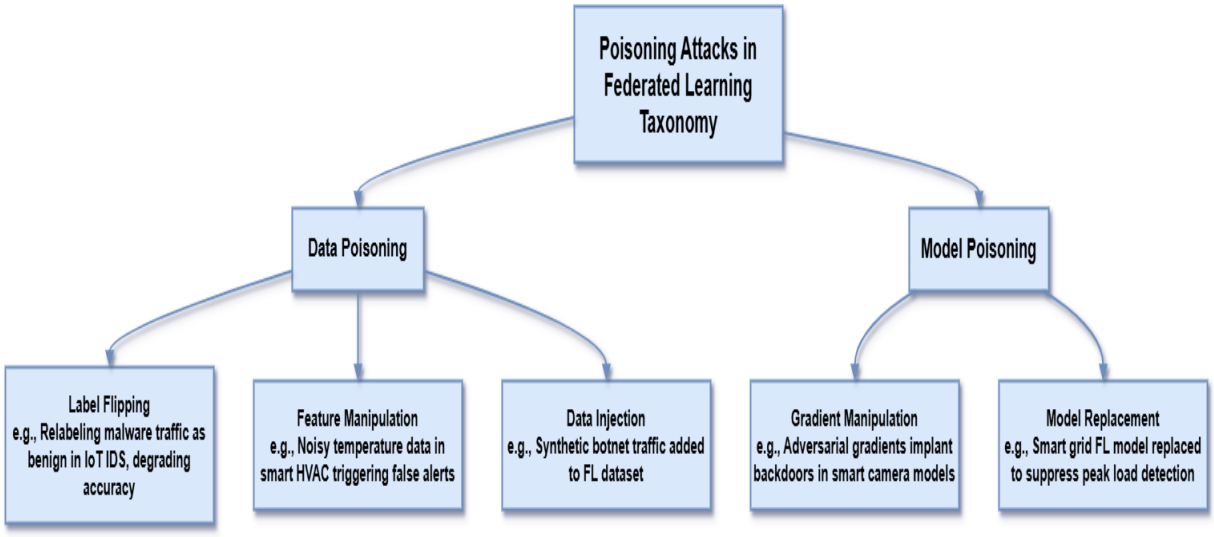

Figure 5: Types of Poisoning Attacks

Data Poisoning Attack

Data poisoning attacks entail manipulation of local training dataset prior to the model training process so that it can influence the generalization of the globally shared model.

- **Label Flipping**: In this attack, the adversary manipulates the labels of the training samples—meaning that malicious network traffic for example is labelled benign. Example: In IoT intrusion detection systems, flipping labels for attack packets led to a drop in model accuracy between 25–40% [14].
- **Feature Manipulation**: In these attacks, attackers introduce non-obvious noise or outlier patterns into the sensor input to influence or bias the learned features. Example: Vuseghesa et al. [14] showed that when they manipulated new temperature data in smart HVAC systems, it led to false alarms and wasted energy.
- **Data Injection**: In these attacks, synthetic malicious data is injected into the local dataset, which is often designed to evade heuristics. Example: Attackers generate fictitious but realistic network flows in the effort to confuse a traffic classifier.

Model Poisoning Attack

These attacks do not change any data, rather they manipulate (gradients or parameters) local model updates sent to the central server.

- **Gradient Manipulation**: Adversarial clients upload poisoned gradients that were maliciously designed to shift the global model's decision boundary. For example, Nguyen et al. demonstrated poisoned gradients enabled backdoor activations in IDS models trained in FL [13].
- **Model Replacement**: Instead of only uploading gradients, attackers upload a new malicious model that is designed to overwrite the global state during aggregation.

2.3.2 Case Study in IoT/FL

1. **Backdoor Attack:**
   As part of backdoor attacks, attackers insert specific "triggers" (such as byte sequences or packet header anomalies) into the attackers training data. When the model sees these during inference, it can cause malicious output. For example, in a smart grid system using FL, triggers that were embedded into the training data caused a compromised model to produce incorrect load predictions. This could lead to operational modification and subsequently risk blackout events [15]

2. **Targeted Misclassification:**
   Targeted misclassification attacks cause an attack in only specific samples or classes while maintaining overall model accuracy. For example, Dunn et al. demonstrated that in a label-flipping attack of an IoT traffic classifier that was using Gradient Boosting Machines, their model's precision went from 81% to 46% when only 20% of model training clients were compromised [16].

2.3.3 Severity: Single-Client vs. Multi-Client Compromises

Attacks can take place by compromising one or more clients or end devices in an IoT Environment.

1. Single Client Compromises:
   A lone compromised device submits poisoned data or model updates. These attacks are stealthy and hard to detect.

2. Multiple Client Compromises:
   Multiple colluding clients inject similar poisoned content, drastically altering global model convergence and boosting attack efficacy.

While poisoning attacks take advantage of the decentralized and privacy-preserving nature of FL, but they can only succeed if there are no reliable and scalable defences. This issue is particularly pressing in IoT, where low-capacity edge devices cannot afford to spend processor cycles on training, detection or retraining. The following section reviews defences that are designed to detect, defend against, or evade these attacks that are lightweight enough for real-world, and operational contexts.

2.4 Defence Strategies against Poisoning Attacks

Poisoning attacks utilize the collaborative nature of Federated Learning (FL), in which untrusted federated participants can manipulate the model updates in a way that destroys global performance. In this section, we categorize and review current defence approaches, distinguishing between proactive and reactive approaches. Additionally, we note the limitations of adapting these approaches to resource-constrained, decentralized environments such as IoT.

2.4.1 Proactive Defences.

Proactive defences are those that aim to identify or mitigate malicious behavior to model updates before aggregation. The majority of the proactive approaches make stronger assumptions or require clearer coordination among clients.

Robust Aggregation Techniques:

These techniques apply specific modifications to the aggregation function to reduce the overall influence of poisoned updates:

- Trimmed Mean: Removes a potentially predefined quantity of the highest and lowest model updates from the aggregation function to remove outliers
- Krum: A method that aims to select the update that is most representative of the bulk of the models updates by minimizing the cumulative distance must its nearest adjacent models updates. Example: Dunn et al. [16] proposed a trimmed mean-based FL model at the aggregation that allowed for more robustness to label flipping in IoT intrusion detection.

Anomaly Detection:

Anomaly detection methods apply statistical or unsupervised methods to detect abnormal updates:

- Z-scores, Mahalanobis distance, and time-series deviation techniques identify updates that diverge largely from others
- Clustering-based filters can be developed to isolate model updates that lay beyond dominant patterns. For example, Baracaldo et al. used provenance metadata to detect poisoned IoT data via update clustering. However, the method relied on a blockchain-backed framework, which is impractical for constrained devices. [17]

Federated Trust and Reputation Systems:

Trust scores denote devices' past behavior or energy profile and influence model aggregation weights. For example, devices returning inconsistent reporting pattern in smart grid networks would see their weight when aggregated in FL (federated learning) down weighted [15].

2.4.2 Reactive defences

Reactive approaches attempt to **detect and mitigate attacks post-facto**, often after poisoned updates have affected the model.

Gradient Inspection:

These methods inspect client gradients with the goal of finding divergence or suspicious client gradient patterns.

This inspection has ability to discover adversarial models by assessing update orientation across a number of clients. This generally requires a baseline "clean" gradient to compare against. For Example: Chiba et al. proposed comparing gradients to discover a poisoned update while looking for deviations from expected learning routes. [18]

Differential Privacy (DP):

DP reduces the amount of influence a client can impose by adding random noise to the model updates, thus limiting the breadth of a successful poisoning experience.

In the survey conducted by Ramirez et al. shows strong theoretical guarantees which creates emphasis on accuracy and privacy producing trade-off; this is especially important for some performance orientated IoT systems [19].

Trigger Pattern Detection for Backdoors:

Specific mechanisms looking for the backdoor triggers (i.e. pixel patterns, packet signatures) that the malicious client created like using activation clustering or latent feature tracking methods to isolate outlier input-trigger behaviours.

For example, in smart lock systems, models can be monitored for consistent misclassification triggered by unusual Bluetooth signal bursts.

2.4.3 Shortcomings of Defences in IoT Scenarios

The theoretical perspectives on most current defences failed to translate into an effective implementation, and in practice, there are significant gaps when utilized in a large-scale IoT ecosystem.

Computational Overhead:

Defences like Krum or advanced anomaly detection methods require substantial calculations and computational (e.g., many pairwise distance calculations). This is an issue for even a single IoT node, as these algorithms place significantly more overhead on IoT devices with limited processing resources and memory.

Dependence in Trusted Data:

A number of anomaly techniques are facilitated by having a source of "clean" data to rely on as a point of reference. This is often not the case in a persistent decentralized, and ever-changing IoT network bubble where the notion of a "clean" reference is mostly irrelevant.

Inability in Handling Collusion:

Many sophisticated attacks are carried out by groups of clients that are each compromised (e.g., multiple clients close together colluding) - thus, they subtly ignore their individual updates accounts while maliciously adjusting their updates. For Example: Zhu et al., illustrated that edge devices in a smart city could carry back doors while working together, imitating legitimate patterns, completely bypassing any clustering-based defence [20].

2.4.4 Comparative Overview of Defence Strategies.

In order to make the comparison of defence mechanisms clearer, an overview of key proactive and reactive strategies discussed in this section is provided in Table 2. This table summarizes each strategy detailing its main strengths and weaknesses, especially in terms of its utilization in conjunction with IoT environments. Notably, some strategies such as Trimmed Mean are lightweight defences and robust when it comes to resource-constrained feasibility (say, in relation to devices operated in an IoT environment), while more resource demanding schemes such as Krum, or, provenance-based clustering, consider the computational or infrastructural expense when applied in distributed IoT settings. In addition, although strategies such as Differential Privacy have great potential, they require considerable calibration that should be thought of in terms of utility loss; a critical point in relation to implementing models in real-time for IoT applications, such as intrusion detection or smart grid control.

Table 2: Comparison of Defence Strategies.

Defence Method	Category	Strength	Limitation	IoT Feasibility
Trimmed Mean	Proactive	Reduces outlier impact	May discard valuable info	Lightweight & practical
Krum	Proactive	High robustness against single client attacks	High computational cost	Not scalable on edge devices
Z-Score Detection	Proactive	Simple statistical flagging	Needs reference distribution	Limited use in real world IoT

Provenance Clustering	Proactive	Groups and isolate suspicious updates	Requires metadata and storage (eg. Blockchain)	Heavy resources dependency.
Gradient Inspection	Reactive	Detects divergence effectively	May need multiple clean baselines	May lag in real-time systems
Differential Privacy	Reactive	Obfuscates sensitive updates	Accuracy degradation	Needs tuning for balance

While many current defences were theoretically featured, we acknowledge that most of them offer relatively limited mitigations. Even if they represent some form of threat remediation, in an IoT ecosystem, and recognizing energy, latency, and coordination, many will quickly become infeasible. The focus of the next section is evaluating these defences from the perspective of metrics specific to IoT to assess practicality as they apply to realistic attack settings.

On the other hand, most of the existing defences still do not consider the main constraints within IoT systems: low computation, low communication, and the dynamic participation of clients. Though a lightweight approach like the Trimmed Mean might be used, this approach may be throwing away some useful updates. While robust methods like Krum or provenance-based clustering will never scale with such overheads. Moreover, the reactive ones, like gradient inspection and differential privacy, need clean baselines or lose utility because they need to be used in real time, so they are not a good fit for many IoT situations. More importantly, the few that exist offer no reliable protection from subtle poisoning attacks that never produce strong statistical deviations. Our defence mechanism, therefore, so as to put the two complementary mechanisms of outlier filtering and cosine similarity-based inspection to work at detecting suspicious updates at a minimum cost. Going on the principle of feasibility and robustness, the method intends to shore up the gap that exists between theoretical defences and real-world deployments in federated IoT environments.

2.5 Evaluation Metrics and Benchmarking in IoT/FL

Poisoning attacks pose risks to Federated Learning (FL) model reliability, particularly in heterogeneous and resource-limited Internet of Things (IoT) environments. To formally evaluate the impacts of poisoning attacks and the effectiveness of corresponding defence measures relies on the use of evaluation metrics, along with datasets as baseline benchmarks for the evaluation process. Despite recent advances in evaluating FL's vulnerability to poisoning attacks [21], neither clearly defined metrics, nor baseline datasets allow researchers to properly evaluate their results.

In current work, there is a lack of IoT-specific consideration, such as the limitations of the devices and the fact that IoT networked environments are dynamic. This section provides a critique on current evaluative metrics and the data selection processes, as well as outstanding challenges for cross-domain benchmarking.

2.5.1 Common Evaluation Metrics

An effective evaluation of FL-based poisoning attacks requires a multi-metric suite examining both impacts on model performance and effectiveness on different levels of effects on overall systems. The following metric types are among the most widely used in the existing literature:

- **Accuracy Reduction:** Gauges the reduction in model performance as a function of poisoning. An example would be a 20% reduction in the classification accuracy of a model after an attack indicating a significant degradation in model trust [16].
- **False Positive Rate (FPR):** Can be important depending on the type of module targeted, especially when considering intrusion detection systems (IDS); as high FPRs mean that legitimate traffic may potentially be flagged as malicious; leading to a high potential of alert fatigue (among other threats) [22].
- **Detection Rate (True Positive Rate):** Measures the percentage of poisoned updates or malicious clients (in the case of ongoing updates) that were identified by the defence.
- **Convergence Time:** Extended or unstable convergence patterns during federated training usually indicate an adverse effect of adversarial updates.
- **Resource Overhead:** Quantifies the energy consumption, memory overhead, and latency incurred by defensive strategies—compelling aspects for use in low-energy edge devices.

For Example: In the technical paper of by Dunn et al. They found that, under label-flipping attacks, Gradient Boosting classifiers exhibited an accuracy drop of 15 - 40% on the ToN-IoT dataset, illustrating that more conventional ML algorithms are susceptible to increased adversarial vulnerabilities in FL environments [16].

Table 3: Summary of Evaluation Metric in FL/ IoT.

Metric	Relevance in IoT or FL	Practical Limitations
Accuracy Drop	Captures overall model degradation	Does not reveal root cause (for example, attack or noise)
False Positive Rate	Critical for IDS performance	Sensitive to class imbalance
Detection Rate	Gauges effectiveness of defences	Hard to benchmark without ground truth
Convergence Time	Indicates training instability	Varies across architectures / datasets
Resource Overhead	Reflects deployment feasibility	Often ignored in academic benchmarks

2.5.2 IoT-Specific Datasets for Poisoning Research

A viable evaluation framework should also use datasets that mirror the distinct traffic patterns, attack vectors, and resource constraints of IoT ecosystems. Although not all are equally useful in the study of poisoning attacks, the following datasets have previously been used:

- **IoT-23 Dataset:**
  The IoT-23 dataset was released in 2020 by the Stratosphere Laboratory (Czech Technical University) to provide a reliable and recent dataset for developing and researching machine learning-based intrusion detection systems (IDS) for IoT environments. Unlike previous datasets, IoT-23 was developed solely on malicious traffic generated by infected IoT devices, representing a wide range of malware campaigns, including botnet attacks across a wider spectrum of malware campaigns than present in prior datasets.

  The dataset consists of 23 labelled capture scenarios (pcap files), containing different network behaviours involving benign activity or malicious behaviours such as botnet communication and malware propagation behaviours such as command-and-control (C&C) behavior. The scenarios also involve well-known malware families including Mirai, Tsunami, Gafgyt, and Okiru, and the devices used for infection originated from embedded Linux-based generic IoT and Raspberry Pi systems to ensure that the traffic looked "real."
  Each scenario file contained:
  1) A labelled description file that indicates benign or malicious flows.
  2) Flow summaries provided by CICFlowMeter.
  3) A description file of the infection scenario, malware type, and device behavior.

  IoT-23 can support both binary classification (benign vs malicious) as well as multi-class classification (based on malware family or behavior type). The dataset can be useful for developing traffic-based anomaly detection systems, botnet detection tools as well as network forensics models [23].

  Its realism heterogeneity and flow specific data granularity makes this dataset suitable for simulating data and model poisoning types of scenarios. This dataset does not contain encrypted payload, so it is not feasible to parse this dataset to encrypted specific model like ET-Bert.

- **ToN-IoT dataset:**
  The ToN-IoT (Telemetry Operating system and Network data for IoT) dataset was designed by the Cyber Range Lab at the University of New South Wales (UNSW) Canberra to address the growing need for realistic heterogeneous datasets to evaluate AI-based solutions to cybersecurity problems, in the context of IoT. To achieve that goal, the ToN-IoT dataset was designed to follow real-world smart city architectures, consisting of a three-tier structure with Edge, Fog, and Cloud layers. The dataset allows researchers to explore complex interactions spanning various levels of system hierarchy. Researchers can also develop, test, and benchmark machine learning models in situations that resemble either real IoT deployments or deployments involving simulated IoT devices.
  ToN-IoT is unique in having four heterogeneous data sources:
  1) **Network Traffic Data** - collected through full packet capture with netsniff-ng and enriched with a network security monitor Zeek (formerly Bro). The Network Traffic data set is provided as raw .pcap files as well as CSV files with features yielding flow-level information such as source/destination IPs and ports, protocol type, sizes for packets, timestamps, HTTP headers and DNS queries.
  2) **Telemetry Data from IoT Devices** - collected from simulated physical and virtual IoT devices (e.g., smart fridge, smart thermostat, GPS tracker, smart Modbus devices, motion-enabled lights, garage door, weather sensors. These devices were orchestrated with Node-RED and Mosquitto MQTT Broker, with telemetry logs generated with real-time exportation as structured CSV files.
  3) **Operating System Logs: (Windows)** – Collected from Windows 7 and Windows 10, using Windows Performance Monitor; Log data includes –
     a. CPU usage,
     b. memory usage,
     c. processes running, and
     d. d) disk I/O statistics.
  4) **Operating System Logs: (Linux) –** Captured by using the atop monitoring tool from Ubuntu servers that responded as orchestrator/nodes and middleware nodes.

  This dataset encompasses both normal operation patterns and nine types of attacks; Scanning, Denial of Service (DoS), Distributed DoS (DDoS), Backdoor, Ransomware, Injection Attacks, Cross-Site Scripting (XSS), Password Cracking, and Man-in-the-Middle (MITM). Each of the attacks were generated using sources and methods offensive tools ie: Metasploit, sqlmap, Hydra, XSSer, Nmap. The socio-technical

attack events recorded are labelled by ground-truth mapping that produces IP addresses and timestamps relevant to each scenario.

In its processed state, the ToN-IoT dataset comprises 22 million and counting records within its modes. Each record has over 44 features that includes two primary labels: label as a binary class identifier (0=Normal, 1=Attack) type as a multiclass identifier that specifies the type of attack

ToN-IoT has use cases such as intrusion detection, anomaly detection, attack classification, and federated learning. The dataset is comprehensive in types of data and contains context specific to IoT, which makes it ideal for testing AI models in a distributed adversarial context in smart environments [24].

- **UNSW-NB15 dataset:**
The UNSW-NB15 dataset was developed in 2015 by the Australian Centre for Cyber Security at the University of New South Wales (UNSW) Canberra. This dataset is intended to be a modern alternative to the KDD99 and NSL-KDD dataset that is considered outdated, and thus the UNSW-NB15 dataset is contemporary. This dataset addresses the most realistic network traffic the captures both normal operation and a large variety of sophisticated synthetic cyber-attack scenarios. The dataset was generated using the IXIA PerfectStorm tool that purposefully simulated interactions of the internal network and external internet network in a lab-controlled environment.

  The resultant traffic contains 2,540,044 records, distributed in 47 features from packet and flow-level metadata. Feature extraction has been accomplished using different tools, like Argus, and Zeek, and features collected here include packet size, duration, protocol type, TCP flags, and connection state. Each record comes labelled for the sake of supervised machine learning. The dataset also experienced a wide variety of nine different attack categories, as follows: Fuzzers Analysis Backdoor DoS Exploits Generic Reconnaissance Shellcode Worms

  The dataset comes in the form of four CSV files: two for training and two for testing, each with a binary label of attack or normal (label = 0 or 1) and a specified attack type for multiclass classification problems. UNSW-NB15, although limited to representing only network traffic, is frequently used in the security research domain because it provides realistic attack traffic, the various classes are evenly distributed, and includes a useful and documented structure. Its utility is primarily as a dataset to train conventional intrusion detection systems, evaluate classifiers in a traffic-based approach, or to report baseline performances in machine learning-based experiments.

  Even though UNSW-NB15 is not intended for IoT-specific application, it is still an important dataset in the field of cybersecurity due to its extensive structure, robust support of modern attacks, and the ease of use to be part of machine learning-based pipelines. [25] [26].

Table 4: Summary of Different IoT datasets.

Dataset	Real-World Traffic	Label Quality	Device Diversity	Support for Flow-based attacks	Comments
IoT-23	✅ High	✅ Detailed labels (malware specific)	✅ Diverse IoT device types	✅ Strong (packet and flow level)	Ideal for realistic simulations includes granular attacks / benign behaviours.
ToN-IoT	⚠️ Synthetic Workloads	✅ Rich Labelling	✅ Smart home, industrial devices.	⚠️ Limited flow level support	Useful for general IDS: limited realism for poisoning evaluation
UNSW-NB15	❌ Non-IoT Specific	✅ Annotated	❌ No IoT Devices	⚠️ Partial	General purpose IDS dataset; not suitable for IoT-centric research.

2.5.3 Gaps in Cross-Domain and Collaborative Evaluation

While there is significant work quantifying poisoning attacks and defences in FL-based IoT systems, there are clear gaps in the state-of-the-art in terms of benchmarking. These gaps arguably limit our understanding of how resilient the defences studied in science are in an authentic heterogeneous and adversarial environment.

Transferability of Attacks Across Domains: Most evaluations are limited to a single dataset or specific IoT context while also examining the effectiveness of the attack against static smart home devices or campus type set-ups. However, poisoning strategies successful in one domain such as consumer devices, may experience dramatically different behavior in other domains such as industrial control or healthcare IoT, where not only the distributions, but more importantly the behavior of devices may be drastically different. Thus, simply evaluating attacks and defences across domains cannot provide evidence of generalizability when there has been no effort to evaluate the critical transferability of the actual attacks and defences.

Dynamic Attack Scenarios and Temporal Adaptation: Most research has also solely focused on static poisoning strategies, injected when initiating the model, or modified at random fixed intervals. However, real-world adversaries may be able to modify their behavior to adapt, by delaying, increasing, or modifying their rate of poisoned updates.

Overlooked Resource-Aware Metrics: Examining metrics around a resource such as energy consumption, latency or memory footprint is often overlooked in evaluations that are critical for deploying on edge devices. Any defence mechanisms that introduce large computational overheads—such as secure aggregation or multi-round auditing—may completely negate FL's viability for battery restricted IoT nodes. Evaluations based on edge resources must match the resource constrained environments to make them meaningful.

2.5.4 Evaluation with Collusion.

An especially important, and still underexplored, kind of threat to FL-based IoT systems is malicious client collusion. Rather than defence mechanisms simply ignoring individuals' update without analysis from the publisher, a set of adversarial clients may carry out coordinated poisoning under specific training rounds.

This collusion provides a means to manoeuvre the global models during training, while circumventing the traditional mechanisms guarding against them. This collusion is particularly credible in IoT due to the variety of ways that edge devices are constrained, whether it is a shared firmware/central vendor/accompanied vendor/supply chain compromised gateway. Thus, colluding adversarial clients can set off coordinated behaviours believing their actions are disguised by other IoT behaviours.

Challenges to Detection and Benchmarks:

Similarly, while Krum, Trimmed Mean and Multi-Krum are not specific to colluding clients, they rely on finding the statistical distance from benign updates. Colluding clients can design updates that behave overall with benign statistics, essentially getting out of the same conspiracy.

Collusion complicit clients will ensure that the clients are consistent with updates with respect to gradient behaviour. Moreover, evaluations that are proposed for the challenges of colluding clients rarely discover simulations which resemble anything close to their intended coordinated behaviours, leading to overestimation of the robustness of the defences.

Need for Group-Level Evaluation Metrics:

In evaluating robustness against collusion, researchers should consider the following:

- **Inter-client update correlation**: High similarity across unrelated clients, whether geographically or logically, may indicate collusion.
- **Consistency of poisoning behavior** across rounds.
- **Cluster stability metrics**: Metrics noting ongoing groups that influence the global model abnormally.

Experimental Implications:

In order to make sure resilience is accurately evaluated, just as (quasi-) benchmarks that simulate multi-client poisoning using a variety of collusion topologies (e.g. fixed pairs, triad rotations, full-group collusion) will be

important in evaluating resilience, this becomes more urgent with technology implementing IoT or industrial IoT, where multiple compromised clients can appear through shared infrastructure.

To summarize, although important work has been done benchmarking various poisoning attacks across IoT and FL systems, current approaches to evaluation often overlook important considerations such as robustness across domains, the possibility for the adaptivity of the attacking behavior, and the fact that clients may collude. These limitations can create a false sense of security, and reduce the generalizability of defences proposed

With these limitations in mind, the next chapter outlines the experimental design and setup which includes realistic attacks based on IoT-23 data. Including independent and colluding poisoning strategies, alongside resource-aware evaluation metrics, we design a method that has the potential for rigorously evaluating the performance degradation of FL-based traffic classifiers against adversarial interference.

3 Adversarial Model

This chapter delineates the adversarial landscape surrounding Federated Learning (FL) systems being operated within an Internet of Things (IoT) context. It builds upon the theoretical concepts defined earlier to formalize realistic attacker motives, operational abilities, and system flaws. Moving above board, we convert high-level threats to specific attack profiles, and detection measurements, and pave the way for an empirical experiment. Additionally, this section maps the defence methods previously introduced to targeted attack behaviours to provide a practical level of coverage for FL systems expected to work in bounded and heterogeneous IoT environments.

3.1 Threat Model

In federated systems composed of distributed IoT devices that jointly train their own FL models without sharing their raw data, attackers can leverage the inherent pound for pound inefficiencies of decentralized systems to intervene when a device is in a vulnerable state and work to impact the outcome of the FL model. The threat model is defined in terms of attackers, goals, abilities, knowledge level, and key assumptions made in the FL pipeline.

Attacker's Goal:

- Traffic Misclassification: Malicious actors which hold access to the FL models and/or data aim to muddle the original FL model update such that suspected malicious network traffic (for example malware, botnet activity etc.) is misclassified as benign traffic thereby avoiding further detection systems.
- Model Degradation: Adversarial threats want to steer or halt the FL training process from occurring during the submission of poisoned update models, slowly degenerating the model accuracy over recurrent models being trained over time.
- Backdoor Injection: Adversaries embed latent triggers (e.g., an exact sequence of packet metadata) in the training data and during inference encounters that they do not expect to affect global performance metrics to enact targeted misclassification.

Capabilities of Attack:

- Single-Client (Local) Attacks: The attacker takes over a single FL participant. This may be a compromised smart thermostat or sensor node, and the attacker uses the local participant's FL training updates to apply data or gradient-based local perturbations or poisoning.
- Multiple-Client (Collaborative) Attacks: Attackers gain control of several compromised IoT devices. This allows them to coordinate and hive-attack variations of local poisoning that would amplify the attack and almost guarantee a greater chance of avoidance against anomaly detection.

Knowledge of the Attacker:

- White-Box: The attacker is able to see all internal aspects of the FL model, including all model architecture, all parameters of the optimizer, and aggregation logic. This level of knowledge allows them to 'tune' the attacks in a precise way like the triggering of a backdoor or a gradient shaping.
- Black-Box: The attacker does not see the central aggregation process, nor the internals of the model, however, they can poison the local data or gradients using indirect means of poisoning local training data.

Assumptions:

- The central federated learning aggregator (server, edge-gateway, etc.) has no access to the FL clients' training data, it relies only on model updates to aggregate.
- IoT clients possess diverse levels of resource availability (computing power, bandwidth, memory), which may affect the feasibility of an attack or the deployment of a defence.

3.2 Attack Scenarios

Built on the previously explained threat model, the following section defines two primary poisoning attack scenarios commonly observed in Federated IoT networks: single-client compromise and multi-client collusion. Each scenario describes the attack vector, expected model performance consequences, and a real-world comparison via an example use-case.

3.2.1 Client Compromise

Attack Vector:

- **Label Flipping:** The attacker reverses labels in its local training data (e.g., relabel "malicious" flows as "benign") causing the learned decision boundary to be misaligned.
- **Gradient Manipulation:** simply create malicious gradient updates to introduce noise or push models in adversarial directions by exploiting simple aggregation algorithms like FedAvg (federated averaging).

Impact:

- Model accuracy will degrade 15–25% in non-robust FL schemes [14].
- Increased false negatives for intrusion detection systems, where up to 30% of malware flows may bypass detection [13].

Example Use-Case:

A compromised smart thermostat will provide training data with a biased temperature gradient during federated training, which means the model trained globally will normalize - and fail to detect - any abnormal heating behavior, effectively masking early warning signs of an industrial system's failure [14].

3.2.2 Multi-Client Collusion

Attack Vector:

- **Sybil Attacks:**
  Attackers use multiple compromised clients to disproportionately affect the global model during global model aggregation (FedAvg).
- **Adaptive Poisoning:**
  Many compromised clients can coordinate together to alter the attack characteristics or pattern dynamically, motivated by feedback from the global model, with the goal of making detection of these attacks increasingly difficult.

Impact:

- Under Adaptive poisioning conditions, the accuracy of global models drops anywhere from 40 to 60% especially when multiple compromised client work together for an attack. The impact is higher as compared to a single client compromising.
- The success rates of backdoor attacks exceed upto 80% when trigger patterns are dispersed across compromised clients. [15]

For Example:
In a smart city federated learning deployment, each roadside camera was compromised with attacks injecting adversarial updates that turned heavy congestion into smooth traffic landmarks throwing routing algorithm to malfunction.

3.3 Detection Parameters

In order to defend against the above threat vectors, this section puts forth detection parameters looking for statistical anomalies in model updates or behavioural changes in IoT device operations.

3.3.1 Statistical Anomalies in Client Updates:

Table 5:Statiscal Anomalies

Parameter	Indicator	Suspicious Threshold	Targeted Threats
Gradient Divergence	Cosine similarity with global gradient	< 0.7	Gradient Manipulation.
Update Frequency	Submission rate deviation	> 10 × average submission rate	Sybil attacks.
Loss Discrepancy	Local vs Global training loss	< 0.1 in high – loss rounds	Label flipping.

Gradient Divergence: Clients' gradients from the global direction diverging too far, can show poisoning. A cosine similarity of less than 0.7 is typically a detection threshold [27].

Update Frequency: A suspiciously abnormal frequency of client updates may indicate that there are a number of Sybil nodes trying to control model updates [28].

Loss divergence: Clients whose training loss is significantly lower than the global average are likely manipulating the training data, especially in the early rounds of training [29].

3.3.2 Behavioural Anomalies of IoT devices.

Table 6: Behavioural Anamolies of IoT devices.

Parameter	Behavioural Indicator	Suspicious Threshold	Targeted Threats
Traffic Spikes	Packet transmission volume	>1,000 packets/min	DDoS/backdoor
Endpoint Interaction	Communication with untrusted IPs	Any external IP	Command/control link
Resource Usage	CPU/memory usage	>90% idle usage	Hidden computation

- **Traffic Abnormalities:** If a compromised sensor is flooding the network with traffic, this would indicate a high likelihood of a botnet exploit [17].
- **Endpoint Communication:** If a device is communicating with unregistered IPs, it is likely an indication of unauthorized command links [30].
- **Resource Consumption:** A device that is experiencing high CPU or memory utilization whilst not in use might be performing malicious local computations [21].

3.4 Defences Strategies

This section contextualizes the previously described proactive / reactive defences within the framework of the adversarial attack conceptualizations. Rather than describing these defences again, we will connect them back to their strategic accounts for countering specific adversarial actions.

3.4.1 Proactive Defences (Aligned to Attack Scenarios)

- **Robust aggregates** (e.g., Trimmed Mean, Krum) will reduce both single-client and multi-client poisoning (reducing influence of outlier in model updates). Best when robust aggregations are against attacks that rely on gradient manipulation and label flipping. [16]
- **Federated Anomaly Detection:** by clustering the client gradients (e.g., k-means) we can make sense of outlier behavior collectively without using the raw data. Primarily useful to counter adaptive poisoning and attacks based of Sybil client behavior. [17]

3.4.2 Reactive Defences (Tuned via Detection Parameters)

- **Dynamic Thresholding:** will help tune detection thresholds against anomalies based on different phases of model convergence making it easier to detect anomalous behavior in important parts of the training. [30]
- **Differential Privacy** ($\sigma = 0.5$): added near Gaussian instead of plain Gaussian noise to updates by obscuring information patterns relied upon by adversarial poisoning attacks but effectiveness may cost accuracy [13].

3.4.3 Trade Offs / Limitations

- **Privacy vs. Detection**: Using (DP) caused a reduction of success rates for attacks of around ~35% however it also altered false positive detection rates by 10%. [16]
- **Scalability**: The anonymity clustering steps atop of filtering out statistical anomalies takes a to ~20% extra time in high-volume FL networks.

After developing a thorough understanding of attacker dispositions, detection mechanisms, and appropriate mitigations, the subsequent chapter will transition the advisory model into a set of experimental studies. Using the IoT-23 dataset as the testing goods, we will also construct both single-client and multi-client poisoning attacks and assess mitigations against that are constrained in resources.

This chapter investigated real-world IoT Federated Learning applications along with the pros and cons of existing defense strategies against poisoning attacks. There have been several proposed methods, both proactive and reactive, which, however, mostly fail to find a balance between robustness and strict resource constraints posed by IoT environments. Furthermore, the complex or more subtle kinds of poisonings evade their statistical thresholds. Hence there has arisen an utmost need to have a more efficient and rather adaptive defense in place. Then, in the latter chapter, we treat adversarial threats on the defensive end, presenting interminable details on how poisoning attacks are created and categorized while also showing how the weaknesses of FL are taken advantage of, reinforcing the need to justify our more focused defense method.

4 Design Motivation and Exploratory Directions.

Federated Learning (FL) has presented an attractive solution for protecting data privacy within distributed IoT systems, but with the emergence of new vulnerabilities—a decentralized FL approach presents many opportunities for poisoning attacks. This thesis details how poisoning attacks could be realized in resource-constrained IoT environments and provides solutions to defend against them. For a thorough and reasonable answer, this chapter has documented the intended motivations, challenges, and design considerations that were made in order to shape the final experimental methodology. This chapter has also linked the initial problem statement to the speculative paths which we tested and refined throughout the research.

4.1 Problem Landscape and Motivation

IoT systems have been increasingly infused with intelligent decision-making support for applications like intrusion detection, smart energy, or geolocation services. Still, with the inherent distribution of IoT devices, and their constraints, they provide unique opportunities for data and model poisoning in an FL community. Part of the motivation for this thesis was to examine whether adversarial injections of training data and model update would compromise an IoT system operating in an FL manner, and to create viable countermeasures that account for the practical constraints and diversity of the devices.

Although there have been significant developments in centralized machine learning security, this literature review revealed a significant lack in actual defence strategies specifically applicable to IoT based FL settings, particularly in adversarial settings with label flipping, gradient manipulation or model replacement attacks. Most of the defence strategies studied were either too resource heavy, dependency on a large clean dataset, or did not scale well to multiple compromised clients performing coordinated attacks. Therefore, this study is framed around 2 central research questions:

1) RQ1: What are the differences in the effect of different poisoning strategies on model fidelity, in terms of performance, in federated IoT environments?
2) RQ2: What are the lightweight and scalable defence strategies that can be used to register and possibly mitigate adversarial attacks without losing model performance or privacy?

4.2 General Objectives

The general objective of this thesis is to examine and implement a secure and lightweight framework for detecting and mitigating poisoning attacks in FL-enabled IoT systems. There were the following objectives set out in order to perform this:

- Classification and simulation of various types of poisoning attacks in federated settings (e.g., label-flipping, weights manipulation, data injection).
- The design and implementation of a traffic classification and a malware detection model that merges both deep learning (RNN-based) classifiers and machine learning (XGBoost Model) using deployment at flow-level network features from real-world IoT traffic.
- Performance evaluation between deep learning (RNN Model) and traditional machine learning models (XGBoost Model).
- To investigate and evaluate pre-trained transformer-based security models (ex. ET-BERT, NetMamba) and compare how practical they are to deploy in a constrained IoT environment.
- To propose and evaluate defence mechanisms that are effective against poisoning but are viable for an edge monitoring solution.

4.3 Work Environment and System Overview

The experimental framework is based on a series of Python-based data processing pipelines, comprising both data processing and analysis, and the use of deep learning and machine learning libraries (ex. PyTorch, Scapy, Pandas, NumPy). The system was implemented on a Windows laptop with an Intel i3 processor and 8GB RAM which, dictated method selection, particularly with due regard to the implementation of heavy-weight models (ET-BERT and NetMamba).

The design of the system gave rise to three fundamental components:

Figure 6. shows the components of the systems and all the steps involved in each component.

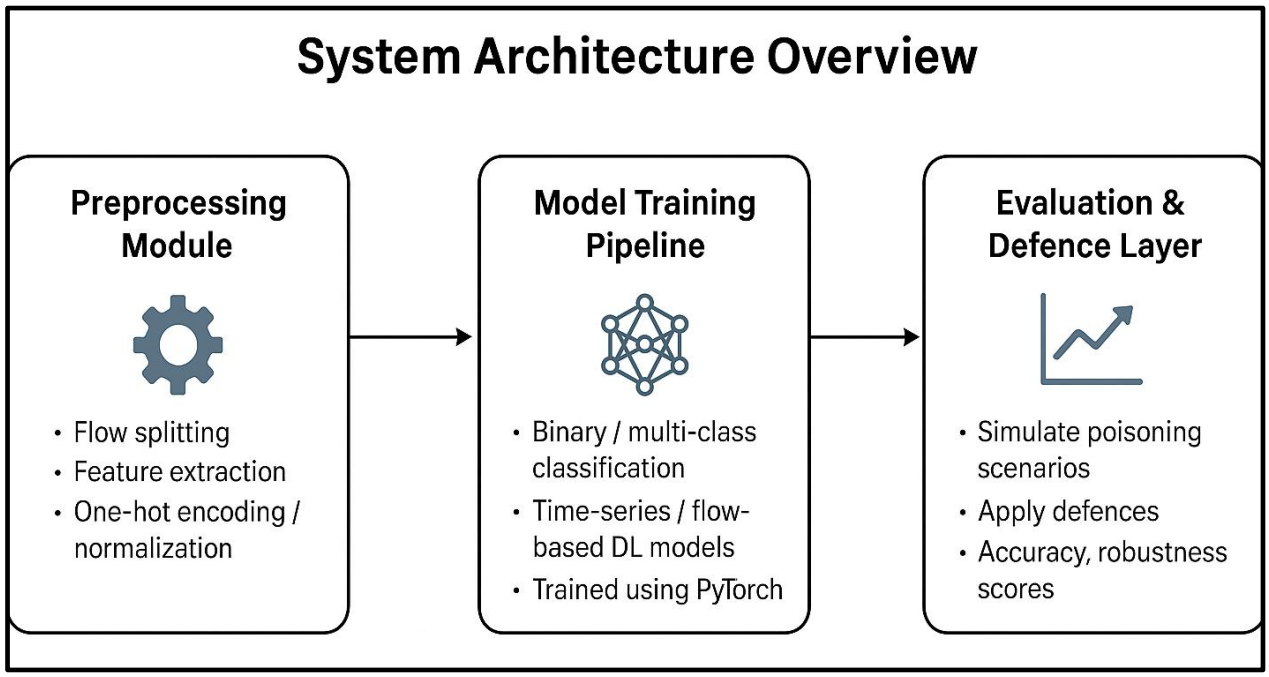

Figure 6: Fundamental Components of Experimental Setup.

a) **Preprocessing Module**: The preprocessing module is responsible for encoding .pcap files from the IoT-23 dataset and pulling out flow-based features such as packet size, inter-arrival times, type of protocol, and TCP flags.
b) **Model Training Pipeline**: This implemented a DL/ML model based binary and multi-classifier that is trained on time-ordered flow features to differentiate between benign and malicious traffic.
c) **Evaluation and Defence Layer**: Here, we considered the robustness of the model against various poisoning scenarios, and we introduced various mitigations including class weighting and outlier removal.

4.4 Exploratory Efforts and Abandoned Directions.

Prior to finalizing adopted architecture, multiple exploratory directions were pursued to identify best support to our problem area and create reasonable parameters for the constraints around the work. The following sections describe this activity where we have attempted to document these stakeholders and articulate why we discarded their efforts in the final implementation.

4.4.1 Dataset Exploration

Multiple datasets as mentioned in the previous chapter under subsection "1.1.1 IoT-Specific Datasets for Poisoning Research". These datasets were closed studies and selected to experiment our proposed approaches of implementing attacking scenarios and then building a defence system to mitigate these attacks. Two different datasets, UNSW-NB15 and ToN-IoT, were investigated first to train and evaluate poisoning scenarios.

- **UNSW-NB15 Dataset:** This study did not consider the UNSW-NB15 dataset because of various limitations that made it unsuitable to use when training robust Federated learning (FL) models, especially by IoT security standards.

  First, aside from the dataset having approximately 2.5 million records, malicious samples found in the data were further low; thus, it becomes even harder to efficiently train a binary classification model to determine benign traffic from attack traffic. The imbalance of benign (0) versus malicious (1) data limited the federated learning model from effectively generalizing in real attack scenarios, particularly in a distributed style, where data heterogeneity is one key challenge.

  Secondly, the dataset was artificially generated in a narrow laboratory-controlled environment, which lacks all the variations and unpredictable behaviours obtainable in the real network traffic. More explicitly, UNSW-NB15 does not address any kind of IoT-communication pattern, nor are device-level identifiers such as MAC addresses available that would go a long way in attributing traffic to specific IoT endpoints, as they are critical in attributing most traffic to specific endpoints in the IoT. The lack of this contextual metadata renders the data of little usefulness in studies that use endpoint as unit of analysis behavior study or learning traffic flow in IoT.

  Thirdly, and most significantly, a spectral view of the dataset in itself (i.e., no traces from edge devices or contextual traffic patterns) made it impossible to train a federated learning model that would learn and differentiate the underlying distributions of benign and malicious flows across decentralized clients. The restriction is likely to impact directly on the performance of the model, which could not converge effectively even to the basic binary classification tasks, pointing towards the fact that the data did not have sufficient complexity and granularity for real-world adversarial learning experiments.

  In this experiment, 10 local clients were trained locally with different machine learning models like CNN, XGBoost + CNN (ensemble variable), RNN etc. Each client trained with the same type of machine learning model created a global aggregation model. So similarly multiple global aggregation model for different machine learning models were create and then the global model updates were then sent to the local models. Table 7 shows the model performance and how it was unable to detect malicious data.

Table 7: Classification Report of UNSW-NB15 Dataset

Classification Report				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign (0)	0.03	1.00	0.05	1
Malicious (1)	0.00	0.00	0.00	36
Accuracy	0.03			37
Macro Avg	0.01	0.50	0.03	37
Weighted Avg	0.00	0.03	0.00	37

Table 8: Confusion Matrix of global model of FL using UNSW-NB15 Dataset

Confusion Matrix:		
Benign (0)	1	0
Malicious (1)	36	0
	Benign (0)	Malicious (1)

Table 7&8 show the evaluation of Global Federated learning training model using UNSW-NB15 dataset.

For all these reasons, the dataset UNSW-NB15 was not fit for the purposes of this research on secure and accurate traffic classification in IoT-oriented federated learning environments using representative, heterogeneous, and device-aware datasets.

- **ToN-IoT Dataset:** There were also a number of important drawbacks regarding the performance of the model and relevance to the research, which eventually led us not to consider the ToN-IoT dataset born out of the Australian Centre for Cyber Security as a structured collection of logs and network traffic generated from IoT environments.

  One, although data surfaced from IoT endpoints, the most distributed attacking scenarios, even logging attacks, were mainly designed for fog-level systems and intermediary infrastructure rather than the devices. Hence, almost all telemetry from the IoT endpoints was irrelevant for teaching traffic classification development models. That gap tied down even more the value of recording entirely in endpoint-specific adversarial behavior or federated learning applications used.

  It was created artificially in a laboratory setup and hence there was an artificial separation between telemetry streams and network traffic. This was weakness for joint inference models and system logs show very little correspondence to the resulting network behavior. Attempts at training classifiers having casually used only telemetric signals, show a high false positive rates indicating benign data passing out of the model as malicious; very clearly, it states failure in well-defined boundary decisions-critical for good performance of traffic classification in adversary cooling levels in other words.

  The below figure 7 shows the confusion matrix of a local traffic classification model at one of the end devices in the IoT Network and table 9 shows the classification report of the global traffic classification.

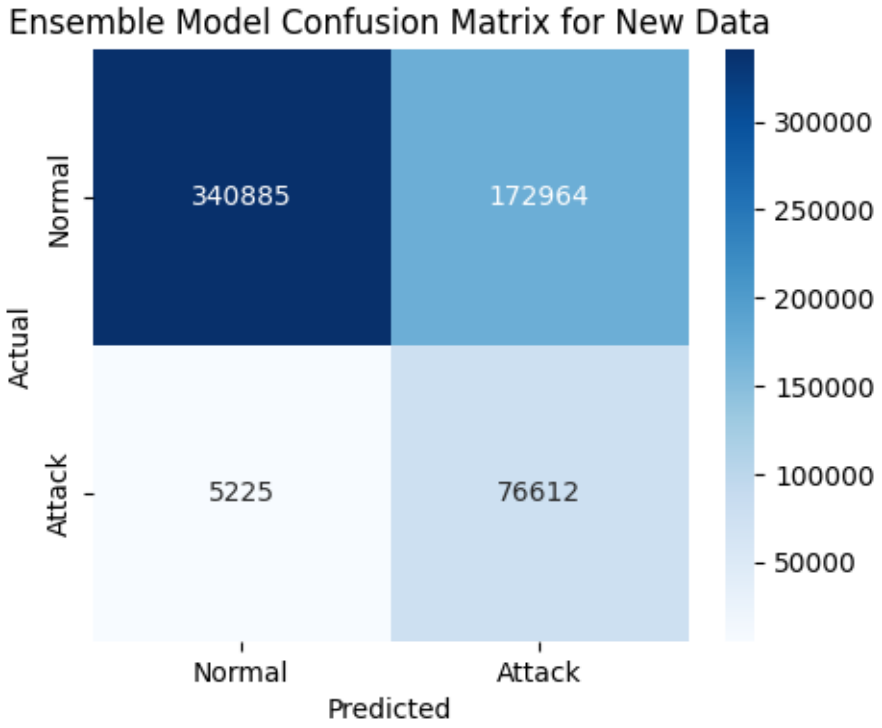

Figure 7: Confusion Matrix of a Local IoT Device in the network.

Table 9: Classification of Local XGBoost model using ToN-IoT dataset.

Classification Report				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign (0)	0.99	0.67	0.80	102804
Malicious (1)	0.31	0.94	0.47	16334
Accuracy	0.71			119138
Macro Avg	0.65	0.81	0.64	119138
Weighted Avg	0.89	0.71	0.76	119138

In this experiment, different IoT devices like GPS tracker, smart fridge, etc acted like the local clients. Each client were trained using XGBoost Model and then a global Model was created using the FedAvg function and these global updates were then sent to the local models to makes amendments if applicable to local models. Table 9 shows the Evaluation Parameter of the local XGBoost Model using ToN-IoT dataset as input.

While it shares the above advantage of labelled traffic, with the likes of UNSW-NB15, the quality of the flows generated remained decidedly mediocre. The generation of synthetic IoT logs restricted to a ceiling the dataset's potential to reflect conditions of realistic deployment and multidevice scenario attacks coherently and lifelike. Therefore, it left underrepresented many of the key dynamics at the attack surface that were relevant to evaluation of poisoning attacks under federated conditions.

Instability in training dynamics resulted from several preprocessing methods that's been applied during the more experimental trials with ToN-IoT. Bad convergence and small generalization were observed to all models even with proper fine-tuning in terms of training. Hence, it could not offer the research objectives that rely on trustworthiness and scalability in training-at-both binary and multiclass classification frameworks.

Given these compounding limitations, the ToN-IoT dataset was not selected for use in this thesis. Instead, the chosen dataset, IoT-23, had better fit the desired attributes for modelling realistic device-level attack scenarios in IoT and aligned better with the federated and centralized adversarial learning frameworks that will be explored in this research.

Therefore, the IoT dataset we decided upon uses a large amount of packet training data from real IoT devices, where tenuous feature engineering would capture suitable alignment with FL training paradigms.

4.4.2 Pre-Trained Model

To ensure state-of-the-art approaches were adopted for the initial rounds of experimentation, the two transformer-based models considered are:

1. **ET-BERT:**

ET-BERT (Encrypted Traffic BERT) is a deep learning model aimed towards the classification of network traffic, especially for encrypted and obfuscated flows, based on Transformer-based architectures took over by Liu et al. (2021). The important features for ET-BERT are using the paradigm of natural language processing (NLP) and applying it into the network traffic representations; tokenizing, embedding, and classifying sequences of network packets as a type of "language."

The general premise of ET-BERT is: to take advantage of the Bidirectional Encoder Representations from Transformers (BERT) model, which apparently performs exceedingly well at the natural language processing tasks such as masked language modelling and next sentence prediction, and hence repurposes this architecture

for grasping the patterns of raw traffic in the source scenario where payloads of packets are encrypted beyond the reach of traditional deep packet inspection (DPI) [31].

Architecture Overview:

The architectural overview of ET-Bert is shown in Figure 8. This figure shows the overview of ET-BERT pre trained model and how ET-BERT is trained over two stages:

1) **Pretraining**: This phase will be on huge amounts of unlabelled network traffic and through the following processes:
   (a) Masked Byte Modelling (MBM): Random byte tokens in the packet stream are masked and predicted.
   (b) Segmented Byte Prediction (SBP): Predicting the presence of a following byte segment given a context segment.

2) **Fine-tuning**: Traffic labels for downstream tasks like:

- Encrypted traffic classification
- Application identification
- Malicious traffic detection

The input representation contains tokenized byte sequences extracted from network flows. These are embedded into vector space and passed through multi-head attention layers and position encoders for classification.

Use Cases and Benefits:

- Effective on encrypted traffic, where payload inspection is not possible.
- It generalizes well to unknown traffic patterns due to prior pretraining.
- Supports classic binary and multi-class classification tasks.
- Suitable for federated learnings through local fine-tuning.

Being deep in Transformers, ET-BERT is computationally intensive and usually needs some GPU acceleration for training and inference; yet it represents the state-of-the-art approach to modern network environments where encrypted or adversarial traffic is often encountered [31].

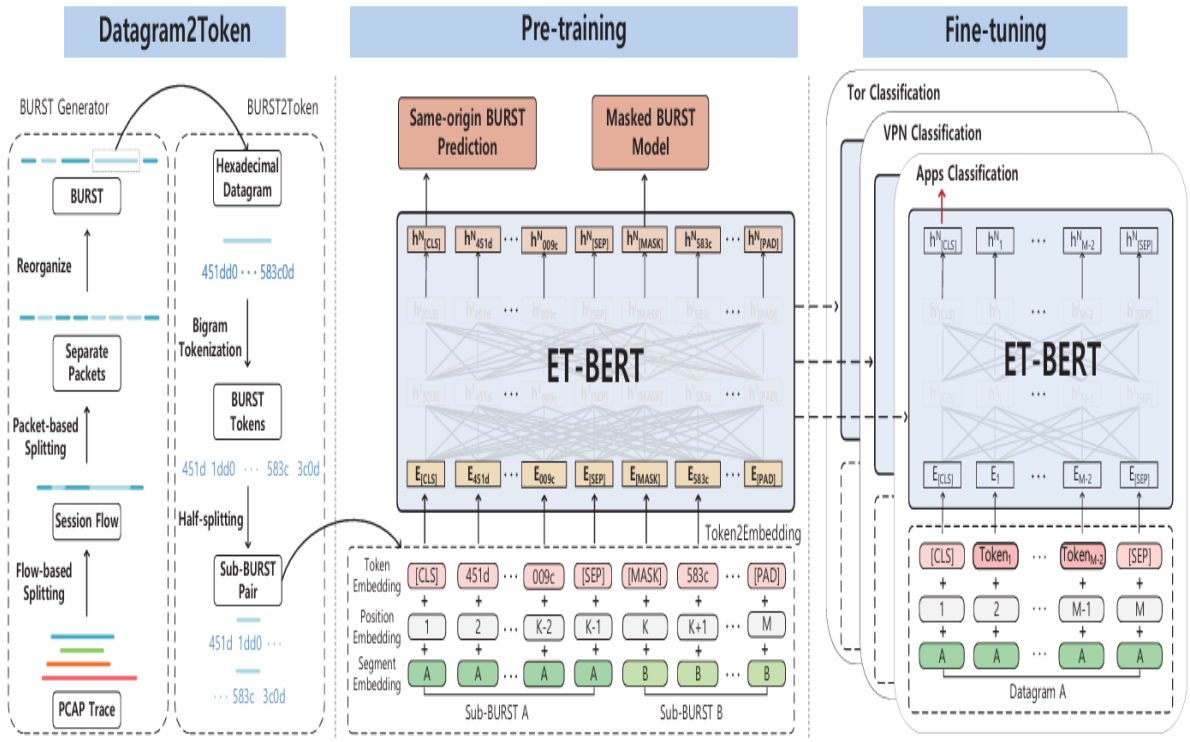

Figure 8: Overview of ET-Bert [31]

Challenges of ET-BERT:

This pre trained model caused challenges in processing our selected dataset "IoT-23" which does not have any kind of encrypted payloads. As ET-BERT specializes in traffic classification with a encrypted data being parsed as an input. As our dataset lacked encrypted payload working with this pre-trained model was a huge challenge.

ET-BERT model targeted traffic classification based on masked byte modelling. Able to do the job, it nevertheless posed challenges in terms of very heavy preprocessing (MBM encoding, SBP encoding) and required GPU resources for fine-tuning and inference. Any efforts to couple ET-BERT with extracted IoT-23 dataset flow sequences always failed due to hardware constraints and lack of encrypted payloads in the dataset.

2. NetMamba:

NetMamba, a new lightweight yet quick pre-trained encoder for representing traffic over network topology, was introduced by Xu et al. (2023) [32] in a bid to address the growing needs for a low-resource operable, high-accuracy traffic-classification and anomaly-detection model, like those to be found in IoT gateways and edge routers.

NetMamba is the very first model to integrate pre-training with resource-efficient fine-tuning, similar to that of NLP-based transformers. Unlike deep and computationally demanding architectures like ET-BERT, the model is fast, compact, efficient, quick, and most importantly I meant for real-time and embedded systems.

Traffic Representation Pipeline:

Before raw packet data enters the NetMamba model, it has standard data transformation pipelines: i.e.,

1) **Flow Splitting**: aggregation of packets by 5-tuple flow keys (IP, port, protocol).
2) **Packet Parsing:** headers and payloads are converted to fixed-size byte arrays.
3) **Packet Cropping and Padding**: packets are normalized to fixed size.
4) **Byte Concatenation**: packets from a single flow are appended to form one array.
5) **Stride Cutting**: long byte strings are split into smaller overlapping strides, which serve as input tokens for the model.

The traffic representation can be seen in Figure 9 below.

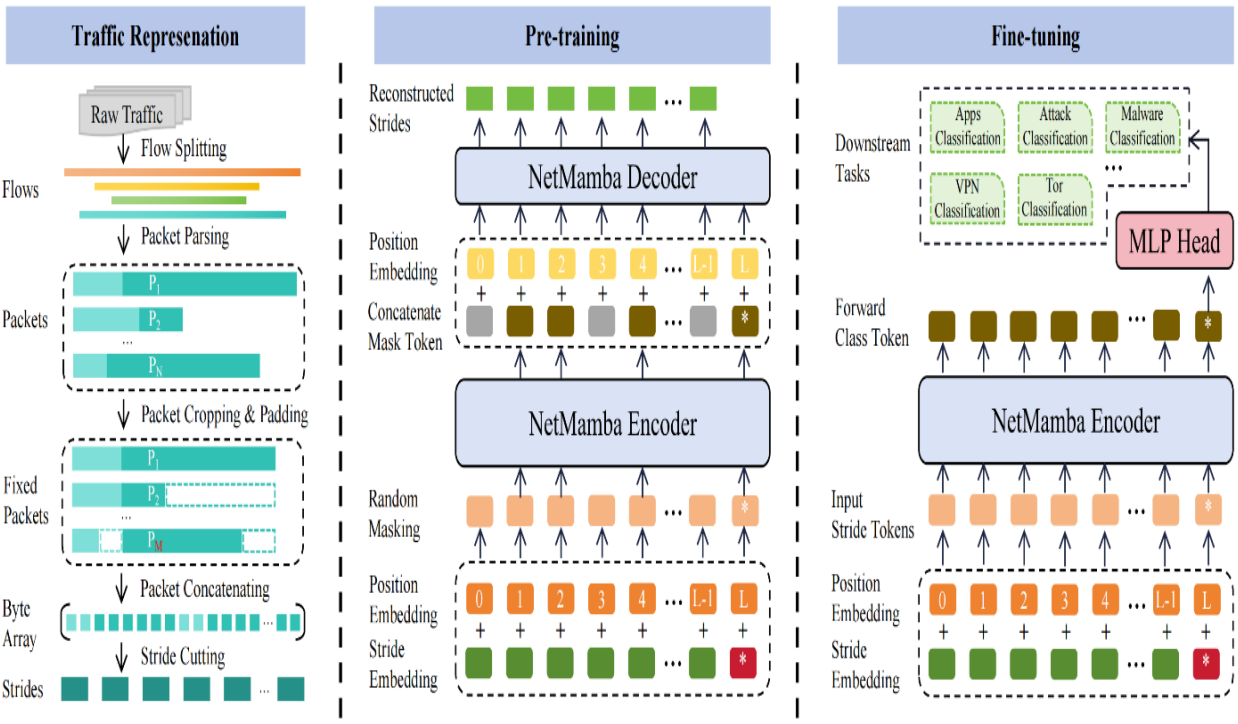

Figure 9: Overview of NetMamba Model [32].

Architecture:

The overall architecture of NetMamba is shown in Figure 9.

- The NetMamba architecture models byte-level representations of traffic flows with lightweight convolutional and attention-based encoders.
- Supports pretraining based on masked stride prediction and next stride prediction similar to BERT's MLM and NSP tasks.
- It was fine-tuned to adapt to a variety of tasks like: Botnet detection, Application identification, IoT network intrusion tolerance.

Key Benefits:

- Fast and memory-efficient, hence applicable for edge deployment.
- Pretraining has no labels that can become scalable for unlabelled data set size.
- Pulling even better than or comparable with heavyweight models such as ET-BERT in low resource settings.
- Slices real-time traffic analysis in IoT scenarios [32].

Challenges of NetMamba Model:

a) NetMamba has specific advantages when applied in federated learning where each node suffers from limited processing power but must still learn from traffic. Therefore, the design also allows incremental learning and efficient model aggregation
b) For encoding network flows into stride sequences, it provides security and efficiency. The stride cutting approach and traffic encoder component showed promise but required heavy GPU compute and memory exceeding the limits of the available resources.

Regardless of what the pre-trained models were supposed to do, these experiments furnished valuable insight into model complexity, preprocessing problems, and issues of resource feasibility. The experiments further solidified the need for tailor-made lightweight models for edge-level deployment and led to the adoption of the deep learning model like RNN-based classifier architecture and machine learning model like XGBoost based traffic classifier architecture.

4.5 Transition to Implementation and Solution Framework.

The limitations identified through these explorations shaped the next development of a custom architecture based on:

1) Sequential pattern recognition using RNNs (suited to time-ordered packet flow data),
2) Traditional Machine Learning Model like XGBoost performance would suit the selected dataset,
3) Label balancing and feature normalization to counter dataset imbalance and inconsistencies,
4) Controlled label-flipping and gradient poisoning simulations to assess resiliency, and
5) Testing under realistic IoT limitations such as low processing power and the absence of complete ground-truth data.

All these factors ensured that the final system was built to address the actual research problem while also simulating a deployment scenario in real-world IoT FL systems.

This chapter clarified the rationale behind the design, the architecture of the system, and the research problems that ultimately led to the implementation. The customized federated learning security pipeline was built through data-set evaluations, architectural experimentation, and resource-based trade-offs. These design decisions are practically congruent with the real-world limitations of IoT deployments and therefore lead the way and set the ground toward eventual results and evaluation.

5 Experimental Design and Setup

In this chapter, we build from the motivations and insights presented in the previous chapter to detail the experimental framework devised to investigate poisoning attacks in a federated IoT environment. The experimental setup is grounded in parallel scenarios of modelling, the first involving a deep learning approach via a recurrent neural network (RNN) and the second using a traditional machine learning algorithm which is XGBoost. The two approaches are evaluated in terms of the effectiveness for binary traffic classification (benign versus malicious) and multi-class classifications for malware detection.

Central to the rationale for different modelling paradigms was the need to establish how susceptible two forms of classifiers, namely neural and tree-based classifiers, are to attacks from poisoning approaches, taking a step further to offer an insight regarding sort-specific attack behavior. By binary and multiclass formulation, the design of the experiment promises a nuanced understanding of how different forms of attacks may meddle with classification granularity at various levels.

The primary agenda of this research revolves around the detection and classification of network traffic in an Internet of Things (IoT) setting while evaluating the robustness of the proposed models against adversarial attacks. The experimental domain revolves around the following two modelling approaches:

- Deep Learning (DL): Using Recurrent Neural Network (RNN)
  Using RNNs, a special type of neural network mainly used for sequence modelling, time-varied dependencies can be captured owing to the presence of hidden states which change as inputs are fed along time steps. Due to this feature, RNNs are suitable for modelling traffic flow sequences, in which such input tokens represent a packet in a flow.

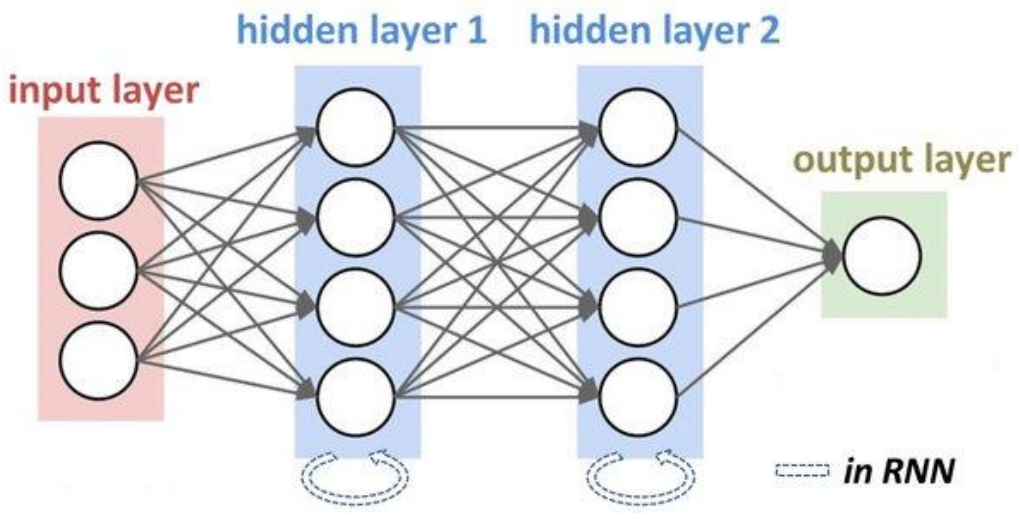

Figure 10: Generalized RNN Model with two hidden layers

In this work, tokenized flows from network traffic are fed to the RNN, enabling the model to learn the characteristics in sequences of packets with exactly what is considered benign or malicious behavior. The architecture is that of:

- An embedding where the tokens are represented in the dense vector space
- One of several one or two RNN or LSTM layers
- The last fully connected layer for classification with softmax/sigmoid activation

- Traditional Machine Learning (ML): Using XGBoost
  XGBoost, which stands for Extreme Gradient Boosting, is a powerful machine-learning algorithm based on an ensemble of decision trees. It is very simple to use- train a new tree that tries to correct the mistake of the previous new one into the series of multiple trees. Figure 11 shows the architectural view of the XGBoost Model

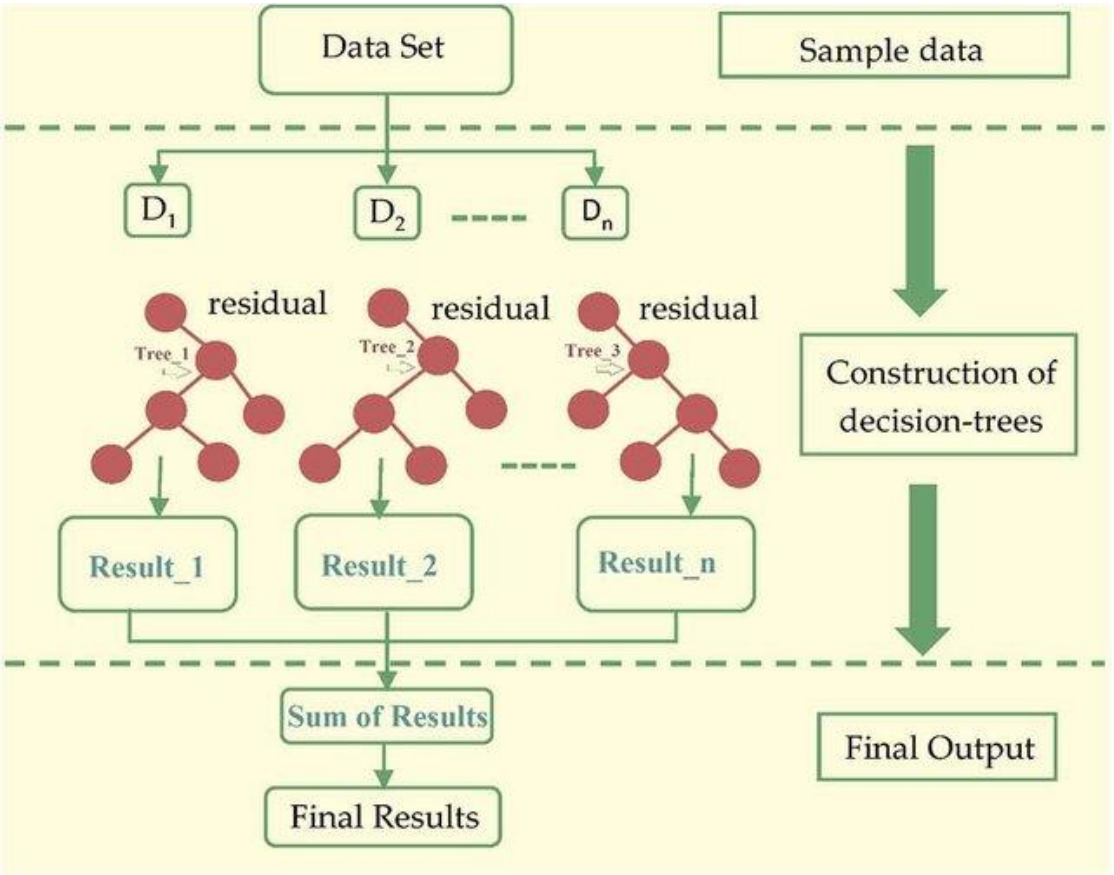

Figure 11: General Architecture of XGBoost [33]

Thereby generating a good model, capable of excelling in binary and multiclass classification tasks, particularly using very structured or tabulated data. As Dimension of analysed flow, 'packet length,' 'inter-packet time,' 'protocol type,' and 'flow duration,' the various flows under consideration have been extracted, pre-processed, and packed into vectors of fixed length. They are input to XGBoost, which uses gradient boosting to find the complex decision boundaries between the benign and malicious traffic patterns.

XGBoost was selected due to:

- High speed and scalability
- Imbalanced class handling is built into the system.
- Feature importance interpretability

Both models will be trained and evaluated on two different facets:

- Binary Classification-to define either benign or malicious traffic.
- Multiclass Classification-to identify the specific malware or attack family.

In both conditions, the same preprocessing pipeline has been employed for extracting and formatting features from raw traffic data. Then the generated models become susceptible to poisoning attacks comprising:

- Targeting training integrity - Mislabelling Attacks
- Multiclass-Targeted Malware Bypass Attacks

Here's how the general architectural flow for traffic classification in IoT environments looks like: Refer Figure 12.

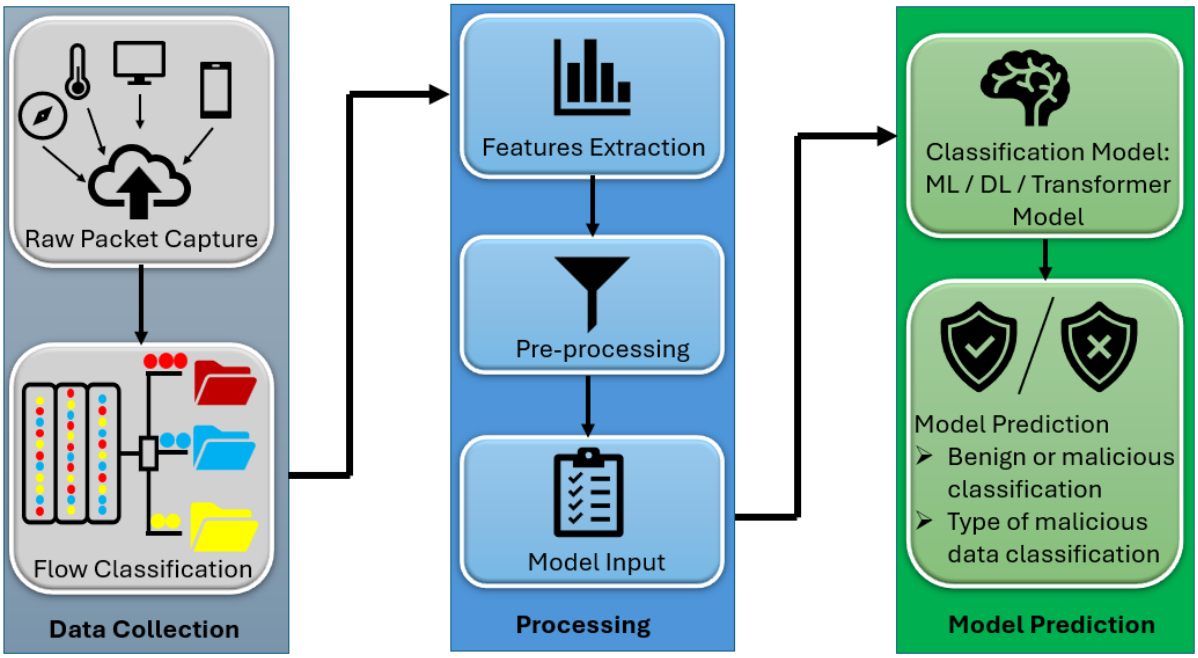

Figure 12: Traffic Classification in IoT Environment.

5.1 Data Preprocessing

Critical stage towards developing a strong traffic classification system is where raw packet level data commences conversion into well-structured and labelled feature representations that will subsequently be suitable for deep learning and traditional machine learning pipelines. This section discusses preprocessing of data that is utilized in the thesis; from raw packet captured by Wireshark tool in form of (.pcap) files to fully labelled feature matrices, along with the justification of model configurations for both Recurrent Neural Network (RNN) and XGBoost classifiers.

5.1.1 Data Packet Parsing and Feature Extraction using Scapy.

The initial raw dataset was collected in the form of .pcap files having network traffic obtained from the IoT-23 dataset. The capture is from an emulation of communication patterns that are observed from different IoT devices during all the simulated attack and benign scenarios. *Scapy* which is a powerful Python library for packet manipulation was used to parse and extract important packet-level attributes from these .pcap files [34].

The pcap files are read and then converted into packets present in the network using the command.

packets = rdpcap(pcap_file)

The network flow is defined using the 5-tuple enumeration:

src_ip, dst_ip, src_port, dst_port, protocol

Packets sharing such 5-tuples are considered to belong to the same flow, and within that flow, metrics are computed both on a packet-wise and flow-wise basis to represent the traffic patterns.

These features have then been extracted or calculated:

1) **Timestamp and Identifier Fields**
   - ***ts:*** Raw epoch timestamp (in seconds) of when the packet was captured. It is extracted using the packet.time feature of Scapy library.

- ***formatted_time:*** Human readable timestamp derived from '*ts'* in the format YYYY-MM-DD HH:MM: SS to allow for debugging and visualization.
- ***src_ip, dst_ip:*** Internet Protocol (IP) addresses of the packet sender and receiver respectively.
- ***proto:*** The layer 4 transport protocol used in the packet (e.g. TCP, UDP, ICMP)
- ***src_port,dst_port:*** Port numbers used by the sender and receiver of the packet.

Table 10: Timestamp and Identifier Fields Feature List.

Feature Name	Defination	Data Type
ts	Raw epoch timestamp of the packet capture using the packets.time feature of Scapy library.	Float
formatted_time	Human readable timestamp derived from '*ts'* in the format YYYY-MM-DD HH:MM: SS to allow for debugging and visualization.	String
src_ip, dst_ip	Source and Destination Internet Protocol (IP) addresses.	String
Proto	Protocol Identifier (TCP, UDP, ICMP) captured from the IP header. The identifier are mapped to numbers like UDP =7, TCP =16 and ICMP	Integer
src_port, dst_port	TCP or UDP port numbers. Such values together with the IP addresses and the particular protocol define the flow.	Integer

The Table 10 shows the features which are identifier fields and respective description along with the necessary details.

2) Basic Packet Characteristics

Table 11 shows the features that give us a packet characteristic which will help us to train the model to learn the characteristics of a benign data packet and the deviations that are observed in the malicious data packets.

Table 11:Basic Packet Characteristics Feature List

Feature Name	Description	Data Type
packet_length	Total length of the packet in bytes, header and payload combined.	Integer
tcp_flags	Identify which flags (SYN, ACK, FIN, etc) for TCP packets are noted to detect connection states.	String

3) Temporal Statistical Features

- ***inter_packet_time:*** The time difference between the present packet and the previous packet in the same flow would shed light on the measure of burstiness or steadiness of the communication. If it is the first packet in the flow, this value is set to 0.

$$\boldsymbol{inter\_packet\_time_i = ts_i - ts_{i-1}}$$

- ***flow_duration:*** It is the difference between the timestamps of the last and first packets in the flow. This indicates how long the flow persisted.

$$\boldsymbol{flow_{duration} = ts_{last} - ts_{first}}$$

- ***mean_inter_arrival_time:*** It is the average of all inter-packet times in a flow. This reflects the regularity of the packet exchanges.

$$\boldsymbol{mean\_inter\_arrival\_time = \frac{\sum_{i=2}^{n}(ts_i - ts_{i-1})}{n-1}}$$

4) Aggregated Flow Statistics

- ***total_packets_in_flow:*** It represents the total number of packets present in each flow.

- ***total_bytes_in_flow:*** Sum of the size of all the packets within the flow. It is represented in bytes.

$$\boldsymbol{total\_bytes\_in\_flow} = \sum_{i=1}^{n} \boldsymbol{packet\_length_i}$$

- ***mean_packet_size:*** Average size of the packet within a flow.

$$\boldsymbol{mean\_packet\_size} = \frac{\boldsymbol{total\_bytes\_in\_flow}}{\boldsymbol{total\_packets\_in\_flow}}$$

- ***packet_per_second:*** It indicates the rate at which the packet is transmitted in the flow. This is calculated by dividing the total packet in the flow by duration of the flow.

$$\boldsymbol{packet\_per\_second} = \frac{\boldsymbol{total\_packets\_in\_flow}}{\boldsymbol{flow\_duration}}$$

- ***bytes_per_second:*** This indicates the throughput rate.

$$\boldsymbol{bytes\_per\_second} = \frac{\boldsymbol{total\_bytes\_in\_flow}}{\boldsymbol{flow\_duration}}$$

Then the collected flow level attributes were arranged into a tabular structure and saved as a .csv file (features.csv) following the schema below as shown in table 12:

Table 12: List of all the features extracted.

Extracted Features List					
ts	formatted_time	src_ip	dst_ip	proto	src_port
dst_port	packet_length	inter_packet_time	flow_duratoion	total_packets_in_flow	mean_packet_size
mean_inter_arrival_time	packets_per_second	bytes_per_second	tcp_flags	label	detailed_label

Every row at this point do not contain any kind of labels. The labelling of these features will be explained in the later subsection part.

5.1.2 Ground Truth Label Mapping

The '***conn.log.labeled'*** file was delivered along with every pcap scenario and contains the ground truth attack labels. Each entry in this file included the following meta information:

- Timestamp
- Connection UID
- Source IP and port
- Destination IP and port
- Label (Specifying whether the packet is benign or malicious
- Detailed-Label (Specifying the attack type for e.g. Port Scanning, DDoS, Command & Control)

Labelling of each flow in features.csv was based on the following steps:

- **Timestamp matching:** The timestamp of the pcap-derived features was then compared with the labelled log using a closeness threshold (in seconds) to allow some tolerance for time desynchronization.
- **IP and port matching:** The "*src_ip*" and "*src_port*" were cross-referenced against the information located in the labelled log.
- **Label propagation:** Once there was a successful location match, the "*label"* and "*detail-label*" were transferred from the log to the feature set.

This ensured that most flows were confidently assigned labels; however, a group of flows remained still. At this point the group of flows without any matches from the conn.log.labeled file were labelled as "*unknown"*

The unknown labels were then intelligently labelled using a small logic. This is explained in the preceeding subsection.

5.1.3 Intelligent Labelling of Unknown Entries

To minimize label sparsity, an intelligent label inference mechanism was implemented by using the flow context. For each "*Unknown"* entry:

- If the "***mean_inter_arrival_time***" ≤ 1 second:
  The label was inferred from the last known labelled packet of the flow.
- If the "***mean_inter_arrival_time***" > 1 second:
  Then entry would be benign since it is presumed that the packet forms part of a new or idle communication, which is not tied to previously malicious behavior.

The threshold of 1 second for mean_inter_arrival_time has been chosen with regard to the common temporal behavior exhibited by malicious and non-malicious traffic. Malicious flows typically show a bursty behavior with relatively short inter-arrival times because of fast-paced activities like exploitation or scanning, whereas benign traffic, especially in IoT scenarios, will have longer, periodic intervals of communication. Hence, using a 1-second threshold will essentially act as a distinguished boundary between active malicious flows against inactive benign ones or mere routine benign ones, consistent with what has been established earlier in the analysis of network behaviours under similar scenarios

This heuristic came from the assumption that flows with high inter-arrival latencies tend not to be temporally continuous with malicious patterns and therefore default to benign classification.

5.1.4 Data Cleaning and Normalization

Data cleaning and transformation pipeline underwent a long processing phase after feature extraction and label assignment to ensure the dataset was aptly ready for training robust machine learning applications. The primary goal of this phase was to get rid of the noise, standardize the representation of features, and enhance converging model generalization.

1) Removal of Incomplete or Malformed Flows

All network flows which were either incomplete or malformed were stripped out from the dataset. Specifically Here:

- Flows with even a single missing value in any of the feature columns were eliminated or parameter with null values were handled in such a way in order to prevent the propagation of error while training with this model.
- Flows with fewer than 2 packets were rejected as being such ephemeral connections, usually, have no value in behavioural pattern analysis because there is not enough time- or volumetric-wise information to distinguish the benign from malicious traffic.

Properly structured flows that exhibit measurable behavior would be retained for this further processing step.

2) Outlier Detection and Removal

Extreme values should not be allowed to affect training processes; hence an outlier filtering step was employed:

- Flows with extremely long durations (beyond the 99.9th percentile of flow duration) were removed from the dataset. They are often the result of logging problems or anomalous sessions that occur at very rare intervals, which tend to mislead the classifiers.
- Furthermore, we removed flows with total byte sizes or packet counts greater than the 99.9th percentile, eliminating case of massive data transfer or abnormal buffering patterns that did not resemble a typical IoT traffic pattern.

This statistical pruning contributed to reducing the variance of the training set while contributing to the improvement of convergence during training.

3) Categorical Feature Encoding

Machine learning models, tree based and neural ones, cannot directly work with string-based categorical variables. Therefore, categorical variables were converted into numerical format using the One-Hot Encoding technique.

- These protocols are mapped to the numeric values (TCP = 6, UDP = 17, ICMP = 1). This mapping comes from IANA protocol number registry. After mapping the proto is then applied with one-hot encoding.
- **One-Hot Encoding** converts a categorical variable containing k distinct values into k binary features. For example, consider the proto column, in which TCP, UDP, and ICMP can be found. It will be broken down into three distinct columns: proto_TCP, proto_UDP, proto_ICMP. To indicate as available the respective protocol and as unavailable in all other cases, a value of 1 is assigned.

This transformation gives categorical information a readable form for learning algorithms and does not assume any ordinal relations between the categories (which is what would inappropriately result from simpler label encoding).

The same would apply to the tcp_flags column as it causes protocol-specific flag information for a flow state analysis.

4) Feature Normalization

On encoding categorical features, the rest of the numeric variables were normalized using min-max scaling across a fixed range of [0,1].

- Min-max scaling is a normalization process wherein each numeric feature x is rescaled according to the following equation:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

  Thus, relative distributions and scales of values are preserved by this transformation. However, all important features will have equal contributions to model training. This is very important for neural networks, as they are very sensitive to feature scale while optimising gradient values.

- The columns normalized include continuous metrics e.g.:
  i. packet_length, inter_packet_time, flow_duration
  ii. total_packets_in_flow, total_bytes_in_flow
  iii. mean_packet_size, mean_inter_arrival_time
  iv. packets_per_second, bytes_per-second

  Normalization will reduce bias in the model towards high-magnitude features and includes faster convergence during training.

5) Data Partitioning

In the end, the cleaned and processed dataset was segregated into two non-overlapping subsamples:

- **Training Set**: Used for training both deep learning classifiers and tree-based classifiers.
- **Unseen Test Set:** Kept aside for eventually testing generalization performance of models towards traffic patterns never seen during training.

Two types of classification targets were prepared:

1. Binary Classification:
   - Purpose: To differentiate malicious from benign traffic.
   - Labels: Malicious = 1, Benign = 0.

2. Multiclass Classification:
   - Purpose: To further identify specific subcategories of attacks that include DDoS, PortScan, C&C communication, etc.
   - The labels are directly derived from the detailed_label column in the dataset.

Such dual use of the labelling permits performance evaluation in the context of different practical operations-from intrusion detection to identification of threat types.

5.2 Data Preparing Model Input: Sequential Tokenization (RNN) and Tabular Transformation (XGBoost)

Initiating any machine-learning models requires the conversion of pre-processed and labelled traffic data into format inputs that are customized for the given model architectures. In our case, two contrasting but somewhat complementary learning paradigms will be presented: Recurrent Neural Networks (RNNs) for the temporal capture of traffic sequences, and XGBoost for the learning of very well-structured tabular data by means of decision-tree ensembles.

As they have differing demands, thus we followed two different data preparation pipelines:

- RNNs are subjected to a tokenization pipeline in which packet flows are transformed into fixed-length, semantically meaningful tokens, retaining key aspects of temporal and contextual information.
- XGBoost is given a feature vector representation in which entries are treated as flattened vectors that offer a summary of data regarding the statistics and structure of that flow.

This section illustrates the complete data preparation pipeline for each model, as a primer to the next chapter where we shall be discussing the architectural details of the models, hyperparameter ranges, and training strategies for binary (benign or malicious) and multiclass (per attack type) classification scenarios in detail.

5.2.1 Preprocessing for RNN Input: Tokenization of Traffic Flows.

In traditional methods, traffic classification approaches use numeric feature vectors aggregated either per-flow or per-session for IoT networks. These techniques, however, do not have much strength in modelling the sequential nature of the packet exchanges, which is vital to classify the changing attack behaviours. To remedy this, we adopt a sequence modelling approach inspired by Miettinen et al [13], where network flows are considered as sequences of symbolic tokens rather than flat vectors, using RNNs.

Rationale Behind Flow Tokenization

Each packet in a network is endowed with categorical and numeric metadata-protocol type, destination port, TCP flags, and packet size. These may, however, be abstracted at a higher semantic level into tokens representing the behavior of that packet with respect to a flow.

The major motivating factors for tokenization involve:

- **Sequence Preservation:** In contrast to aggregate statistics, token sequences conserve the chronological relationship of events.
- **Vocabulary Reduction:** Discretized continuous features (such as packet size) yield finite vocabularies suitable to foliate discrete, finite topologies for RNN training.
- **Contextual Embedding:** Tokenization may also be done and learned in a manner similar to NLP models, providing contextualization as well.

Tokenization Pipeline

Within the tokenization pipeline, the following is a 7-stage preprocessing of conversion of raw flow level packet metadata into token sequences:

Step 1: Binning of Packet Lengths:

Due to the inherent variability in the amount of each packet, the generation of the maximum likelihood estimator is not possible with respect to overfitting fine-grained packet length data.The following classes are used for discretization based on packet length:

- LEN_SMALL (<100 bytes)
- LEN_MED (100-499 bytes)

- LEN_LARGE (500-999 bytes)
- LEN_XL (≥1000 bytes)

This gives restriction of noise levels and thus smoothens the data for sequence modelling.

Step 2: Token Construction

For each packet of a flow, a combined token is created entirely of four fields put together:

`<protocol>_<destination port>_<packet length bin>_<TCP flag>`

For instance, if a packet is using TCP in port 80 with medium size and ACK flag applies, it might become:

`TCP_80_LEN_MED_ACK`

This would build a semantic token vocabulary that would represent packet behavior patterns.

Step 3: Flow Grouping

Packets are grouped into flows using the following 4-tuple:

$$flow\_id = src\_ip + dst\_ip + protocol + dst\_port$$

This prohibits sequence token communications between specific devices and services through the available session.

Step 4: Vocabulary and Label Encoding

Generation of Vocabulary and dictionary is as follows:

- A **token-to-index vocabulary** for embedding tokens during RNN training, and
- A **class-to-index dictionary** to encode the flow labels based on a first-most frequent attack class label in that flow.

Now only the top MAX_SEQ_LEN tokens are retained per flow, if short, padded for uniformity.

Step 5: Sequence Encoding and Padding

Thus, every flow becomes a token indices sequence, which is then padded either with a special token PAD=0 or shortened to a fixed length (MAX_SEQ_LEN = 100). Default for unknown tokens is UNK=1.

Step 6: Labelling the flow

The flow is labelled as “Malicious” if any of data packet in the flow is a compromised data packet. If not, then the flow is labelled as “Benign”

In case if we have multiple types of attack in one flow, then the majority of the compromised type of data packets in the flow is used.

Step 7: Saving Pre-processed Inputs

Finally, the following structures are saved:

- **tokenized_sequences.npy**: Shape [num_flows × MAX_SEQ_LEN]
- **labels.npy**: Class labels for each flow.
- **vocab_mapping.json:** Maps tokens to indices.
- **class_mapping.json:** Maps class labels (e.g., 'Mirai', 'DDoS', 'Benign') to indices.

This serves as input for training in the RNN model.

Table 13: Summary of Dataset Tokenization Parameters

Parameter	Value
Token Format	PROTO_DSTPORT_LENBIN_FLAGS
Max Sequence Length	100 tokens
PAD Token Index	0
UNK Token Index	1
Packet Length Bins	4(SMALL, MED, LARGE, XL)
Flow ID Construction	src_ip + dst_ip + proto + dst_port

Table 12 gives the summary of all the tokenization parameters that were set to create the data for feeding into the RNN Model.

Example Token Sequence

Just for example, it is possible to receive a simplified flow like this resulting in a tokenized sequence:

```
["TCP_80_LEN_MED_ACK", "TCP_80_LEN_LARGE_PSH", "TCP_80_LEN_MED_ACK", ...]
```

Then, in transformation:

```
[5, 21, 5, ..., 0, 0] # padded to 100 tokens
```

Integration with RNN Architecture

These tokenized sequences served as the input layer of an RNN-based classifier that learns to predict the attack type or benign class. While an embedding layer takes as input the token index and provides as output a dense vector, this representation allows the RNN to learn temporal and semantic relationships among packets in a flow.

A more in-depth discussion of the RNN architecture and training configuration will be provided in Section 5.3.

5.2.2 Preprocessing for XGBoost Input: Tabular Transformation.

As discussed in Chapter 5.1, the network traffic data was obtained from raw .pcap files and then converted into flow-level features. Each flow had been labelled according to heuristics and contextual conditions either binary (benign/malicious) or multiclass (attack type) criteria. The outcome of this preprocessing pipeline was a CSV file in which each row consisted of one network flow endowed with statistical features and labelled by its respective class.

The current section will deal with the changes of that labelled CSV dataset into an appropriate format useful for training an XGBoost classifier. Input samples for XGBoost which is an algorithm on gradient boosting decision trees, requires fixed-length and numerical feature vectors for each data sample. This is why this entire step is concerned with turning each labelled flow into a rich, well pre-processed tabular format.

Step1: Loading and Inspecting the CSV File

The preprocessing pipeline creates a CSV file that holds the numerical and categorical features, as well as the corresponding labels. Examples of columns found in this file are:

- **Flow level statistics:** flow_duration, packet_length, inter_packet_time, total_packets_in_flow, mean_packet_size, packets_per_second, and others.
- **Categorical features:** proto, tcp_flags.
- **Label columns:** label (for binary classification) and detailed_label (for multiclass classification).

This file will first be loaded into a pandas DataFrame for further processing:

```
import pandas as pd
df = pd.read_csv("packet_flow_features_labeled.csv”)
```

Step 2: Feature Selection

To ensure optimal learning occurs and overfitting is reduced, only relevant and non-redundant features should only be retained.

- **Excluded fields:** Columns like “*ts”, “formatted_time”, “src_ip”, “dst_ip”, “src_port”* and *“dst_port*” would be discarded because they have a limited discriminate power and might introduce the chaos.
- **Retained fields:** All numerical features and key categorical features (“*proto”, “tcp_flags”*) that encode essential traffic behavior.

This makes the set of features a high-quality subset:

```
selected columns = ['flow_duration', 'total_packets_in_flow', 'to-
tal_bytes_in_flow','mean_packet_size','inter_packet_time','pack-
ets_per_second','bytes_per_second','proto','tcp_flags','label','detailed_label']

df = df [selected columns]
```

Step 3: One-Hot Encoding of Categorical Features

As an extension, certain features require one-hot encoding since XGBoost does not natively support categorical strings:

```
df_encoded = pd.get_dummies (df, columns=['proto', 'tcp_flags'])
```

This, in effect, expands the dimensionality by converting every unique value in proto and tcp_flags into binary columns.

Step 4: Label Mapping

- Binary Classification: The “*label”* column will serve as the target (0 for benign, 1 for malicious).
- Multiclass Classification: The “*detailed_label*” column will be transformed into integers using LabelEncoder.

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df_encoded['detailed_label'] = le.fit_transform(df_encoded['detailed_label'])
```

The label encoder’s mapping is saved for future decoding of predicted results.

Step 5: Feature Scaling

Even though XGBoost can accept an unscaled dataset, improving convergence or generalization on highly skewed features through standardization or normalization is advantageous. The code below illustrates this.

```
from sklearn.preprocessing import MinMaxScaler


scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(df_encoded[numerical_columns])
```

Scaling is performed for numerical columns alone, leaving the one-hot-encoded features unchanged.

Step 6: Prepare Final Input Matrices

Finally, upon the completion of preprocessing.

- X → Feature matrix (either a NumPy array or a pandas DataFrame)
- y → Label vector (either label for binary or detailed_label for multiclass)
- w (optional) → Sample weights based on the class distribution

```
X = df_encoded.drop(['label', 'detailed_label'], axis=1)
y_binary = df_encoded['label']
y_multiclass = df_encoded['detailed_label']
```

To weight the samples, we use

```
from sklearn.utils.class_weight import compute_sample_weight

weights = compute_sample_weight(class_weight='balanced', y=y_binary)
```

Step 7: Convert to DMatrix Format for XGBoost

The last step is to convert the dataset into a DMatrix used in the XGBoost classification:

```
import xgboost as xgb

dtrain_binary = xgb.DMatrix(X, label=y_binary, weight=weights)
dtrain_multi = xgb.DMatrix(X, label=y_multiclass, weight=weights)
```

This format allows efficient training and evaluation in the XGBoost framework.

This tabular transformation pipeline kicks off on the heels of the labelling flow dataset being stored as a CSV document (Chapter 5.1) and priming a structured input for model consumption for the XGBoost classifier. With the correct encoding, scaling, and weighting, the pipeline is set up for robust learning for both binary and multiclass sub-tasks of traffic classification.

The next chapter discusses the training set-ups and hyperparameter configurations for XGBoost and RNN models across various classification paradigms.

5.3 Model Design and Initialization – RNN & XGBoost Model

This study focuses on the two extremes of machine learning paradigms: sequential pattern analysis for anomaly detection with RNNs and interpretable decision-based classification with the gradient-boosting decision tree model XGBoost. Each model is trained in binary (malicious vs. benign) and multiclass (depending on the attack type) scenarios. This twofold classification of the model helps in analyzing how each algorithm copes with varying levels of labelling granularity.

5.3.1 Recurrent Neural Network (RNN): for Sequence Modelling

RNNs were constructed to process temporal sequences and suit well in network traffic analysis, where each flow can be considered a time series of packet-level or flow-level features. Therefore, in our conception of RNNs, the sequence of packets in a flow will carry significant behavioural signatures—like inter-arrival delays and fluctuating byte volumes—that can characterize the behavior as either normal or attack-like.

Model Architecture:

Architecture of the RNN model was initialized as follows:

- Input Layer:
  - It takes a fixed-length padded sequence of time-ordered flow features.
  - Input dimensions: [*batch_size, sequence_length, feature_dimension*]
  - Example: Each input sample is a matrix of shape (20 packets × 18 features).

- RNN Layer (LSTM):
  - We have used long short-term memory (LSTM) units whose memory is designed for long-term dependencies so as to mitigate the vanishing gradient problems.
  - *hidden_size* = 128: It defines the size of hidden state vectors.
  - num_layers = 2: It allows the network to learn more complex representations through hierarchical feature abstraction.

- Dropout Layer:
  - *dropout_rate* = 0.3: Introduced between LSTM layers so that overfitting can be avoided by randomly dropping a certain number of neurons from the network during each iteration of training.

- Fully Connected Output Layer:
  - Maps final LSTM output to label space.
  - Output dimensions:
    1 neuron (with Sigmoid activation) for binary classification.
    n neurons (with Softmax activation) for multiclass classification (where n is the number of attack classes).

Loss Function and Optimizer

- Loss Function:
  - Binary Cross Entropy (BCE) for binary classification.
  - Categorical Cross Entropy for multiclass classification.
- Optimizer:
  - We selected the Adam optimizer due to its suitability for sparse gradients and noisy data. The learning rate was set to 0.001 based on our empirical observation of its stable convergence across epochs.
- Class Weights:
  To deal with class imbalance, especially in binary classification where benign samples vastly outnumber malicious ones, we applied inverse frequency class weighting in our loss function.

Training Parameters:

Table 14 shows the hyperparameters that are set for the RNN Model to be trained. These parameter on fine tuning can increase the performance of the model.

Table 14: Training Parameters of RNN model.

Parameter	Values
Batch_size	64
Epoch	6
Learning rate	0.001
Optimizer	Adam
Weighing Scheme	Inverse class frequency
Padding Strategy	Zero-padding (pre or post sequence)

5.3.2 XGBoost for Tabular Classification

XGBoost (Extreme Gradient Boosting) applies gradient boosting with the highest performance, unlike RNNs, to directly implement some tabular modelling and structure data in which feature interactions, as well as feature importance, can be explicitly modelled. More importantly, it is also interpretable, hence one can analyse the features that correlate most with the classification of traffic types.

Feature Representation

- Input features are equivalent to the output of the cleaned and normalized features.csv dataset.
- XGBoost maintains per-flow snapshots without succession or temporal ordering.

Model Configuration

Table 15 represents the parameters that are set in the XGBoost Model for initializing the model.

Table 15: Model Parameters for XGBoost Model.

Parameter	Values	Description
n_estimators	100	Number of trees to build
max_depth	6	Limits complexity of individual trees
Learning rate	0.1	Step size shrinkage used to prevent overfitting
subsample	0.8	Fraction of training data used per boosting round
colsample_bytrees	0.8	Fraction of features used per tree
scale_pos_weight	Inverse class frequency	Helps correct for class imbalce.

Training Objectives

- Binary Classification:
  - Objective function: binary:logistic
  - Evaluation metric: logloss, AUC, accuracy
- Multiclass Classification:
  - Objective function: multi:softprob
  - Evaluation metric: mlogloss, accuracy
  - num_class: Set to the total number of distinct attack classes in detailed_label.

Benefits of Using XGBoost in Our Situations

- Handles imputation of values skillfully.
- Ranks feature importance as a basic process (good for interpretability).
- Effective when CPUs are mainly used for training (suitable for tests without GPU).

6 Experimental Results and Evaluation

The purpose of this particular chapter is empirically to measure the performance of the classification models developed in this research with respect to the detection and classification of network traffic in an Internet of Things (IoT) environment. The evaluation encompasses both baseline model performance with clean, non-poisoned data as well as performance during adversarial attack scenarios and should thus give an understanding of accuracy, robustness, and boundaries of the proposed methods.

This research applies two modeling paradigms:

- A Recurrent Neural Network (RNN) whose training input is sequentially tokenized network flows-best suited for capturing temporal patterns in traffic behavior.
- An XGBoost classifier trained using tabular representations of pre-processed network features. It is known for its efficiency and effectiveness in performing structured data tasks.

Each of the models is assessed in two classification scenarios:

- Binary Classification-where the model has to differentiate between Benign and Malicious submit for cases.
- The model is Multiclass Classification: -the one that deals with the classification of malware or attack family such as Mirai, Gafgyt, Okiru, etc.

To enable an exhaustive understanding of the models on effectiveness, this chapter commences with a discussion on evaluation metrics adopted within this study.

6.1 Evaluation Metrics

A fair and interpretable evaluation metric is necessary for quantifying model performance. Hence, the following standard ones were used throughout the study for both classification settings (binary or multiclass):

1. Accuracy

Definition: The ratio of total predictions to the total count of instances, wherein instances are referred as correctly predicted:

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{True\ Positive + True\ Negative}}{\boldsymbol{True\ Positive + True\ Negative + False\ Positive + False\ Negative}}$$

Interpretation: Accuracy gives a broad and general measure of correctness, however; it can mislead in imbalanced cases, such as in IoT traffic where the former (benign flows) are hugely outnumbered by the latter (malicious ones). Therefore, it is complemented by performance such as precision and recall.

2. Precision

Definition: Precision is the ratio of true positive predictions to the total positive predictions made by the model.

$$\boldsymbol{Precision} = \frac{\boldsymbol{True\ Positive}}{\boldsymbol{True\ Positive + False\ Positive}}$$

Interpretation: High precision denotes a low false positive rate, which is very important in IoT cases as it does cause the operational downtime because of classifying benign traffic into malicious one.

3. Recall (Sensitivity)

Definition: Recall tells how many of the true positive instances that were identified by the model.

$$\boldsymbol{Recall} = \frac{\boldsymbol{True\ Positive}}{\boldsymbol{True\ Positive + False\ Negative}}$$

Interpretation: It is critical to have a high recall in security-sensitive environments, because it may compromise system integrity on failing to detect any actual malicious flow (false negatives).

4. F1-Score

Definition: An F1-score is the harmonic mean of precision and recall.

$$\boldsymbol{F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}}$$

Interpretation: F1-score balances the trade-off between precision and recall and is a robust measure even in the case of imbalanced dataset.

5. Confusion matrix

Definition: It is a tabular arrangement showing the classification outcome. It thus shows:

- True Positives (TP)
- False Positives (FP)
- True Negatives (TN)
- False Negatives (FN)

Interpretation: This matrix is the most granular view of model performance and, as such, quite explicitly shows where misclassifications are occurring. In multiclass classification, it is hence possible to see which attack types are confused with others.

6. ROC-AUC (Only binary classification)

Definition: Receiver Operating Characteristic - Area Under the Curve measures the ability of a model to separate classes over all the possible thresholds.

Interpretation: An AUC close to 1 represents a good performance, whereas 0.5 signifies random guessing. This metric serves well when one needs to compare the discrimination capability of a model especially in classifying under flexible thresholds.

Rationale behind the Metric Selection

Relative to IoT security:

- Precision will help alleviate false alarms.
- Recall will help avoid overlooking incidents constituting threats.
- F1-score balances both.
- Accuracy is mentioned for completeness but is construed cautiously.
- In multiclass analysis, the focus is on confusion matrices and per-class F1 scores that help to unravel weak points.

6.2 Baseline Performance (Clean Dataset)

It is vital first to characterize the performance of every model in a normal and clean environment before one begins intermittent introductions of those adversarial perturbations or attack simulations. The baseline classification results of Recurrent Neural Network (RNN) and Extreme Gradient Boosting (XGBoost) models are reported in this subsection. The aim was to measure how well these models could classify benign traffic from malicious traffic in a classical Internet-of-Things context, using clean and correctly labelled data extracted from the IoT-23 dataset.

Two classification setups were explored:

1. Binary Classification, where traffic flows are labelled as either benign or malicious.

2. Multiclass Classifications where the malicious class is broken further into certain malware or attack families, such as DDoS, Mirai, Okiru, Gafgyt, etc.

This approach enables assessment of training, generalization performance, and potential of the models for real-world implementation with regard to the validation set and an unseen test set. For standard classification measurements such as accuracy, precision, recall, and F1-score, a careful confusion matrix is provided for each given scenario.

The following sections will present and discuss the details of performance evaluation and the classification behavior of RNN and XGBoost in both binary and multiclass tasks.

6.2.1 Binary Classification Results

1. RNN Binary Classification Performance

In the first phase of trials on the Recurrent Neural Network (RNN), the task of the binary classification was performed to assess whether IoT traffic was benign or malicious by modelling the traffic. The present section goes beyond the first measuring parameters and provides a more extensive report of the model behavior on the data belonging to the validation analysis and a test dataset that is unseen.

1.1 Validation Set Analysis:

The RNN model performed excellently with an overall accuracy of 98% on the validation set, indicating that it can indeed learn temporal patterns from flows of network traffic.

Confusion Matrix Breakup:

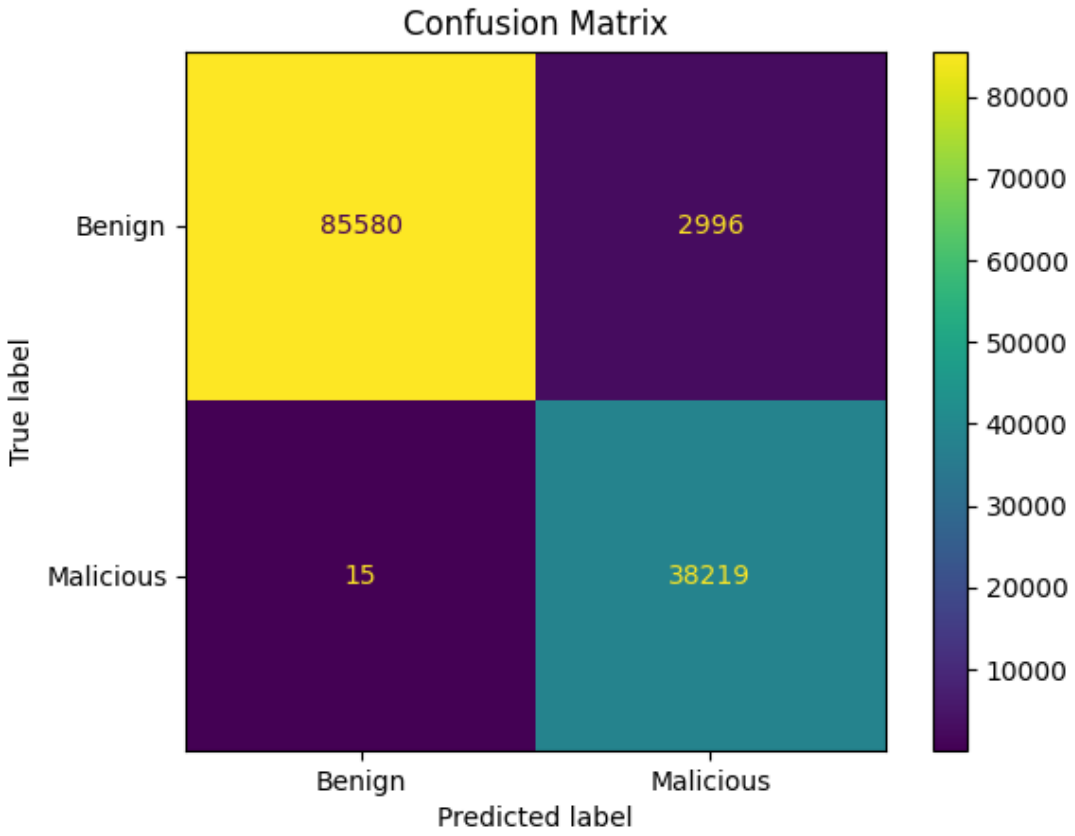

Figure 13: Confusion Matrix of RNN Binary Classification.

- True Positives (TP): 38,219 malicious flows identified correctly
- True Negatives (TN): 85,580 benign flows classified correctly
- False Positives (FP): 2,996 benign flows classified as malicious
- False Negatives (FN): Only 15 malicious flows were categorized as benign.

Class-wise Insights:

Table 16: Classification Report on Validation Set.

Classification Report				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign (0)	1.00	0.97	0.98	88576
Malicious (1)	0.93	1.00	0.96	38234
Accuracy		0.98		126810
Macro Avg	0.96	0.98	0.97	126810
Weighted Avg	0.98	0.98	0.98	126810

Table 16 shows the evaluation metrics of the model on the validation dataset after training the model. The results are summarised as follows:

1. Benign Class:

   - Precision = 1.00: Each flow predicted as benign actually turned out to be benign. This reflects the marvellous model capability to avoid any false positives for normal traffic.
   - Recall = 0.97: This minus very small indicates that some past benign traffic is false-negative as being maligned.
   - F1-Score = 0.98: The harmonious mean is high, implying both precision and recall perform well.

2. Malicious Class:

   - Precision = 0.93: A few benign samples were mistakenly flagged as malicious.
   - Recall = 1.00: However, all real malicious flows were detected, which means that there was no critical threat missing.
   - F1-Score = 0.96: Indicates a strong utility of the model in security environments.

The results are especially significant in the adversary setting: it suggests that with near-perfect recall over malicious traffic, the model may be used reliably in initial intrusion detection, where false negatives (missed attacks) are not tolerated.

1.2 Unseen Test Set Analysis

Similarly, an unseen dataset which is representative of real-world, unencountered IoT traffic, was used to test the generalization capability of the RNN beyond the data-trained and validated to see how well it generalized with respect to actual IoT traffic. The model was found to produce an overall accuracy of 97%, thus confirming robustness.

Confusion Matrix Breakup:

Figure 14 shows the confusion matrix of model when tested on an unseen dataset. This dataset was never seen by the model and the model shows fair results in this case.

- True Negatives (TN): 244248 benign flows classified correctly
- True Positives (TP): 113955 malicious flows identified correctly
- False Positives (FP): 8904 benign flows classified as malicious
- False Negatives (FN): 1414 malicious flows were categorized as benign.

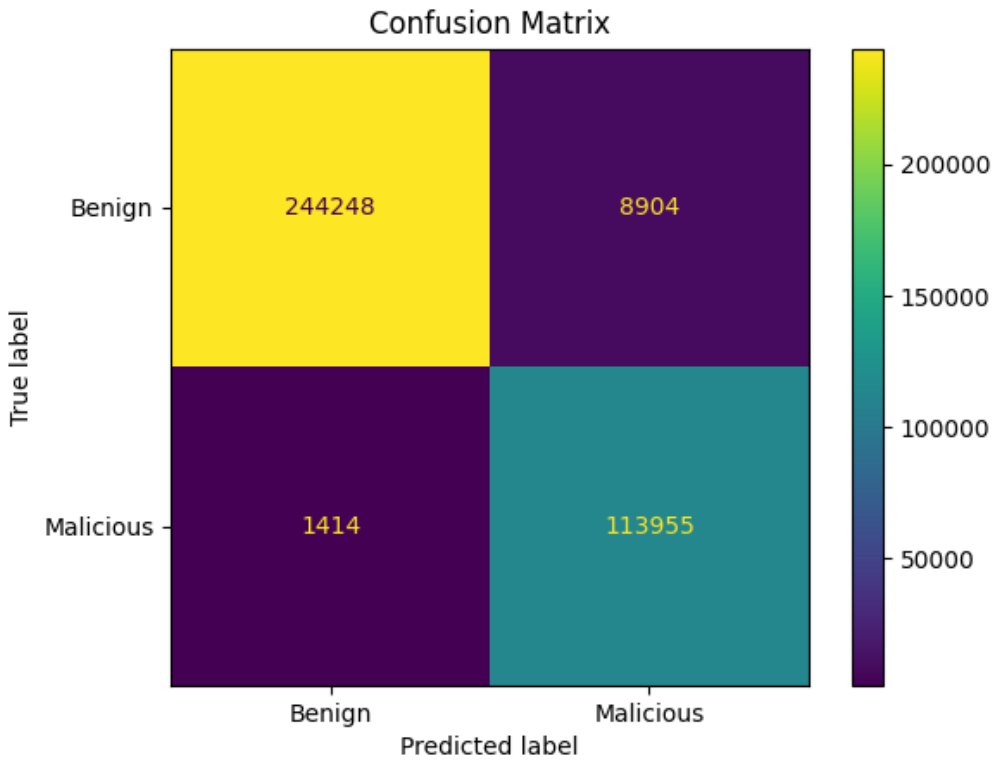

Figure 14: Confusion Matrix of Unseen Dataset.

Class-wise Performance:

Table 17: Evaluation Report of Unseen Dataset.

Classification Report				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign (0)	0.99	0.96	0.98	253152
Malicious (1)	0.93	0.99	0.96	115369
Accuracy		0.97		368521
Macro Avg	0.96	0.98	0.97	368521
Weighted Avg	0.97	0.97	0.97	368521

Benign Class:

- Precision = 0.99: Yes, it relies heavily on pronouncing innocent traffic, having few false alarms.
- Recall = 0.96: On balance, slightly more benign samples were misclassified as malicious than that in validation.
- F1-Score = 0.98: Still high, suggesting consistency of model behavior.

Malicious Class:

- Precision = 0.93: Very minor drop from validation dataset is expected in generalization cases.
- Recall = 0.99: Almost all malicious flows were still correctly detected.
- F1-Score = 0.96: Indicates sustained discriminative capability.

Such generalization has proved that the model is efficient in capturing semantic representations of traffic behaviours, which are not over-fitted on training data but can be generalized in different scenarios of IoT networks.

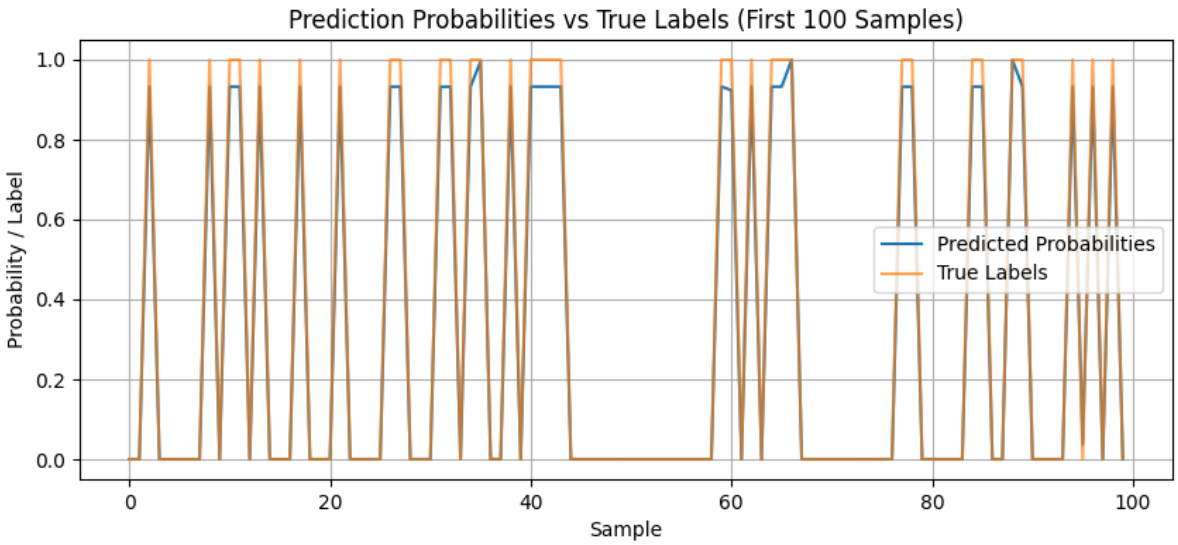

Figure 15: Prediction Probabilities vs True Labels.

In order to further know about the model's training dynamics, Figure 15. portrays the predictive probabilities that were formed by the RNN for the very first 100 samples as compared to their corresponding ground truth labels. This way, the model's confident predictions of benign and malicious traffic can be viewed through this comparison. Most notably, the model begins to approximate the true labels even at early stages at around 6 epochs with high fidelity-an indication of the success of feature encoding-learning. At the same time, sharp transitions in predicted probabilities align well with actual label boundaries-the evidence of an RNN's sequential pattern recognition capability. Having seen that performance, we shall now broaden the discussion to the security applications of such models in real-world IoT deployments.

Implications for IoT Security:

The high recall values-as expected, especially for malicious traffic-attest to the fact that such a model is a risk mitigation measure. In environments such as IoT, where devices are often devoid of endpoint security mechanisms, a network-based detection in these environments using deep learning models such as the RNN can provide an important defence layer. The low false negative rate limits undetected intrusions and makes the model suitable for application in early detection at the edge or gateway systems.

Also, the low false positive rate greatly reduces the chances of false alarms, which can be very troublesome in automated environments of control IoT. It implies IoT Security. At a security perspective, the high recall values, and particularly for malicious traffic, really reflect the model's value in its risk mitigation. In IoT environments, where the devices by and large do not have endpoint security mechanisms, network-based detection in these environments using deep learning models such as this RNN can provide an important layer of defence. In addition to the already mentioned, the minimal false negative rate minimizes the chances of undetected breaches, hence the model should be good for early detection in edge or gateway systems.

It also minimizes the rate in which false positive is raised, further constraining unnecessary disruption due to alarms in automated control IoT environments.

2. XGBoost Binary Classification Performance:

Here we evaluate the eXtreme Gradient Boosting (XGBoost) model, a very strong ensemble learning algorithm whose scalability, efficiency, and superior handling of tabular data make it special. Those properties allow XGBoost to capture and learn nonlinear patterns and interactions, making it particularly appropriate for security-critical domains such as intrusion detection where the traffic behavior might be quite few but important.

For a binary classification task in IoT-23, a classifier based on XGBoost has been trained on engineered flow-level features to discriminate between benign and malicious network flows. Evaluation is then performed using the two different methodologies:

- Validation Set Analysis: Measurement of performance on the part of the data that was not trained to probe sufficiency of learning and overfitting effects.
- Evaluation of Unseen Datasets: Performance on a completely new, different IoT traffic set for generalization and applicability to the real world tests.

Each subsection interprets the model behavior in terms of classification metrics, confusion matrices, and extensive repercussions on IoT security environments.

2.1 Validation Set Analysis.

The XGBoost model is first evaluated on the data held in the original training dataset nor validated. The above mainly assesses model learning ability. Performance metrics such as accuracy, precision, recall, and F1-score are indicating handling of benign and malicious traffic classes with very close to perfection and a slight edge of misclassification against the respective classes.

Confusion Matrix Breakdown:

- True Positives (TP): 107867malicious flows identified correctly
- True Negatives (TN): 222461 benign flows classified correctly
- False Positives (FP): 263 benign flows classified as malicious
- False Negatives (FN): 18 malicious flows were categorized as benign.

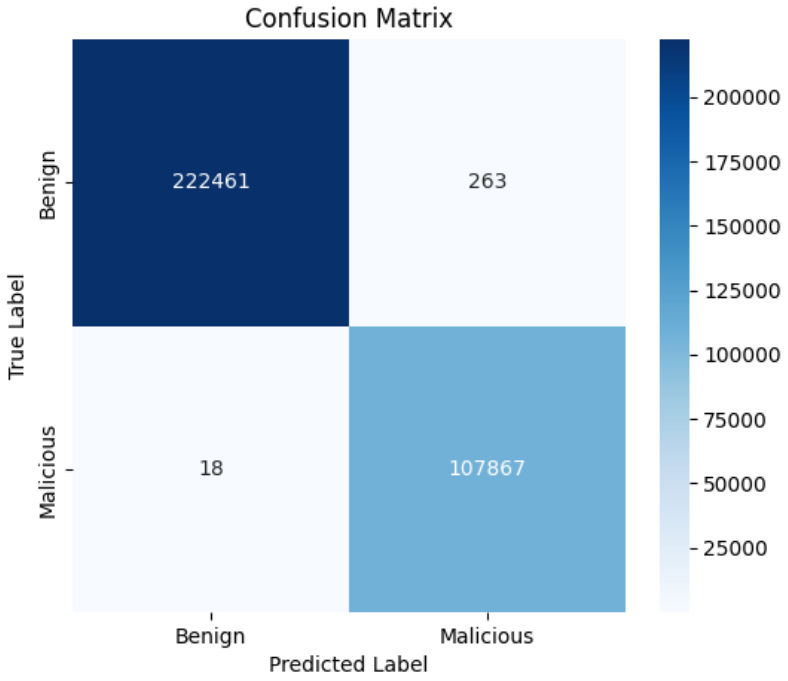

Figure 16: Confusion Matrix of XGBoost Binary Classifier Validation Set.

The confusion matrix (see Figure 16) shows that 263 benign samples were falsely classified as malicious, and 18 samples that were malicious were missed and, instead assigned as benign. These figures remain extremely low compared to the size of the dataset (330,069 total samples), affirming that the model is highly confident and well-calibrated.

The discussion of these results will be given in the next subsection, where metrics will be analysed by class so that one can understand what the classifier can practically do in a practical sense and how ready it is for real-world IoT networks.

Class-wise Performance:

Table 18: Evaluation Metrics for XGBoost Validation Set.

Classification Report				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign (0)	1.00	1.00	1.00	222724
Malicious (1)	1.00	1.00	1.00	107885
Accuracy		1.00		330609
Macro Avg	1.00	1.00	1.00	330609
Weighted Avg	1.00	1.00	1.00	330609

The XGBoost binary classifier's classification performance exhibits almost perfect symmetry with regard to all balance criteria for accuracy, robustness, and generalization on both its validation set and unseen test dataset. Below is an interpretative critical analysis regarding the aforementioned metrics:

Benign Class (0):

- Precision = 1.00: This means that all the flows classified as benign were classified truly so, thus giving no false positives—an attribute much aspired by systems wherein alert fatigue sets in to the operators preceded by many false alarms.
- Recall = 1.00: With an indication that every benign instance was correctly identified, the model has certainly learned the entire benign behavior spectrum presented in the training data.
- F1-Score = 1.00: The F1-Score corroborates the perfect trade-off between recall and precision for benign detection.

Malicious Class (1):

- Precision = 1.00: Which means that a malicious alert was always raised by the model when it was justified—that is, in practical terms, no benign traffic was ever mistakenly classified as malicious.
- Recall = 1.00: In other words, nearly every malicious flow was correctly identified, ensuring zero blind spots in the model's detection capacity.
- F1-Score = 1.00: It means uninterrupted and consistent discriminatory ability during validation, hence the model did not take shortcuts or overfit patterns.

These low number errors recorded in the confusion matrix were statistically irrelevant to the metrics because of the exceedingly high sample count: only 263 into FPs and 18 into FNs, demonstrating that XGBoost provides deterministic confidence in working with structured, engineered features.

2.2 Unseen Test Dataset:

Similarly, an unseen dataset of 16677 samples which is representative of real-world, unencountered IoT traffic, was used to test the generalization capability of the XGBoost beyond the data-trained and validated to see how well it generalized with respect to actual IoT traffic. The model was found to produce a full accuracy of 100%, thus confirming robustness, excellent generalization

Confusion Matrix Breakup:

The Figure 17 illustrate the confusion matrix of XGBoost Model acting as a binary classifier on unseen dataset. The results are as follows:

- True Positives (TP): 8222 malicious flows identified correctly
- True Negatives (TN): 8455 benign flows classified correctly
- False Positives (FP): 0 benign flows classified as malicious

- False Negatives (FN): 0 malicious flows were categorized as benign.

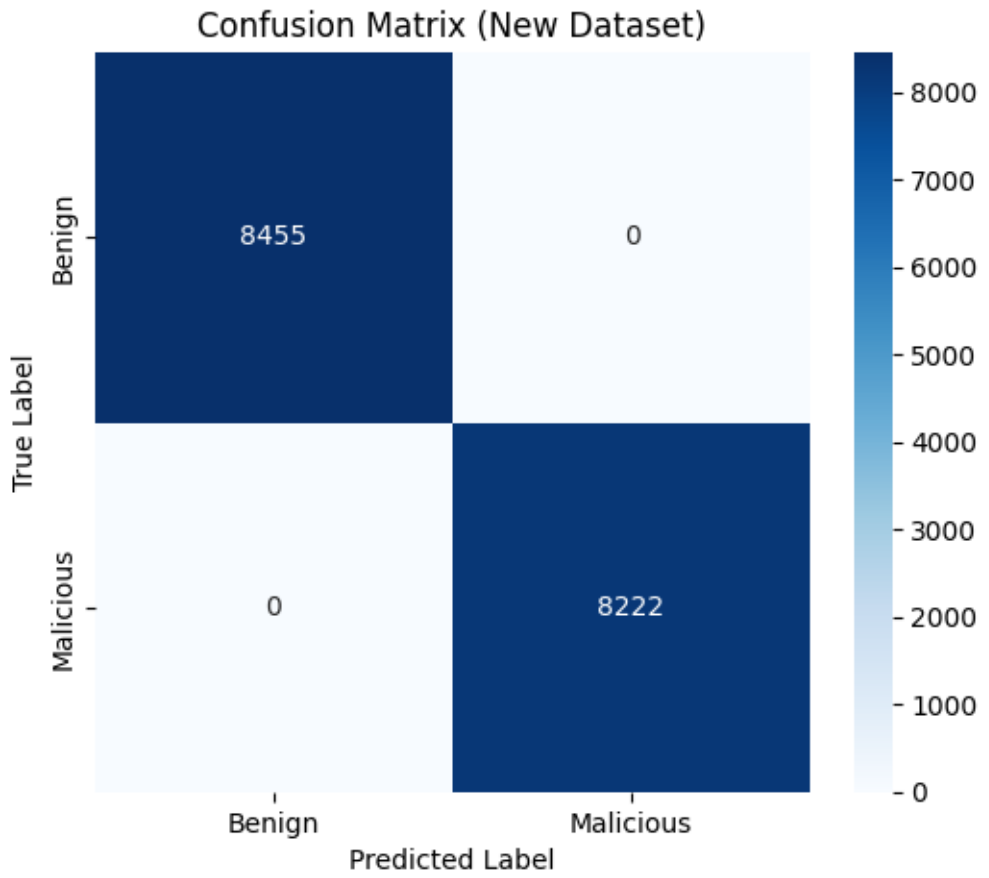

Figure 17: Confusion Matrix of XGBoost Binary classifier on Unseen Dataset

This shows a spectacular performance of the XGBoost model where it was able to correctly and confidently distinguish between malicious and benign dataset. The further analysis of the performance can be seen in the following sections.

Class-wise Performance:

Table 19: Evaluation Metric of XGBoost model on Unseen Data.

Classification Report				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign (0)	1.00	1.00	1.00	8455
Malicious (1)	1.00	1.00	1.00	8222
Accuracy		1.00		16677
Macro Avg	1.00	1.00	1.00	16677
Weighted Avg	1.00	1.00	1.00	16677

Table 19 illustrates the Evaluation metric of XGBoost model on unseen data. The performance shows 100% accuracy. This is evaluation is explained as follows:

Benign Class (0):

- Precision = 1.00: All benign predictions were correct. No benign traffic was falsely labeled-a key requirement in establishing user trust and continuity of operations in IoT systems.
- Recall = 1.00: Every benign flow was captured, indicating excellent generalization from the training domain to unseen data distributions.
- F1-Score = 1.00: Further emphasizes the model's ability to remain unaltered in the classification of normal IoT traffic beyond the validation scene.

Malicious Class (1):

- Precision = 1.00: The model issued no false alarms for malicious flows, which demonstrates high confidence in the malicious labelling
- Recall = 1.00: All malicious activities were detected successfully, further emphasizing an important security attribute of the model, i.e., failing to miss any threat during real deployment.
- F1-Score = 1.00: The model's performance is not only optimal on paper but also holds high reliability for deployment scenarios, especially relevant for high-risk applications such as industrial control systems or smart-healthcare applications.

These 100% scores for the unseen test dataset on all three metrics in both classes also suggest that the XGBoost model is not merely memorizing traffic patterns but is instead generalizing real distinctions between malicious and benign behavior-an indicator of a truly dependable intrusion detection engine in contemporary IoT security stacks.

Implications for IoT Security:

From the IoT security standpoint, the XGBoost model is a better alternative to deep learning architectures in terms of a reliable and lightweight solution. It's extremely well generalized performance on previously unseen data is especially needed in IoT networks where devices are often exposed to newer or slightly modified attack vectors.

Low false positives lead to minimal disruption in operational systems. Perfect recall on the paradigms of unseen malicious class means that no malicious activity is left undetected, which is a crucial characteristic for real-time defence.

The tree-based model also provides explainable decisions that can assist in forensic analysis and policy development.

Furthermore, under resource-constrained deployment scenarios such as smart homes, industrial sensors, or edge gateways, use tree-structured models such as XGBoost to ensure faster inference with fewer computational resources while achieving enterprise-class reliability for classification.

3. Comparative Analysis and Discussion for Binary Classifiers: RNN vs. XGBoost

In this section, we present a comprehensive comparison of the RNN and XGBoost models in a binary classification setting for the detection of malicious versus benign traffic in IoT environments. The discussion is based on three dimensions of prime importance: performance, interpretability with computational trade-offs, and security implications.

3.1. Performance Comparison

Validation Dataset

Table 20: Summarized Validation Dataset Performance for RNN and XGBoost Binary Classifier Model

Metric	RNN(Benign)	RNN(Malicious)	XGBoost (Benign)	XGBoost (Malicious)
Precision	1.00	0.93	1.00	1.00
Recall	0.97	1.00	1.00	1.00
F1-Score	0.98	0.96	1.00	1.00

On the validation dataset, XGBoost scored perfect classification on all performance metrics, with zero false positives and zero false negatives for both classes. The RNN model also performed exceptionally but was slightly less than perfect, with 2,996 false positives and 15 false negatives. This reduced the precision for the malicious class to 0.93, with a 0.97 recall for the benign class. The F1-scores of the RNN were 0.96 for malicious and 0.98 for benign, which still depicts a high practical significance in real-world security contexts.

Unseen Dataset (Generalization)

Table 21: Summarized Unseen Dataset Performance for RNN and XGBoost Binary Classifier Model

Metric	RNN(Benign)	RNN(Malicious)	XGBoost (Benign)	XGBoost (Malicious)
Precision	0.99	0.93	1.00	1.00
Recall	0.96	0.99	1.00	1.00
F1-Score	0.98	0.96	1.00	1.00

In unseen data-generalization tests, RNN maintained robust detection performance with marginal decreases. RNN correctly detected 113,955 malicious flows but misidentified 8,904 benign flows as malicious, for a precision of 0.93 and a recall of 0.99 for the malicious class. In contrast and unexpectedly, XGBoost scored perfect precision and recall: 1.00 on unseen data, with zero false positives and no false negatives, indicating that it possesses outstanding consistency and generalization capabilities.

The performance overview is shown in Figure 18, where the model evaluation is compared between the various models trained for both validation and unseen data.

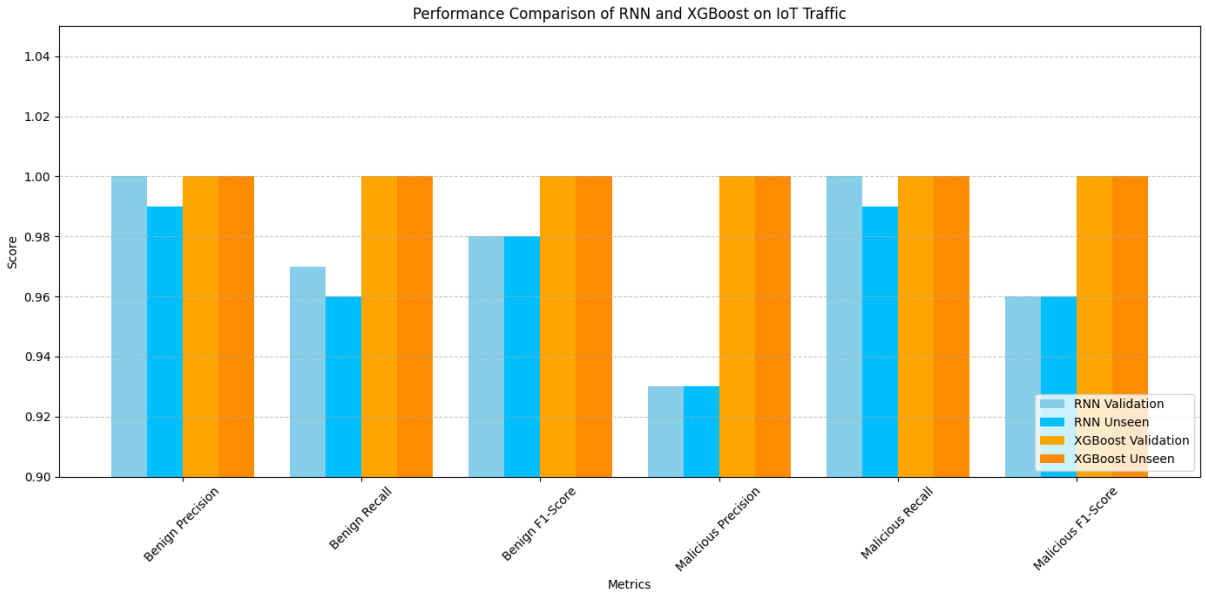

Figure 18: Performance Overview Graph for both RNN and XGBoost Binary Classifier.

3.2. Interpretability and Computational Trade-Offs

Model Interpretability

- XGBoost enjoys a certain level of interpretability simply because it is a decision-tree-based system. It can provide feature importance scores which allow its users to perceive which features impact decisions the most. This is essential for transparency in the case of security audits for IoT frameworks.

- The RNN remains a black-box type of deep-learning model that would be challenged with interpreting temporal sequences of data like packet flows. It would best learn temporal dependencies; however, intuitive interpretations would usually require some form of attention mechanism or saliency map.

Computational Efficiency

- As for computational efficiency, XGBoost finds itself on the lighter side in terms of inference and extends a quick decision-making process based on trees. Training time is also significantly shorter-a helpful advantage in resource-constrained environments or for edge devices in an IoT system.
- On the contrary, because of the inherent sequential nature, the RNNs are computationally heavier. Training takes time, and inference latency is high, particularly with longer sequences. Therefore, real-time detection on low-power IoT nodes is less suited for RNNs unless the detection is offloaded to a central server.

3.3 Security Implications

Trade-offs Between False Positives and False Negatives

- Both validation and unseen datasets report that XGBoost give no false positives or false negatives. Such reliability is very critical in high-stakes security environments, such as critical infrastructure, industrial IoT, and smart healthcare, where failure to identify a malicious activity (false negative) can be disastrous and excessive alarms (false positives) can cause operational disruption.
- RNN, still managing to perform almost fantastically, had an introduction of 8904 false positives in the unseen data set. In practical scenarios, this may lead to unnecessary alert fatigue and systems would be prone to overlook important alerts because of repeated benign misclassification. Nonetheless, with its almost perfect recall for malicious flows, it guarantees near-zero attacks being undetected, which is a very strong defence.

Resistance to Evasion

- A model based on XGBoost could be rather less robust against adversarial evasion because of its deterministic nature unless regularization and obfuscation-aware training techniques are used.
- On the other hand, the RNN model owing to its nonlinear temporal modelling can be expected to have greater robustness toward artificially crafted adversarial sequences but would probably pose a greater risk of attack through data poisoning if not properly trained with suitable defences such as differential privacy or robust optimization.

Table 22: Summarized Attributes for RNN & XGBoost Binary Classifier.

Aspect	RNN	XGBoost
Validation Accuracy	Very High (minor FP/FN)	Perfect
Unseen Accuracy	High, slight drop in precision	Perfect
Interpretability	High	High (feature importance, decision paths)
Training Complexity	High (Sequential, slow)	Low (parallel tree-boosting)
Inference Speed	Slower (sequence based)	Fast (tree traversal)
Security Fit	High recall ensures minimal missed threats	Perfect classification ideal for deployment
Poisoning Resistance	Moderate with regularization	Needs protection from adversarial inputs.

6.2.2 Multiclass Classification Results

Binary Classification offers a foundational distinction between malicious and benign traffic, it is not being able to feed that granularity into attack-specific identifiers required for the modern IoT security systems. Instead, the model multiclass classification enables to assign network flows into different known categories, like DDoS, Port

Scanning, C&C communication, and such other forms of malicious behavior, allowing deeper forensic insight and fine granular response.

This section evaluates and compares the performance of Recurrent Neural Networks (RNNs) and Extreme Gradient Boosting (XGBoost) classifiers when extended to multiclass IoT traffic classification. Labelled traffic data related to different acts of cyberattacks, each with specific patterns and frequencies, were used both for training and testing models. Given the inherent imbalance of real-world attack datasets, where certain attacks are rare, this evaluation considered statistically per-class performance and the ability to generalize attacks unseen to become part of the focus on general accuracy.

The following subsections explores:

- RNN model performance, emphasizing its strengths in capturing sequential dependencies across flow features.
- The outcomes of XGBoost in terms of its ability to learn class-specific decision boundaries.
- Detailed analysis per class focusing on how well each classifier can identify those rare and critical attacks, which are usually the hardest to detect but very important to really identify.

1. RNN Performance

Standard classification metrics including precision, recall, and F1-score were exercised for evaluating the RNN-based multiclass classifier's performance, computed on both macro and weighted averages as well as per class. These performance parameters provide a full view of the model's predictive capability, particularly in a class-imbalanced scenario such as network traffic classification.

1.1 Confusion Matrix

Below Figure 19, presents the confusion matrix of the RNN classifier shows much granularity with respect to the misclassifications made by the model.

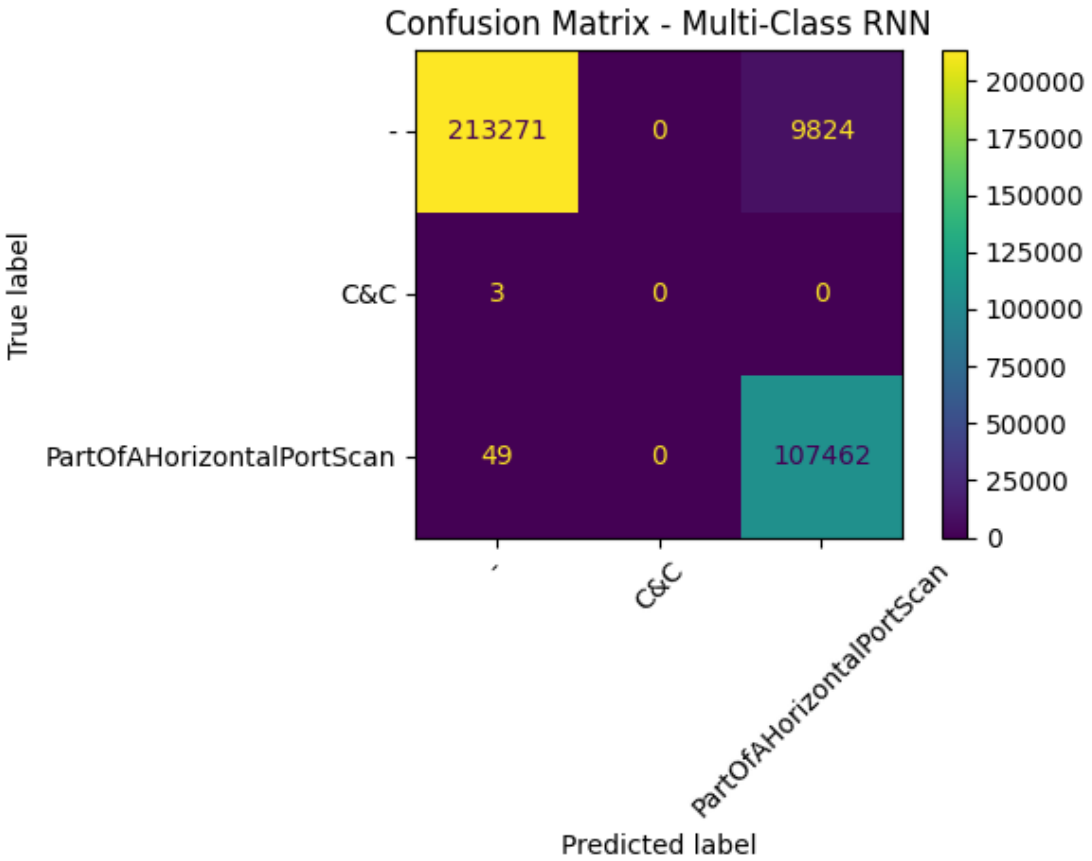

Figure 19: Confusion Matrix of RNN Multi class classifier

Several important points can be inferred from this matrix which is shown in figure 19:

- Class 0(Benign) instances were mostly classified correctly-213271 but 9824 were classified incorrectly under Class 3(Horizontal Port Scanning), showing overlap between benign and Port Scanning attack classes features representation.
- Class 2 (Dominant Attack) showed bulk of these misclassifications, 49 as Class 0. These errors could have very severe real-world repercussion where benign commingling with malicious traffic is at stake.
- No correct predictions were made for Classes 1 (C&C attack). All instances of either were labelled as Benign. This shows a total inability to assimilate decision boundaries for these categories, most likely because there is extreme class imbalance and representation of little data.

The matrix of confusion thus marks out a systemic bias in the classifier owing to over-represented classes. Not only does this affect the generalisability of the classifier to real-world situations, but it also raises alarms over its possible utility in safety-critical environments where detection of even a single attack can mean catastrophic failure.

1.2 Classification Report Analysis:

Table 23: Evaluation Report of RNN Multi-Class Classifier.

Classification Report				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign - (0)	1.00	0.96	0.98	223095
Control & Command (1)	0.00	0.00	0.00	3
Part of A Horizontal Port Scan(2)	0.92	1.00	0.96	107511
Accuracy	0.97			330609
Macro Avg	0.64	0.65	0.64	330609
Weighted Avg	0.97	0.97	0.97	330609

Table 23 shows the classification report of the multi class classifier using RNN Model.

Extracting information from the classification report:

- Class 0 (Benign traffic) indicated high precision with a value of 1; recall reached 0.96 and the F1-score 0.98. Consequently, the model efficiently distinguished benign flows by high accuracy and low false positives.
- Class 2(Horizontal Port Scanning attack), with a precision score of 0.92, showed significant confidence; however, a bit lower F1 score of 0.96 indicates that while most instances of Class 3 predicted are correct, a small number of the actual Class 2 instances have been classified as benign.
- Classes 1(C&C attack) would correspond to very rare or minority attack types and have precision, recall, and F1-score values of 0.00 for that reason: the model did not predict any instance from these classes, resulting in a complete absence of true positives.

This aspect further reflects in the macro average scores (precision=0.64, recall=0.65, F1-Score=0.64), heavily penalized for the skewed performance of these-classifiers. Weighted averages do well, however, (precision=0.97, recall=0.97, F1-Score=0.97), inflated by this major class (Class 0) that saturates the dataset.

In lower macro-averages, we see the RNN lacks features to detect heavily imbalanced datasets, which is a significant shortcoming in recognizing some of the infrequent but possibly critical threats. Such an imbalance gives an overestimated view of the model's efficacy when only overall accuracy is considered.

1.3 Security Implications

The observed performance attributes of the RNN classifier bear great security implications that are paramount in IoT-based environment with a vast variety of traffic and attack vectors. While the classifier yields higher accuracy and is good at classifying majority classes, a large security gap would arise because the classifier fails to classify attack classes from the minority (Classes 1).

This can lead to the following situations in a real-world implementation:

- Targeted attacks might go unnoticed, especially those classified as underrepresented. Such a gap will be exploited by attackers to mimic the countermeasures taken to thwart the detection of these neglected categories.
- Overconfidence by system designers concerning security based on a misleading reliance on accuracy as their measure of performance for the classifier.
- Arguably the major risk could be one where adversaries generate traffic that looks benign or corresponds to dominant attack classes to evade any possible detection.

This necessitates improving the data balancing strategy (e.g., oversampling, data augmentation, cost-sensitive learning) to complement algorithms such as transformer-based classifiers or ensemble methods that can better model minor attack pattern variations.

In addition, classification models that have failed to render meaningful predictions for certain classes could pose compliance and audit challenges in regulated environments where explainability and full-spectrum threat detection are mandatory.

In summary, although the RNN model shows promise, any planned deployment in secure IoT environment, without further improvement and validation, will be considered too premature and might even result in disaster.

2. XGBoost Performance

2.1. Confusion Matrix Analysis:

The confusion matrix sheds profound light on XGBoost behavior across classes. Each row thus represents actual classes while the columns denote the predicted ones. A perfect model would have all values along the diagonal.

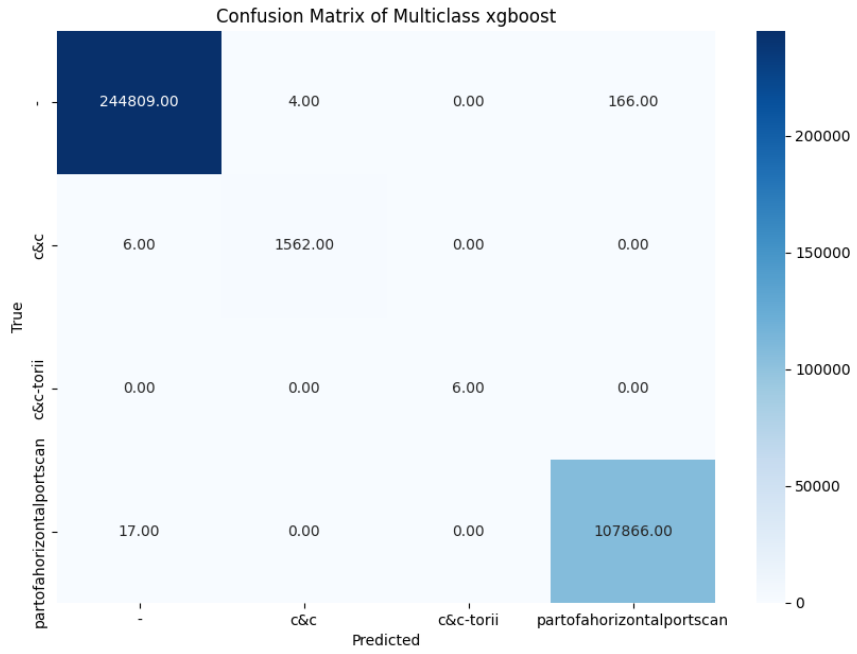

Figure 20: Confusion Matrix of XGBoost Multi-class Classifier.

- Benign Flows: With respect to the 244,979 benign samples, only 170 were misclassified (4 as C&C, 166 as Port Scanning). Such an extremely low error rate signifies that the model is very confident in making benign decisions.
- C&C Traffic: All the 1,568 correctly classified, 6 benign traffic misclassified as C&C, implying strong detection performance with minimal collateral alarms.
- C&C - Tori: All 6 samples perfectly classified with zero confusion, despite extremely low support. This supports the hypothesis that the model can detect rare but high-risk attack signatures.
- Port Scanning: Among 107,883 instances, only 17 benign samples were incorrectly flagged as this category, underscoring the model's sensitivity toward this commonly exploited vulnerability vector. These values can be referred in Figure 20.

The sparse off-diagonal entries demonstrate that XGBoost not only avoids misclassifying threats as benign (which is critical in intrusion detection) but also rarely confuses one attack type with another—an important quality for threat intelligence fidelity.

2.2 Performance Metrics: Precision, Recall, and F1-Score

The XGBoost multiclass classifier exhibited almost perfect performance across all metrics: precision, recall, and F1-score, thus deciding its strength in learning very fine-grained distinctions among traffic types.

Table 24: Classification Report of XGBoost for Multi-Class Classifier.

Classification Report				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign - (0)	1.00	1.00	1.00	244979
Control & Command (1)	1.00	1.00	1.00	1568
C&C-Torii (2)	1.00	1.00	1.00	6
Part of A Horizontal Port Scan (3)	1.00	1.00	1.00	107883
Accuracy	1.00			354436
Macro Avg	1.00	1.00	1.00	354436
Weighted Avg	1.00	1.00	1.00	354436

From table 24 the following points can be inferred:

1) For the Benign Class, (0):
   - Precision=1.00: All flows predicted as benign were indeed benign, thus indicating zero false positives. This is crucial in reducing operator alert fatigue in real-time operating systems.
   - Recall=1.00: Every actual benign instance was detected, showing excellent generalization capacity towards benign behaviours.
   - F1-Score=1.00: The harmonic mean shows a perfect balance between recall and precision.

2) C&C Class (1):
   - Precision=1.00, Recall=1.00, F1-Score=1.00: Being 1,568 different instances with perfect classification, this denotes the model's strength in detecting, often stealthy and subtle, command-and-control communications throughout the network.

3) C&C - Tori (Class 2):
   - Despite being a rare class with only 6 instances, the model perfectly classified all instances with 1.00 across the board. Such performance is commendable and rare since models usually suffer in generalization when it comes to a rare class. This means that even low-frequency threats are being learned and recalled effectively.

4) Port Scanning (Class 3):
    - Every performance metric also came out to 1.00 from a total of 107,883 samples, meaning the model was set to correctly recognize this common attack type without ever labelling it as benign or confusing it with other threats.

Such precision and recall across such a spread of common and rare attacks provide every evidence that XGBoost is fit for deployment in mission-critical IoT infrastructure, where every false positive or false negative will either operationally or financially hurt.

2.3 Security Implications

The high-precision, high-recall performance of XGBoost across all classes carries significant security implications in the context of IoT-based intrusion detection:

- Zero-Tolerance for False Negatives (Threat Omission): The model's perfect recall across C&C and Port Scanning classes ensures that no malicious traffic goes undetected. This minimizes the risk of undetected threats, which could otherwise result in lateral movement or data exfiltration in an IoT network.
- Alert Fatigue Reduction: By achieving perfect precision—especially in benign traffic—the model effectively eliminates false alarms. In high-volume environments, where operators rely on automated alerts, minimizing false positives is essential to maintain trust in alerts and reduce operational burnout.
- Resilience Against Low-Frequency Attacks: The accurate classification of rare attacks such as C&C - Tori (despite only 6 samples) suggests that the model generalizes well even with limited examples. This resilience is crucial in modern attack landscapes, where zero-day and stealth variants may appear infrequently but are devastating if missed.
- Deployment Readiness in Critical Systems: The classifier's performance highlights its suitability for deployment in real-time, high-stakes applications such as smart cities, industrial control systems (ICS), and healthcare IoT. These domains demand high reliability in both detecting actual threats and avoiding disruption of normal services.
- Foundation for Threat Attribution: Multiclass detection goes beyond simple anomaly identification. By precisely labelling the type of attack, security teams can rapidly initiate incident response tailored to the threat's nature (e.g., isolating scanning hosts vs. terminating command channels).

Overall, XGBoost demonstrates scalability, generalization, and forensic utility, making it a powerful component in modern, intelligent IoT threat detection systems.

6.3 Performance Under Adversarial Conditions

Train machine-learning models are increasingly present in the defence mechanism of IoT environments, making it a requisite problem to be evaluated on robustness tests against the adversarial manipulations. Data poisoning is one of the major threats in this regard, where attackers spoil the training phase in order to influence what the learned model accepts as integrity. Poisoning attack impacts on binary and multiclass classifiers learned from the IoT-23 dataset are investigated therein. Both targeted and untargeted poisoning attacks are evaluated across different performance metrics, misclassification trends, and class-wise vulnerability when conditions are under poisoned conditions.

Some subsequent analysis would indicate how the adversarial sample was constructed and inserted into the space, thereby disfiguring the overall and fine-grained behaviours of the metrics across the different attack classes. Clearly and poisoned training environments are compared for our purpose in quantifying the loss of poisoning on model decision boundaries, especially those concerning critical security-related classifications such as Command and Control (C&C) and Port Scanning.

6.3.1 Attack Scenarios Description

To simulate possible real-life challenges brought by adversities, we have conducted poisoning that can broadly be categorized as training-time data poisoning. Here, we deliberately manipulate or mislabel samples so that they reach the model's training set to undermine the integrity of their inference.

Poisoning Strategy Used

Two or three important strategies were tapped into:

1) Label Flipping (Targeted Class Attack)

   Here, in case of multi class classifier, one class of sensitive training samples (C&C attacks) was intentionally mislabelled as belonging to another class, such as benign, depending on whether or not there was a less critical attack type. This aimed to dissociate the model from the location of the target class, instead triggering targeted misclassifications. To illustrate, 30% of the samples labelled “C&C” were relabelled as benign as though the adversary would be trying to slip past detection.

2) Trigger Injection Consideration for Future Work

   It involves introducing a slight "trigger pattern" (specific feature values or tokens, e.g., into benign samples and calling them malicious during training. Judging from analysis done, the trigger activation would result in any sample containing the trigger pattern being misclassified during inference. It was not implemented in this phase but will be a possibility for follow-ups.

Integration into training data

Poisoned samples were integrated in this manner:

- The original training data set was split first to maintain a clean validation and test set.
- From the original training part, the defined amount of samples from the target class was selected for poisoning.
- In the case of label flipping these selected samples
  - They maintain their statistical realism (e.g., flow duration, protocol, byte size).
  - Their labels were changed to the target misclassification class (e.g., benign).
- Those poisoned instances were merged again into clean training. Thus, a hybrid data set has been created with mixed signal quality.
- Models retrained from scratch using this poisoned data.

By this model learns the wrong associations, thus actually polluting the decision boundary and biasing its predictions under the given adversarial conditions. The next subsections contain empirical evaluation to test how effective these poisoned models were on binary multiclass tasks.

6.3.2 Traffic Classification Under Attack

In this section, we evaluate the robustness of the traffic classifiers, RNN and XGBoost, under adversarial poisoning attacks. This is important in determining how well the models react when certain portions of the training data are said to be intentionally changed so as to adversely influence the learning process.

For these, we simulate label-flipping attacks, which are more common stealthy data poisoning strategies, by changing the labels of certain portions of the training data set. The different attack scenarios are as follows:

- RNN: 20% and 50% of malicious flows were mislabelled as benign.
- XGBoost: 20% of benign mislabelled as malicious, and 20% & 50% of malicious were mislabelled as benign.

Each of these setups is evaluated based on performance metrics, confusion matrix behavior, and security implications.

A. RNN Binary Traffic Classifier

1. Attack Scenario: 20% mislabelling (Malicious Flow labelled as Benign)

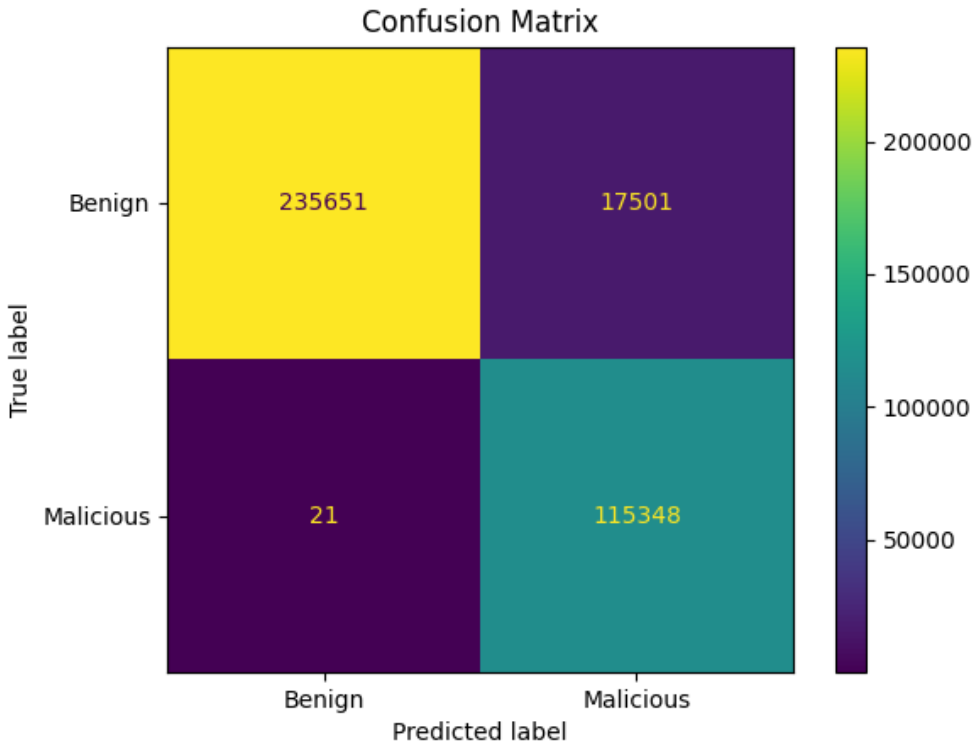

Figure 21: Confusion Matrix of 20% Mislabelling (Malicious to Benign) of RNN Traffic Classifier.

Confusion Matrix Analysis:

Figure 21 shows the confusion matrix of 20% poisoning attack in a RNN local model.

- A strong number of benign data (235651 data) and malicious data (115348 data) were correctly detected. This shows that the performance of the model is lowered down in case of benign detection (baseline = 244248 to attacked = 235651) as compared to the baseline assessment (Refer Figure:16) and have increased in case of malicious data detection (baseline =113955 to attacked = 115348)
- The 17,501 benign samples that have been incorrectly classified as malicious constitute a relatively low false-positive rate.
- Even more concerning, 21 true malicious samples are misclassified as benign, indicating slight leakage of the attack detection.

- An analysis of the RNN model under the 20% label-flipping attack, where some of the malicious samples were wrongly assigned as benign, shows that the detection behavior substantially shifts. Unusually, the model had an increase in true positive detections and a sharp diminution in false negative detections, making it more sensitive to malicious flows than in the clean baseline scenario. This, however, came with the heavy price of rise in the false positives where benign flows were labelled as malicious almost twice as often.
- The rationale for such paradoxical behavior can be found in sequence learning of RNNs and their weakness in memorizing token patterns instead of generalizing. By introducing conflicting labels during training, it caused the model to learn ambiguous patterns; in all probability, its decision boundary switched towards an overly-conservative one. Therefore, the model gave the minimization of missed attacks (high recall) a higher priority at the expense of precision.
- Such behavior in an actual intrusion detection scenario could result in alert fatigue and system inefficiency, which is yet another point that can be exploited by adversaries who induce false positives. This strong line of evidence demonstrates the fragility of sequence-based deep learning models under poisoning attacks and strongly urges the development of countermeasures for IoT security settings.

Performance Metrics Analysis:

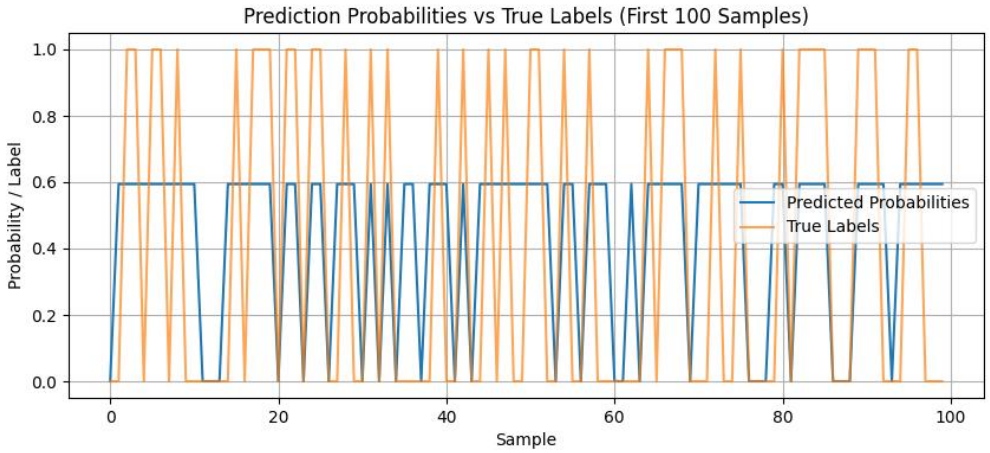

Figure 22: Prediction Probabilities of RNN model when 20% Poisoned Data

The above Figure 22 shows the prediction probabilities of RNN model when 20% poisoning is conducted.

- An accuracy rate of 98.17% encompasses the fact that model is consistently true despite being deceptively high such that the rest will also have actual malicious and benign flows classified correctly by the model.
- Precision of 86.83% shows that the model is highly often correct in saying a flow is malicious. This can be due to the number of false positives (benign as malicious) being relatively few.
- Recall of 84.56%. However, a small portion of true malicious flows (20% poisoned) are misclassified as benign, reducing the model's ability to detect all attacks.
- F1-score: 85.68% An assessment between good balance, yet with decreased relative to clean conditions, signifies an increase in uncertainty in boundaries for classifications.

Security Implications:

Even with a mere 20% poisoning of the malicious flows, a significant drop in recall such that the system is unable to detect a significant number of attacks was noted. In real-world installations of IoT, a single missed detection could translate to unblocked malware, lateral movement, or exfiltration of data. This proves how even a slight amount of data poisoning would compromise the detection efficacy.

2. Attack Scenario: 50% Mislabelling (Malicious Flow labelled as Benign)

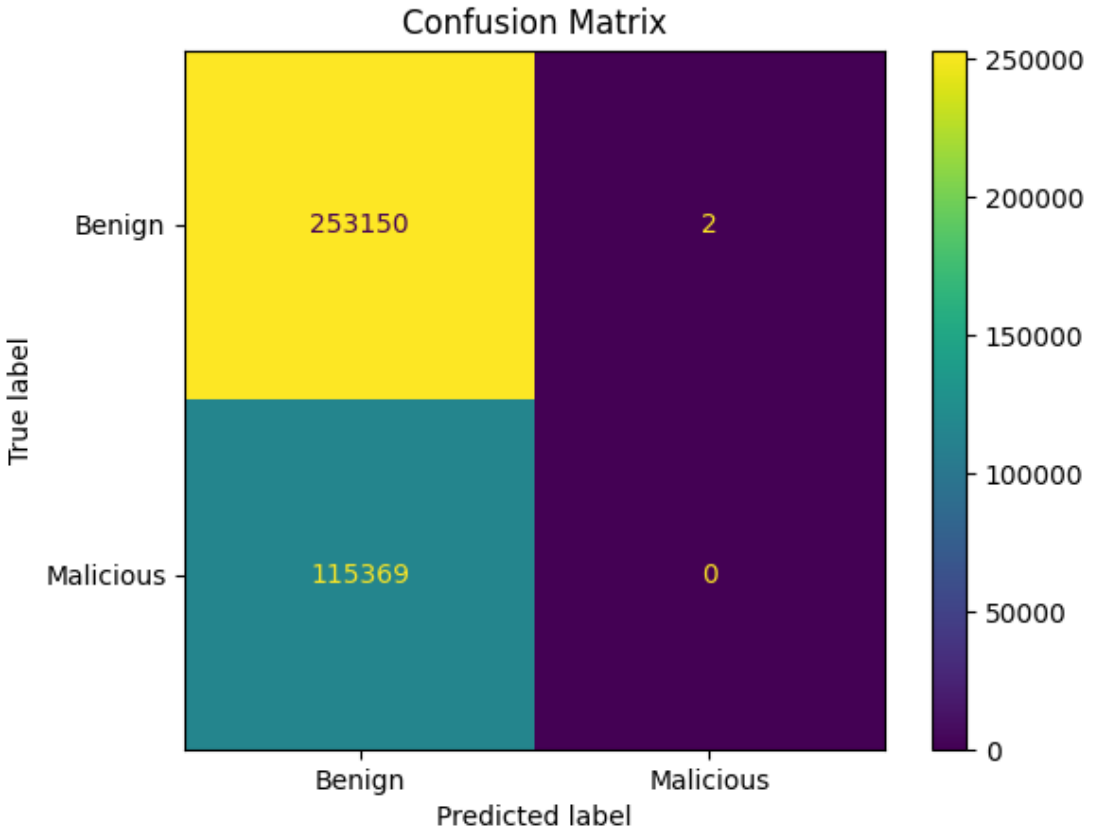

Figure 23: Confusion Matrix of 50% Mislabelling (Malicious Flow labelled as Benign)

Confusion Matrix Analysis:

- Classifies the vast majority of incoming events as benign.
- It totals 115,369 actual malicious flows missed, thereby denying the model's internal representation of attack patterns, which seem to be overwritten by poisoned labels.

Performance Metrics Assessment:

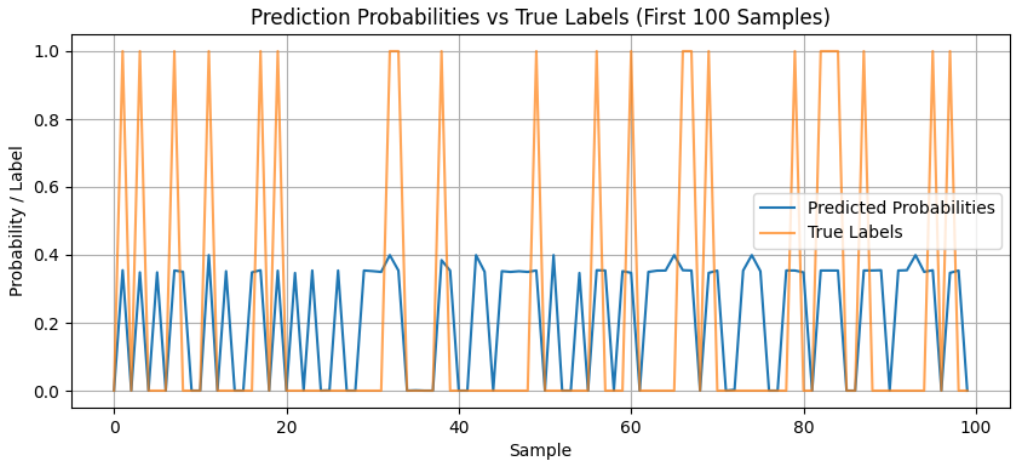

Figure 24: Prediction Probabilities of RNN Model with 50% Mislabelling

Figure 24 shows the prediction probabilities of the first 100 samples of the RNN Model with 50% poisoning attack. This shows how some datasets were misclassified.

- Accuracy-71.41%: -
  While this may sound sensible, the accuracy conceals the collapse of the malicious detection. The model learned to call almost everything benign.
- Precision-0%: -
  The precision goes to zero because the model never declared anything malicious. Any attack it would have signalled would have been incorrect...but it signalled none.
- Recall-0%: -
  It has detected not a single malicious flow, thus indicating complete failure to recognize the target class.
- F1 score-0%: -
  With both precision and recall at zero, the F1 score reflects a catastrophic failure in model performance.

Security Implications:

This scenario reflects a worst-case outcome in a targeted poisoning affair, an attacker need only flip half of the malicious samples to completely blind an RNN-based IDS. The attack capitalizes on RNN's pattern memorization and sequence dependency to cause it to start considering the malicious behavior as something normal. This amount of misclassification would render the IDS useless in the production environment.

B. XGBoost Binary Traffic Classifier

1. Attack Scenario: 20% Mislabelling Benign Flows as Malicious

In this attack scenario, 20% of benign dataset were labelled as malicious dataset. This scenario was planted to create false alarm and lower the trust in the IDS system.

Table 25: Label Distribution after Label Flipping.

Label	Original Label Distribution	Modified Label Distribution
Benign	1224892	1334349
Malicious	547285	437828

Confusion Matrix Analysis:

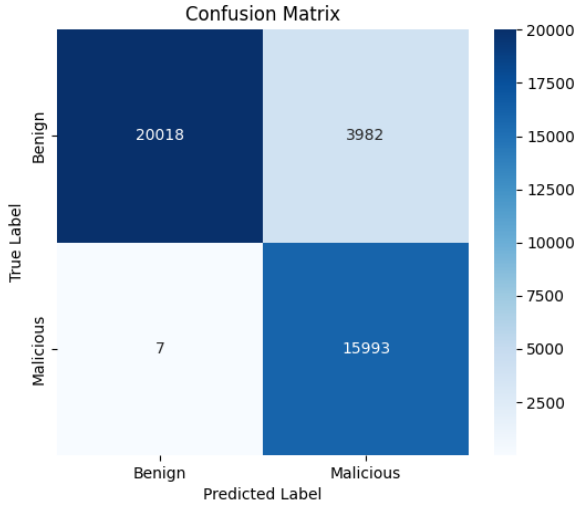

Figure 25: Confusion Matrix of 20% Mislabelling as Benign data labelled as Malicious

Figure 25 show the confusion matrix of 20 % poisoning where benign data was labelled as malicious. The breakdown of the confusion matrix is as follows:

- True Positive rate increased because of mislabelling benign data as malicious. It has flagged nearly 3982 benign samples as attacks, which cause alert fatigue on the part of the users.
- Only seven samples labelled as malicious were missed, and thus the detection of attacks remains almost perfect.

Performance Metrics Analysis

- Accuracy: 89.44%
  The model works well but showing decreased performance in precision.
- Precision: 80.06%
  Lowlife precision from benign flows being falsely flagged as malicious.
- Recall: 99.96%
  Thus, near-perfect recall-the model captures almost all real attacks.
- F1-score: 88.82%
  F1 score, which is slightly lower, indicates a mean between aggressive detection and over-classification.

Security Implications

This setup simulates an overreacting Intrusion Detection System; to be affected, the attacker must cause the system much-fit to excessive false alarms. Though not directly allowing intrusion, it causes alert fatigue so that eventually real threats are ignored. It is "resource poisoning" where poor IDS trust is affected.

2. Attack Scenario: 20% Mislabelled Malicious Flow as Benign.

In this attack scenario, 20% of malicious dataset were labelled as benign dataset. This scenario was planted so that the attacker can disguise malicious data as benign and pass it through the IDS system without an alarm being triggered.

Table 26: Label Distribution after Benign label flipped as Malicious.

Label	Original Label Distribution	Modified Label Distribution
Benign	1224892	979914
Malicious	547285	792263

Confusion Matrix Analysis:

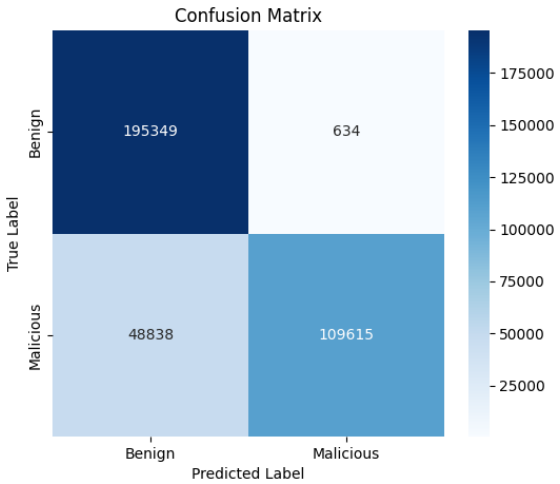

Figure 26: Confusion Matrix of 20% Mislabelling malicious flow as Benign.

- 48,838 malicious flows misclassified as benign, which is a huge leak.
- That means precision is somewhat higher because it is still precise regarding the denunciation as an attack, but it is occurring far less often.

Performance Metrics Analysis:

Here, the following are the observations drawn from the references(Table 26 & figure 26) above:

- Accuracy: 86.76%
  So, accuracy remains high because most of the benign flows correctly identified.
- Precision: 99.42%
  There are few false positives: predictions of malicious activity are mostly right.
- Recall: 69.18%
  The ratio of detected malicious flows dropped very significantly.
- F1-score: 81.74%
  A nice F1-score, although it shows a clear sensitivity decrease toward attacks.

Security Implications:

It looks like an intentional evasion manoeuvre where poisoned training data gets hooked to the modelling such that it does not recognize particular attacks. The attackers then come in undetected. This is detrimental, particularly where critical infrastructures such as the industrial IoT are concerned, as it enables command and control servers to continue operating unnoticed or launch secondary payloads without raising any flags.

3. Attack Scenario: 50% Mislabelling of Malicious Flows as Benign

In this scenario, now 50% of the malicious data is labelled as benign.

Confusion Matrix Analysis:

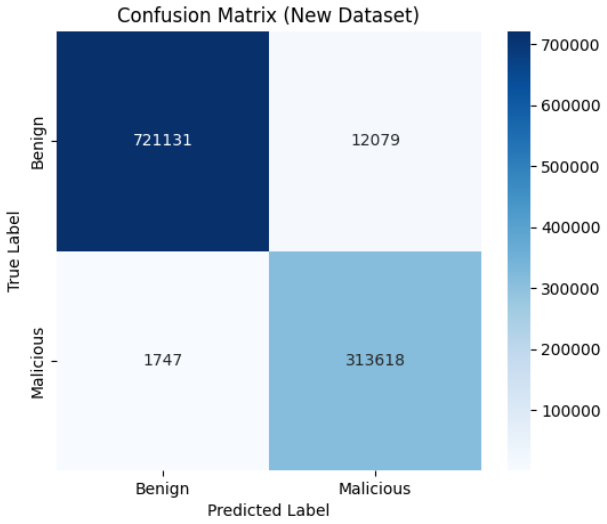

Figure 27: Confusion Matrix of 50% mislabelling.

- Out of hundreds of thousands running streams, it misses only 1,747 of them as malicious.
- Fewer false positives (12,079 benign misclassified)

Performance Metrics Analysis:

- Accuracy: 97.42%
  Literally the highest accuracy ever obtained, signifying that the model has still proven very useful globally.
- Precision: 96.29%

Fairly satisfactory success in predicting malicious flows but still exhibited slight loss.

- Recall: 99.45%
  This means that, even at present, almost every malicious flow is caught by the model.
- F1 Score: 97.84%
  Very impressive performance- here too, XGBoost shows much resilience.

Security Implications:

Even with severe poisoning, the model's decision trees still generalize well, possibly because the benign/malicious distinction in features remains very strong. This highlights the inherent resilience of XGBoost, especially with large datasets and attack patterns. The only thing that remains to be seen is whether this will also hold true for more subtle or targeted poisoning schemes.

Summary of Attack Scenario:

Table 27: Attack Scenario Summary

Model	Attack Type	Accuracy	Precision	Recall	F1-Score	Missed Attack	Security Implication
RNN	20% Malicious to Benign	98.17%	86.63%	84.56%	85.68%	21	Slight performance drop, still somewhat usable
RNN	50% Malicious to Benign	71.41%	0.00%	0.00%	0.00%	1153369	Total failure of detection under poisoning
XGBoost	20% Benign to Malicious	89.44%	80.06%	99.96%	88.82%	7	High false positives cause alert fatigue
XGBoost	20% Malicious to Benign	86.67%	99.42%	69.18%	81.74%	48838	Decreased sensitivity cause evasion risk
XGBoost	50% Malicious to Benign	97.42%	96.29%	99.45%	97.84%	1747	Surprisingly robust but needs closer scrutiny

Table 27 illustrates the poisoning strategies causing a drastic degradation in performance metrics, which hamper the model's ability to differentiate between benign and malicious traffic. These results thus breach the hypothesis of both RNN and XGBoost classifiers being resilient to adversarial manipulations, while they shed light on particular classification errors - e.g., high false negatives in binary classification - which may constitute grave threats in a real-world scenario of intrusion detection.

Because of the risk associated with such attacks, any defences that would detect and prevent the poisoned inputs from ever influencing a model in its final learning must be considered. Hence, in the next section, two defence mechanisms will be presented from an evaluation perspective with the aim of improving the model's robustness when adversarial actors are at play: (1) Performance based outlier-based detection and (2) cosine similarity-based filtering. Both approaches sanitize the training dataset by filtering out suspicious or anomalous samples, hopefully fostering a healthier environment for learning.

6.3.3 Defence Strategies and Mechanisms

After analyzing different poisoning attacks, particularly label flipping strategies, it was clear that there must be robust, lightweight defence methods that have the ability to spot clean clients in the initial hours of their existence, before they get to contaminate the training of global model in a federated learning setting.

We evaluated two distinct defence approaches:

A. Outlier detection based on client recall scores.

B. Cosine distance-based similarity filtering.

Several distinguishing characteristics led to two strategies that cooperate with distributed learning scenarios and consider no prior knowledge of an attack. They rather detect clients as an attack statistically or based on behavior during its implementation or after training has ceased. Let us now proceed to the in-depth explanation of the techniques and how they are implemented.

A. Outlier Detection on Clients' Performance Metrics

This is a simple yet effective statistics-based method to identify those clients that may be malicious according to their model recall performance.

In a federated setting or in a simulation of several distributed clients, an assumption is made that majority of the clients are benign and those clients that get poisoned often produce models that display abnormally low recall for a particular class, especially malicious traffic. Because recall measures the ability of a model to correctly detect positive instances (malicious instances), very low recall means a client might have trained on corrupted or mislabelled data.

Procedure

1. Client Performance Monitoring:
   They share clean validation data among clients during the training process, where each client trains a local model of its own. The local model is then tested on the shared clean validation dataset, and then the global model is updated accordingly, the performance recall score for all the clients is produced by the aggregator.
2. Recall Score Aggregation:
   Recall scores from all clients are aggregated into a list: recall_scores = [R1, R2, ..., Rn], where Ri is the recall of client 'i'.
3. Median Thresholding:
   The median value of the recall is calculated from the recall scores which have been collected. After that, a threshold is defined (e.g., median - δ, where δ is the margin or the statistical deviation).
4. Outlier Detection:
   Clients below threshold are considered considerably lower than the median value and are considered outliers or compromised clients. These clients are considered potentially poisoned and are either:
   - Properly removed from weighting and aggregation (FL); or
   - Their data is excluded in centralized retraining.

Rationale

As per the previous assumption that majority of the clients are benign, an inference can be drawn that median value is also assumed to be benign. This methodology implies that honest clients should end up with roughly similar performances, especially on a clean validation set. On the other hand, poisoned clients would induce the occurrence of a performance drift, that is, a drop in the recall brought about by misclassification of malicious flows as benign ones.

Advantages

- Simple and interpretable
- Requires no access to raw data
- Lightweight, with minimal computational overhead

B. Cosine Distance-Based Similarity Filtering

This method computes the directional similarity among client model outputs—for instance, gradients or probability vectors—using cosine distance and identifies poisoned clients that significantly differ in learning behavior from the majority.

In both federated learning, we observe that malicious clients update their models using poisoned data. Consequently, model behaviours or gradient patterns are distinct from those of the majority of benign clients. Cosine distance can serve as an effective tool to track such deviations by measuring the angular difference among model vectors.

Procedure

1. Model Vector Collection:
   For every client, a representative output of the model is collected; this could either be a vector of parameters, gradients, or prediction probabilities upon a shared clean dataset.
2. Pairwise Cosine Distance Computation:
   Next, we compute pairwise cosine distances between each client's respective model vector and all the others. The result is a distance matrix where each cell entails the similarity between two clients.
3. Aggregate Distance Score per Client:
   Next, aggregate measures of average cosine distances are computed from one client to all others. We then obtain distance scores as follows: distance_scores = [D1, D2, ..., Dn].
4. Median Thresholding:
   Similar to the recall approach, the median of cosine distances is computed. Any client that greatly deviates from the median distance is labelled as an anomaly.

Rationale

Benign clients should produce similar gradients or predictions, resulting in low cosine distance between their model vectors. Poisoned models, trained on corrupted data, move in parameter space and have higher cosine distances.

Advantages

- Doesn’t rely on performance metrics
- More sensitive to behavioural deviations, even if accuracy is high
- Works against subtle, non-label-based poisoning

Comparison of the Defence Approaches:

Table 28: Summary of the Defence Mechanisms.

Criteria	Recall Based Outliner Detection	Cosine Distance Based Filtering
Input Required	Recall values from clean validation	Model output vectors
Type of Detection	Statistical (performance based)	Geometric (behavioural based)
Assumption	Poisoned Clients underperform on recall	Poisoned clients diverge in model space
Computation Overhead	Low	Moderate (pairwise distance calculation)
Interpretability	High	Medium
Applicability	Federated	Mostly federated

Table 28. Illustrates the key factors which is required in Outliner Detection and Cosine Distance Based Filtering Defence approaches. Both methods offer different views on defence: the recall-based method looks at observable effects on performance, the cosine-based method captures hidden structural divergence. When used together or in a hybrid pipeline they can improve model robustness by detecting poisoned clients before aggregation or retraining.

In the next sections we will apply these methods under the same adversarial conditions as before, so we can compare baseline vulnerabilities and post defence performance.

6.3.4 Evaluation of Defence Results

In this section we evaluate the two proposed defence strategies—Outlier Detection using Recall Scores and Cosine Distance-Based Filtering—under controlled poisoning. The evaluation was done across two case scenarios, each representing a different level of poisoning and client compromise and using both IID and non-IID data distributions to test the defence mechanisms.

Case Scenarios and Setup

We simulate a federated learning environment with 10 clients, each training on their own local subset of the IoT-23 traffic dataset. Poisoning is introduced through label flipping, where a portion of the malicious traffic is mislabelled as benign, so clients will learn to ignore the true threats.

The scenarios are:

Case 1 – Low Poisoning (20%)

- Poisoned Proportion: 20% of malicious samples mislabelled as benign.
- Malicious Clients: 2 out of 10 clients (20% of clients).
- Goal: To check whether lightweight poisoning is detectable and whether global performance is impacted.

Case 2 – High Poisoning (50%)

- Poisoned Proportion:50% of malicious samples mislabelled as benign.
- Malicious Clients: 5 out of 10 clients (50% of clients).
- Goal: Assess model degradation and recovery under aggressive poisoning.

Each scenario was applied to IID-distributed datasets across clients. For the outlier-based approach, an additional evaluation was done on non-IID datasets to test its robustness in heterogeneous environments.

Results and Observations

A. Performance Under IID Conditions

1. Case 1: 20% Poisoning

- **Cosine distance strategy** found the 2 poisoned clients by gradient deviation. But global model performance (e.g. F1-score, recall) only improved slightly after removing these clients. So, while the defence works in detection, 20% poisoning is not strong enough to hurt the global model.

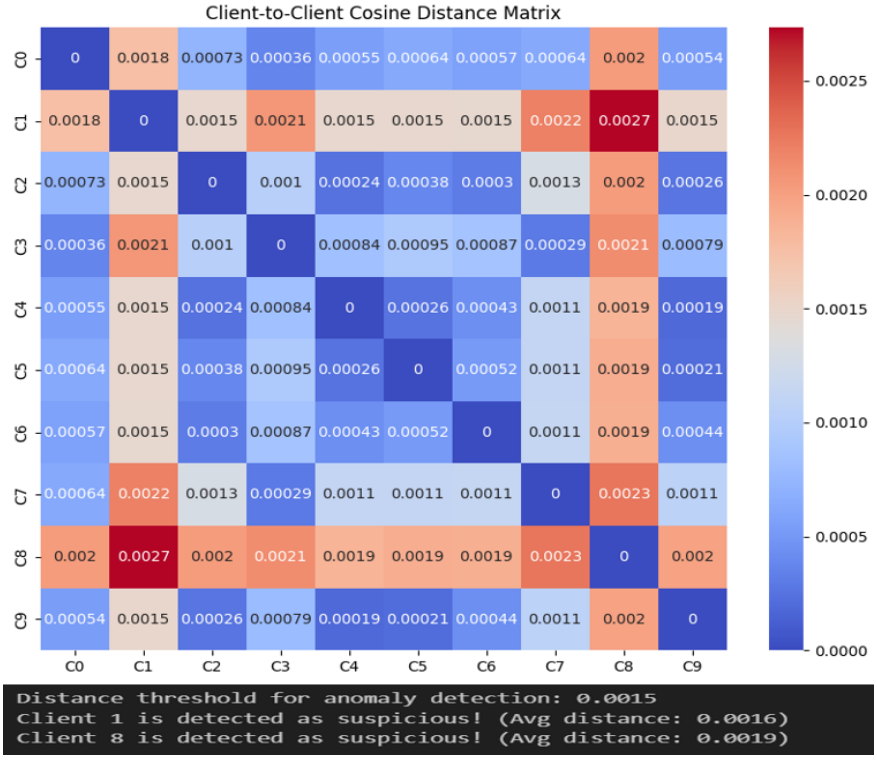

Figure 28: Cosine Distance Evaluation Defence Approach for 20% Attack

Figure 28 show the Cosine distance calculated with every client. The median threshold calculated is 0.0015. So it is observed that the distance of malicious clients is greater than threshold which indicated helps to identify compromised clients. The global performance with and without defence do not show significant change.

- **Recall based approach** found the 2 malicious clients as outliers. But threshold for recall outlier detection is very close to malicious clients' scores, so the system is prone to detect malicious clients falsely.

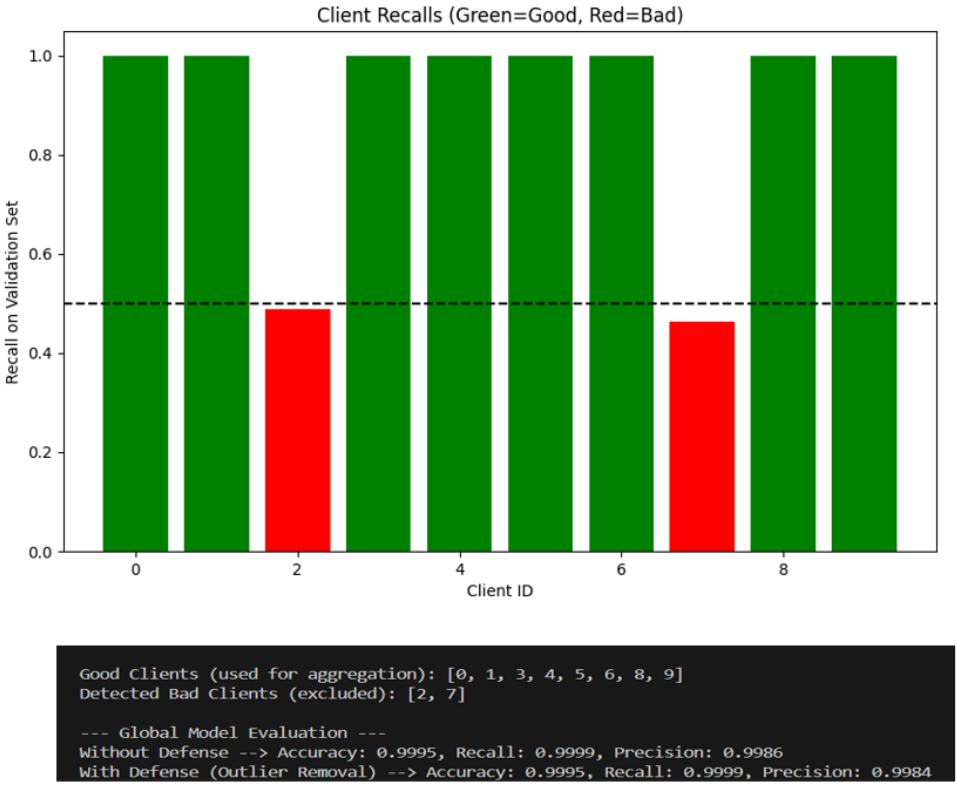

Figure 29: Outliner Defence Approach for 20% attack case.

Figure 29 Shows that despite correct detection, no significant uplift was observed in global model performance after defence in 20% poisoning case, so low level of poisoning may not be enough to warrant removal in practice. Given that this experiment was conducted where the dataset was IID

2. Case 2: 50% Poisoning

- **Cosine Distance Approach:**
  Once again, the cosine-based filter was able to accurately catch all 5 of the attackers due to their substantially different model behavior. Deletion of these clients made global performance return to near the baseline (i.e., no attack) levels. This shows this approach even under strong poisoning levels is effective.

  Figure 30 demonstrates the cosine distance calculation and how the distance of malicious client is always higher than the normal clients.

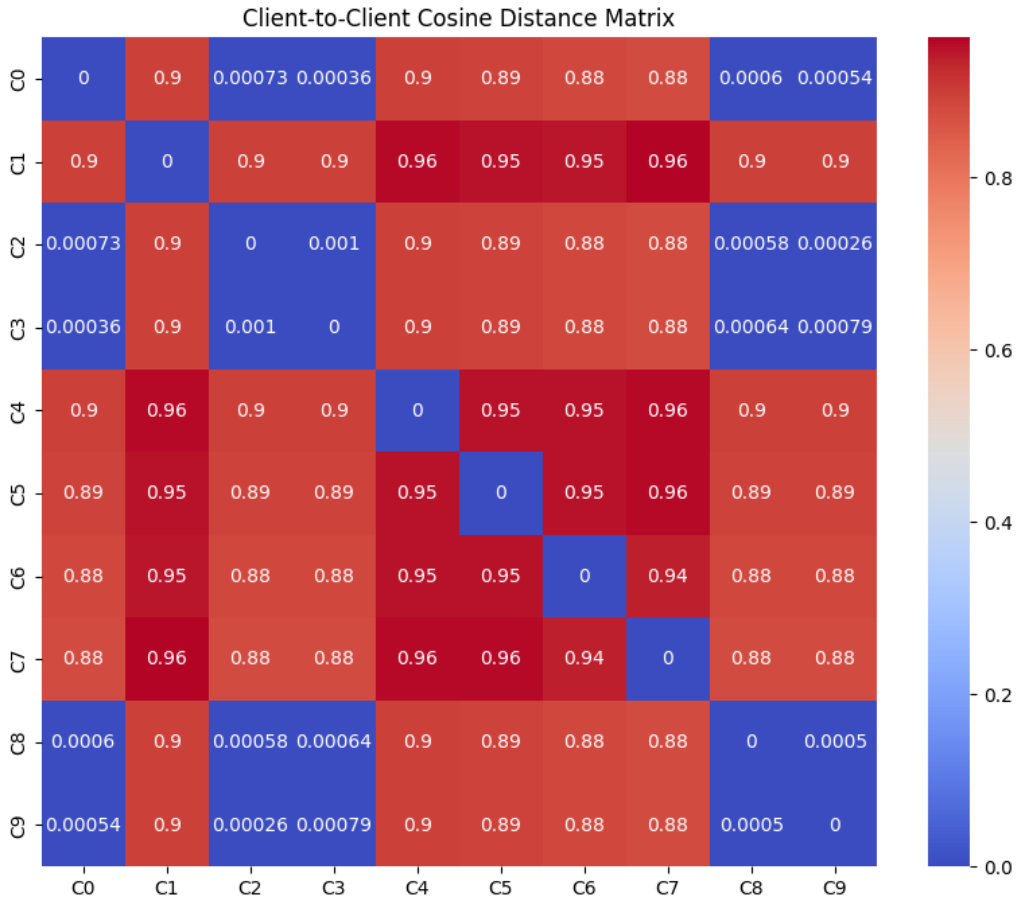

Figure 30: Cosine Distance Calculation with 50% Attack Case.

Figure 31 shows the global model performance before and after the Cosine similarity defending approach is applied. A significant drop in global model performance mainly in accuracy and F1 score is seen when this defence approach is not used. Using this approach, we can maintain the global model performance even when malicious clients are present in the environment.

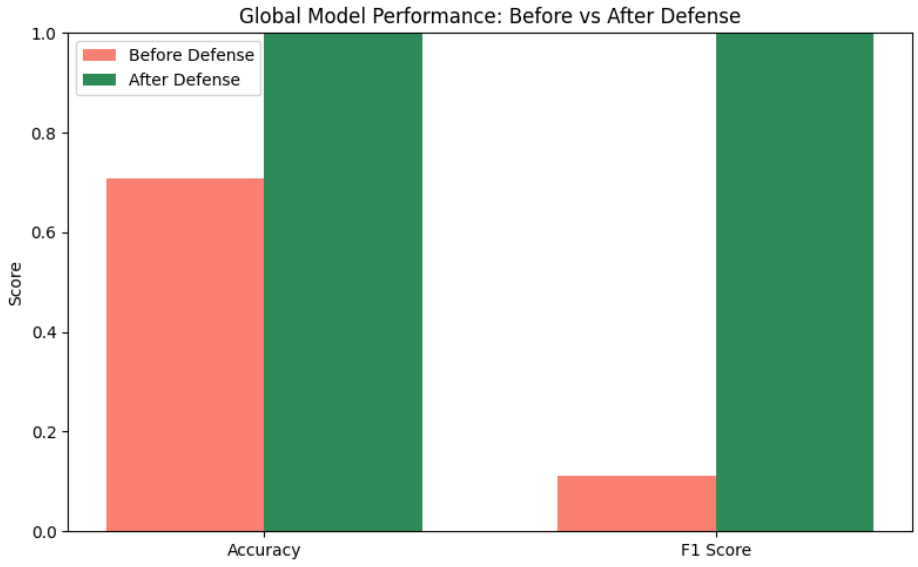

Figure 31: Global Performance Before and After Cosine Distance Defence Approach

- **Outliner Approach:**
  This approach also behaves correctly by classifying clients as outliers for whom the recall is very low. The median threshold effectively distinguished benign users from malign ones, and global recall, accuracy, and F1-score increased remarkably after removing the identify of obfuscated users. This confirms the usefulness of performance-based detection knowing that the attack is high enough to threaten the model.

  Figure 32 shows how the outliner approach detects the malicious clients when 50% poisoning attack is conducted. There is huge difference between the threshold and the recall values of the malicious clients. This shows that this approach works brilliantly when the poisoning attack is severe without a chance of causing any kind of false alarm.

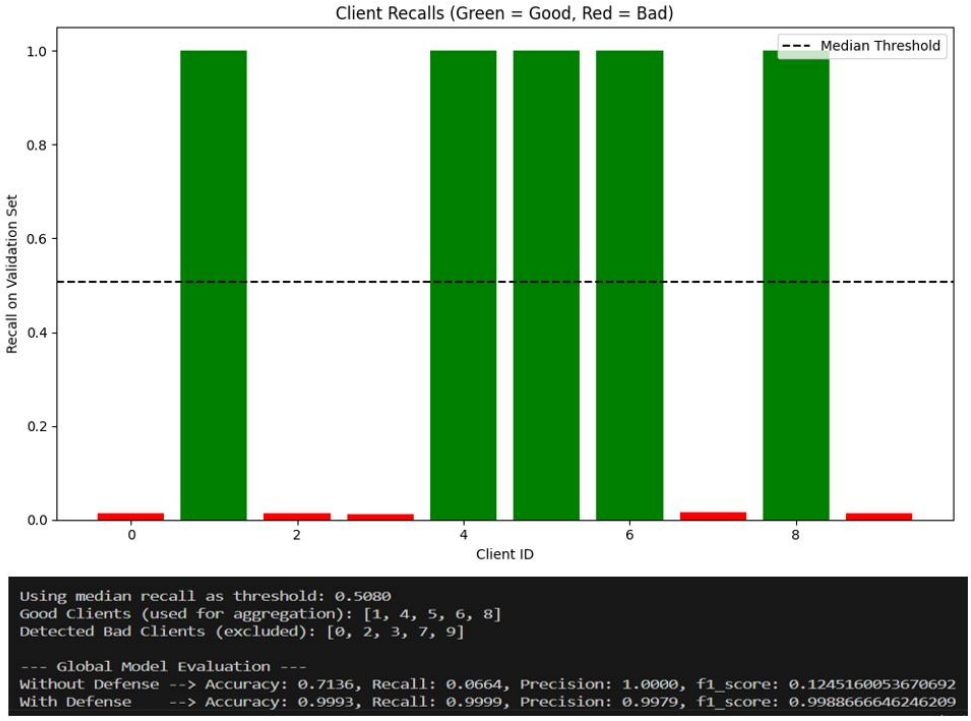

Figure 32: Outliner Approach for 50% Attack

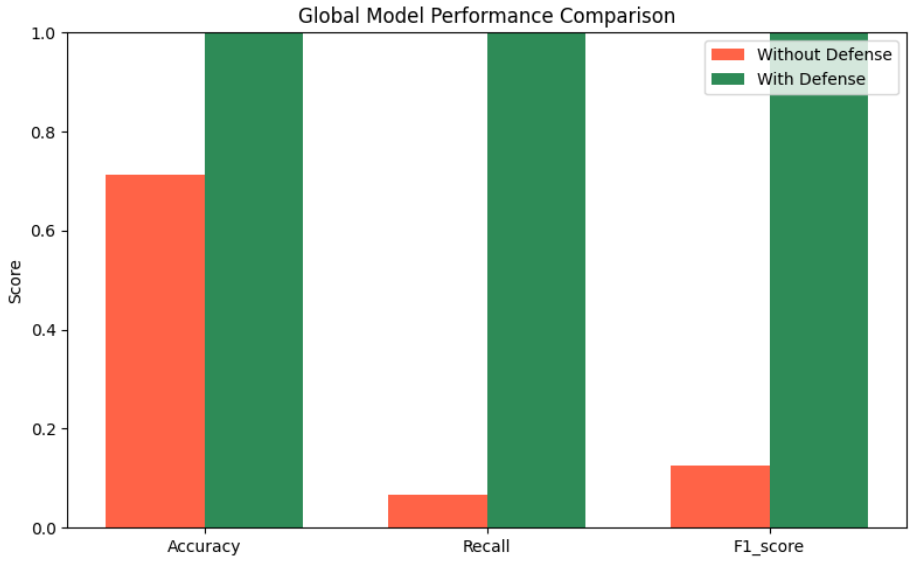

Figure 33: Global performance before & after Defence

Figure 33 shows the global model performance before and after the defence. The significant change in the global model performance mainly in accuracy, recall and f1_score values show that the poisoning attack causes a significantly high damage to the global model, and this can be defending using this approach.

B. Outliner Approach Performance Under non IID Conditions

To audit the generalizability of the outlier-based detection to non-IID settings, we made a data split where:

- A few of our clients had 90% good and 10% malware data
- Some had 80% malicious but there were also 20 % benign.
- A few had a balanced mix

This deployment mimics the non-uniformity in the traffic seen by various IoT devices in the real world.

1. Case 1 – 20% Poisoning (Non-IID)

In this case, the data was skewed as mentioned in the above description. This skew data was poisoned by 20% meaning only 20% of the malicious data was labelled as benign. Figure 34 illustrates the outcome of this experiment.

Figure 34: Outliner Defence Approach for 20% Attack in a non-IID environment.

- In this case the median recall threshold was unreasonably high when considering all clients, as some clients had 95-99\% benign data; this led to artificially high recall scores.
- Thus, in all the output clients' recall values were inferred to be less than a threshold; this implies all client is the suspecting client which is a false alarm.
- This evidence a major limitation of the outlier recall approach in non-IID: unbalanced class distributions can lead the statistical baseline to be distorted, and benign clients to be wrongly categorized as outlying.

Figure 35 Illustrates the global model performance with and without the outliner defence approach is applied. It is seen that when the poisoning attack ratio is smaller like 10-20% the global performance of the model remains almost same with and without defence. If this poisoning attack ratio increases this cases a huge impact on the global model performance.

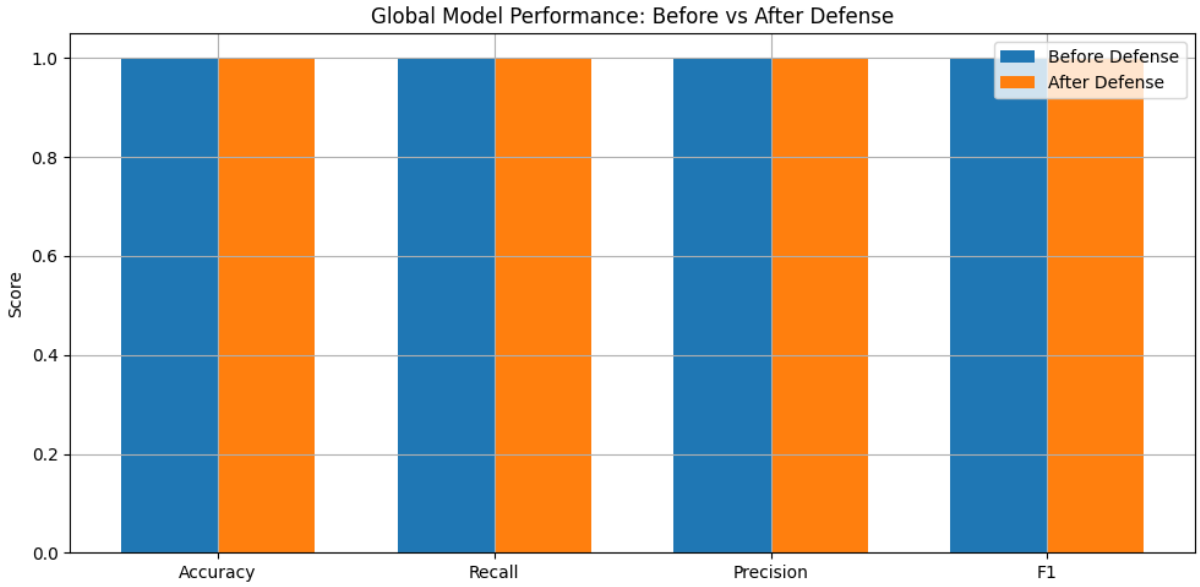

Figure 35: Global Model Performance of Outliner Defence Approach for 20% Attack.

Case 2 – 50% Poisoning (Non-IID)

In this case the same skewed dataset was used but the only difference was that the poisoning attack was increased to 50%. Figure 36 shows the outliner defence approach for 50% poisoning attack.

Figure 36:: Outliner Defence Approach for 50% Attack in a non-IID environment.

- On the other hand, the recall results of the malicious clients were still significantly lower with at most 50% poisoning, and the benign clients were effectively separated from the malicious clients using the outlier detection.
- As a result, malicious clients were well identified, and global model performance was effectively recovered through filtering.

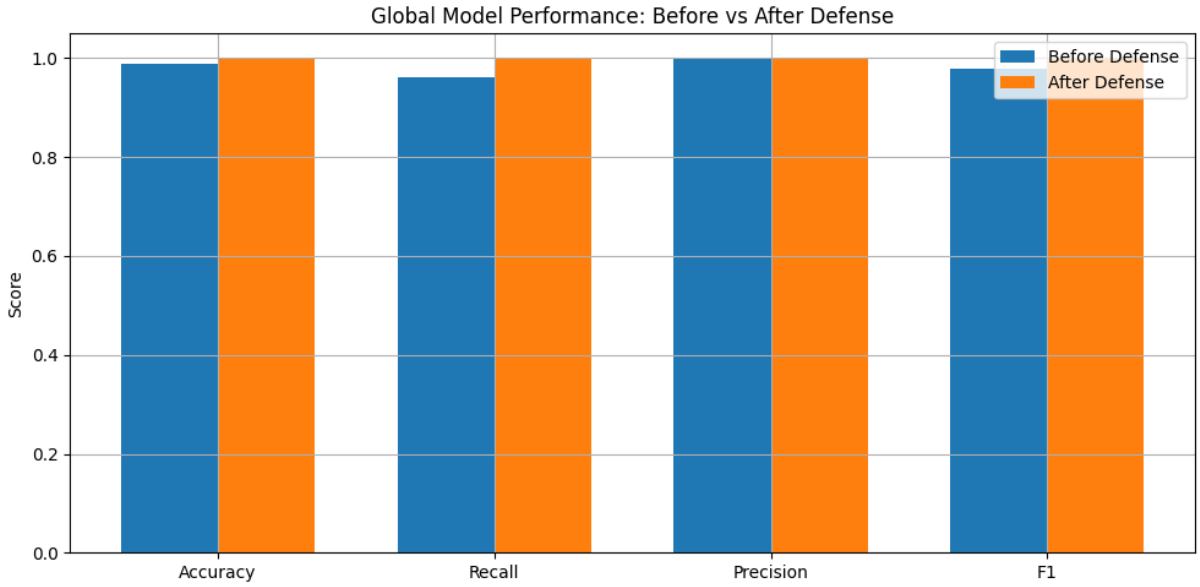

Figure 37: Global Model Performance for 50% Poisoning Attack in non IID environment

Figure 37 illustrates the global performance of the model when 50% poisoning was achieved. In an environment where a non IID dataset is used it is observed that the global performance when half of the dataset is poisoned still show significantly low performance degradation as compared to a IID setting (Refer Figure.16). But still in this case there is a slight difference between the accuracy and recall rate before and after the defence is applied.

Table 29: Summary of all the Findings

Case	Cosine Distance (IID)	Outliner Detection (IID)	Outliner (non-IID)
20% poisoning (2 Malicious clients)	Accurate detection, small performance gain	Accurate detection, small performance gain	Over-detection, all clients marked malicious
50% Poisoning (5 Malicious clients)	Accurate detection, high performance recovery	Accurate detection, high performance recovery	Accurate detection, small performance gain.

Table 29 shows the summary of all the findings using the different experimental scenarios. The key observations are as follows:

- Cosine distance defence approach is invariant to data distribution among clients under two poisoning levels and in IID settings.
- Outlier-driven techniques are sensitive to the homogeneity among the data points, causing false alarms in the presence of non-IID environment without any form of threshold adaptation.
- Low level poisoning (e.g. 20%) might not have a significant impact on the overall performance of the system, even when malicious clients are correctly diagnosed, leading to the question of when do we need to intervene.
- Hybrid approaches with both detection mechanisms in federated IoT environments in real life can provide a compromise between performance sensitivity and structural robustness.

6.4 Analysis and Discussion

This section represents a full-fledged discussion regarding the experimental results obtained during training of models using clean and poisoned IoT traffic datasets. It evaluates the resulting different behavior performances between the Recurrent Neural Network (RNN) and the XGBoost classifier. Such insights are crucial to understand the model vulnerabilities and robustness of the models regarding adversarial threats in the IoT ecosystem.

1. Performance Degradation of RNN in Poisoned Scenarios

In fact, the RNN model was able to show a pretty good performance with clean data but was able to make quite a large drop in terms of its recall and F1 scores when exposed to evaluation on poisoned datasets. This performance degradation stems from a number of technical and security-relevant factors.

- Overfitting to Spurious Patterns: RNNs are sequence-based models that can capture temporal patterns present in input data. However, in cases where training data is poisoned with very subtle perturbations or not very noticeable mislabelled flows, the RNN tends to overfit on those manipulated sequences. This occurs more with class-imbalanced datasets like IoT-23, where the model might learn the overrepresented class (benign flows) in a way that does not generalize towards the less frequent malicious patterns.
- Lack of Regularization Mechanisms: Without aggressive regularization such as dropout, early stopping, or weight decay, the RNN is left with the freedom to memorize inputs during training—especially when the poisoned flows are crafted in a way that seems benign but are connected with a malicious label. The model learns those poisonous associations without any boundaries and later fails to distinguish the original malicious activity during inference.
- Introduction of Token Manipulations: During tokenization, this includes adverse adversarial manipulation of feature sequences to poison the dataset. Since RNNs heavily depend on input sequences, slight rearrangements or creating tokens even within a model would derail its temporal learning process and consequently leads to a misclassification. It shows that RNNs are very strong when it comes to sequential modelling, but it's not very robust to adversarial perturbations in input flows unless trained to do so explicitly.
- Gradient Sensitivity and Label Leakage: Yet another factor that affects performance is the way poisoned RNN gradients react to such data. While such direction gradients from a training set of poisoned examples may leak among other possibilities, gentle suppression of activations belonging to malicious classes, it does so at the cost of decreased recall, where true attacks are classified as benign—a very critical failure from a security viewpoint.

This section interprets the results observed across clean and poisoned datasets:

- Why RNN might have failed in certain attack cases (e.g., overfitting, lack of regularization, token manipulation)
- Why XGBoost may show resilience or brittleness (e.g., reliance on robust tabular features)
- Importance of model interpretability and feature attribution
- Key lessons regarding model robustness in IoT traffic classification

2. Relative Robustness of XGBoost

XGBoost, in intervened conditions identical to RNNs, did not appear to suffer effects. While less so damaged and continued rather stable precision-recall characteristic, its performance deteriorated significantly. Motor underlines the following:

- Feature-wise Decision Tree Learning: This is judged as using criteria not ordered on the dependent sequential steps but are discrete in tabular features to produce those decision-rules and may use these tokens to apply transitivity for current manipulations with ordering in other cases typical of poising attacks. Overall, the evaluation feature importance will sustain its resistive behavior against the influence of adversarial outliers in individual samples.
- There would be the built-in pruning, L1/L2 regularization plus, not forgetting, shrinkage of learning rate. These features are the ones directly incorporated by any XGBoost mechanisms in order for them to

decrease model complexity for possible overfitting into the patterns adversaries have introduced. This would most probably work where there are instances of sparsity of contaminated samples or mis-labelled purposely, as xgboost tends to just ignore these noisy patterns during its training.

- Capable Handling of Imbalanced Data: with a proper tuning, say, scale_pos_weight, XGBoost can thus handle class imbalance so that the model does not entirely ignore minority malicious classes. This grants it a better chance of attack detection even in poisoned environments.
- Interpretable Feature Attribution: XGBoost allows interpretation of feature importance using the metrics of gain and cover. Such transparency helps identify which features are whereby manipulated or exploited to poison the dataset, thus providing feedback towards improving both the dataset and model architecture.

3. Model Interpretability and Its Importance in Security

Interpretability Whence has cast the model trust in security-critical domains like IoT. Although RNNs are highly capable at representation, they hold black-box properties that create diagnostic challenges against failure cases under poisoned conditions. On the contrary, tree-based models like XGBoost:

- Feature Importance Rankings: Helping to identify which features are most affected by poisoning.
- Decision analysis at path level: The reason a given flow was marked as benign or malicious can be traced.

This will help to design defence mechanisms, and it also identifies the poisoned data flows proactively with forensic inspection.

4. Security Lessons and Their Implications for Classifying IoT Traffic

Thus, the above analysis gives the following important takeaways:

- Robustness by Design Should be Proactive: Adversarial training, strong regularization, and a variety of model explainability tools make RNNs competitive with other neural models in an IoT scenario that contains adversarial agents.
- Defending with Ensemble and Hybrid Approaches: It can take advantage of the temporal context obtained from RNN with the strong decision boundaries offered by XGBoost by employing an approach that incorporates a temporal model like RNNs along with a tabular classifier such as XGBoost.
- Critical Evaluation Metrics for Attack Awareness: High precision may only reveal the surface of the weaknesses of a model (like false negatives); thus, recall and F1-score have to be prioritized in security applications since it is always more dangerous to miss an attack than to raise a false alarm.
- The Need for Continuous Monitoring and Learning to Counter Poisoning Attacks: With the static models, one cannot accommodate the very dynamic environment of IoT. Validation, retraining, and anomaly detection mechanisms must be continuous towards keeping pace with evolving changes in attacks.
- Trustworthy AI is at the Heart of IoT Security: Differential robustness against poisoning attacks has to be treated as a minimum for the safe deployment of AI systems in IoT rather than an afterthought.

5. Defence Mechanism Implications

The investigated defences, Outlier-based Recall Filtering and Cosine Distance Thresholding showed different capabilities depending on the poisoning intensity and the data distribution.

In IID settings, all both approaches successfully identified the malicious clients, and in 50% poisoning, remarkably enhanced the global model performance. There was a single exception with 20% poisoning, where the defence correctly identified the infected clients, but the global metrics did not change significantly. This emphasizes an important finding: low-volume attacks might not visibly affect overall performance but can still damage model integrity which highlights the importance of client level inspection.

The non-IID experiments, in particular with the Outlier Recall method, exposed difficulties. Misclassified distribution of thresholds over clients biased the median threshold and caused false positives—detecting honest clients as malicious in the cases with low poisoning. In contrast, Cosine Distance was more stable (based on model update compared to performance metrics) and therefore suitable for diverse environments.

These findings highlight the need for adaptive, distribution-aware defence mechanisms in federated IoT environments. The results from this study lay the groundwork for implementation of more robust, client-level anomaly detection methodologies, which are further investigated in the subsequent chapters.

7 Conclusion

The fast-expanding stage of the Internet of Things (IoT) has presented much better connectivity, and automation promises across all sectors, and it has also brought along unprecedented new risks with regard to security. This thesis has addressed one such immediate concerning challenge which is in regards with AI-based intrusion detection systems (IDS) having trained models subjected to initial data poison attacks - compromising model integrity during training - which led to permanent disturbing misclassification during that inference time.

The thesis examined the intricacies surrounding interactivity, the security of IoT networks, and adversarial machine learning in an attempt to understand and respond to data-poisoning attack challenges against AI-based intrusion detection systems. With IoT ecosystems pulsating far and wide, such vulnerabilities call for smart detection mechanisms that can withstand evolving cyber threats.

To this end, the thesis began by surveying the foundational concepts in IoT security, federated learning, and data poisoning, followed by a review of literatures pointing toward both accomplishments and inadequacies of current intrusion detection solutions. Such groundwork gave inspiration to the experimental contributions of this work.

Using IoT-23 as a representative benchmark dataset, a series of classification models were instantiated, trained, and evaluated in both clean and adversarially poisoned conditions. In the binary classification scenario, RNNs and XGBoost classifiers were trained to separate normal behavior from malicious traffic, while in the multiclass framework attack types could be further dissected for fine-grained detection.

The experiments provided important insight into the potential for data poisoning, especially with respect to label flipping, to severely degrade detection performance. RNNs were susceptible to the poisoning, especially at high levels of mislabeling, leading to huge decreases in recall with deceptively high precision. XGBoost also showed resilience but used more structured tabular features optimally. However, both models showed targeted misclassification patterns upon attack, suggesting that the data required for training must be treated with utmost integrity. Modest levels of label switching generally cause a drastic loss in detection capability, with misleading precision and dangerously suppressed recall not uncommon outcomes.

Despite these adversarial threats, two defensive mechanisms were proposed and tried in a federated learning ontology:

1. Outlier Detection Based on Recall Deviation;
2. Cosine Distance Thresholding of Client Updates.

Both methods proved to be effective, particularly in high poisoning scenarios under IID conditions, by successfully separating out malicious clients and confining their influence to the edge. Essence proved more stable than the alternative in non-IID situations, which happen when the distribution of data is biased towards one client or another. This provides hard evidence that defending against client-level anomalies is both feasible and important for preserving overall model soundness.

In summary, this thesis reaffirms that a full approach to IoT security systems should keep adversarial robustness at center stage. When we measure the severity of poisoning it is those defences that are lightweight and capable of being upheld promote the resilience of federated IDS pipelines, while at the same time they can be fully trusted, transparently audited and continuously maintained. This provides a starting point for future research on adaptive and proactive intrusion mitigation.

8 Future Scope and Perspectives

The findings and insights from this thesis provide multiple avenues for carrying forward both further research and practical developments in AI-based IoT security under adversarial conditions. This study made significant contributions in showing how poisoning attacks could influence the robustness of machine-learning-based systems, particularly for intrusion detection systems (IDS). On the other hand, it raises very interesting questions and significant opportunities for further growth. This chapter puts forward several potentially interesting directions that could build on the work presented in this thesis.

8.1 Backdoor Attacks and Trigger Injection

The thesis has mainly focused on label-flipping attacks, but future studies could extend into backdoor attacks-a very quiet and stealthy form of poisoning, where models are manipulated to behave maliciously under certain conditions of specific triggers.

Research questions of serious interest include:

- Could subtle hidden patterns, or byte-level triggers, be incorporated in network payloads so that they can bypass most IDS?
- Could transformer-based models like ET-BERT or NetMamba be misled into not finding the traces through some hidden triggers while still classifying them to be malicious flows?
- What kinds of trigger detection frameworks can be developed to search the trained models for vulnerable threats in them?

By analyzing backdoor attacks in the context of IoT traffic classification, very serious security threats can be revealed as well as models that will be built not only accurate but also tamper-evident.

8.2 Combine Federated Learning with Secure Model Aggregation.

Federated learning (FL) is a bright hope for privacy-preserving collaborative training across a wide array of distributed IoT devices. However, FL would also be subject to poisoning attacks as malicious clients insert poisoned updates into the system.

Immense research can be regards:

- Simulation of federated poisoning attacks using malicious IoT nodes.
- Private secure aggregation techniques (e.g., differential privacy, homomorphic encryption) for privacy preservation while preventing poisoned updates.
- Testing and evaluation of robust federated aggregation algorithms, including Krum, Trimmed Mean, or FoolsGold against manipulations from compromised participants.

Incorporating federated learning into IoT-based IDS systems enables real-time, distributed threat modelling while keeping the data localized and confidential

8.3 Model Interpretability and Explainable AI (XAI)

With the rise of more sophisticated forms of poisoning attacks, understanding the nuances of a model's decisions will become vital. Therefore, it would be of immense help to introduce explainable AI methods into the workflow of Intrusion Detection Systems to facilitate the identification of strange patterns in decision-making that can be traced to adversarial theory.

Potential research could include:

- SHAP or LIME for interpreting feature importance in both normal and adversarial scenarios.
- Visual dashboards for security analyst to interpret model predictions in real time.
- Attention maps generated through transformer models like ET-BERT to gauge disruptions in model attention.

Explainability will become essential for achieving trust and transparency, as well as for serving as a forensic aid in adversarial AI situations.

8.4 Hardware Constrained Deployment and Edge Security

IoT systems are usually low on computational resources, thus the need for lightweight and energy-efficient models for security. Future work may include:

- Optimization for edge inference of models such as NetMamba and RNNs by means of quantization, pruning, and/or distillation for edge inference.
- Trade-off analysis between model complexity and robustness in low-power settings.
- Perform anomaly detection on-device with very low-latency as required in time-sensitive cases like healthcare or industrial IoT.

This would achieve a seamless transition from academic study to real-life applicability in resource-constrained IoT environments.

8.5 Cross-Domain Application

To sum up, the methods and insights developed in this research can be extrapolated beyond traditional IDS and may influence:

- Healthcare IoT, where poisoning of medical records can lead to erroneous diagnoses.
- Smart grid networks, where tampered traffic could hide energy theft or grid manipulation.
- In autonomous vehicles, poisoned training data may mask hazard or road sign detection.

Cross-sector research will confirm the generalizability of poisoning defence strategies, therefore promoting the safer deployment of AI across critical sectors.

In summation, this intersection of ever-advancing poisoning attacks and ever-increasing use of AI sensing systems necessitates a multi-pronged defence against these threats involving robust algorithms, trusted data pipelines, explainable models, and resilient learning. The work done in this thesis sets a pathway toward this avenue and paves the way for real-life applications for positively impacting advanced secure and intelligent IoT infrastructures.

9 References

[1] Kejun Chen, Shuai Zhang, Zhikun Li, Yi Zhang, Qingxu Deng, Sandip Ray & Yier Jin, „Internet-of-Things Security and Vulnerabilities: Taxonomy, Challenges, and Practice," *Journal of Hardware and System Security,* Bd. 2, pp. 97-110, 2018.

[2] Victor R. Kebande, Joseph Bugeja, Jan A. Persson, „Internet of Threats Introspection in Dynamic Intelligent Virtual Sensing," arXiv, Cornell, 2020.

[3] Rodrigo Roman, Jianying Zhou, Javier Lopez, „On the features and challenges of security and privacy in distributed internet of things," *Computer Networks,* Bd. 57, Nr. 10, pp. 2266-2279, 2013.

[4] Kuzlu, Murat and Fair, Corinne and Guler, Ozgur, „Role of artificial intelligence in the Internet of Things (IoT) cybersecurity," *Discover Internet of things,* Bd. 1, p. 7, 2021.

[5] Md. Tarek Hossain, Rumi Afrin, Mohd. Al-Amin Biswas, „A Review on Attacks against Artificial Intelligience(AI) and their Defence Image Recognition and Generation Machine Learning,AI," *Control Systems and Optimization Letters,* pp. 52-59, 2024.

[6] Yohanes Yohanie Fridelin Panduman, Nobuo Funabiki, Evianita Dewi Fajrianti, Shihao Fang and Sritrusta Sukaridhoto, „A Survey of AI Techniques in IoT Applications with Use Case Investigations in the Smart Environmental Monitoring and Analytics in Real-Time IoT Platform".

[7] P. Srinivasa Rao, Syed Irfan Yaqoob, Mohammed Altaf Ahmed, Pardaeva Shakhnoza Abdinabievna, Syed Mufassir Yaseen & Mahendran Arumugam, „Integrated artificial intelligence and predictive maintenance of electric vehicle components with optical and quantum enhancements," *Optical and Quantum Electronics,* Bd. 55, Nr. https://doi.org/10.1007/s11082-023-05135-7, 2023.

[8] Kaushik Mishra; Goluguri N. V. Rajareddy; Umashankar Ghugar; Gurpreet Singh Chhabra,Amir H. Gandomi, „A Collaborative Computation and Offloading for Compute-Intensive and Latency-Sensitive Dependency-Aware Tasks in Dew-Enabled Vehicular Fog Computing: A Federated Deep Q-Learning Approach," *IEEE Transactions on Network and Service Management,* Bd. 20, Nr. 4, pp. 4600-4614, 2023.

[9] Venus Mohammadi, Amir Masoud Rahmani, Aso Mohammed Darwesh & Amir Sahafi , „Trust-based recommendation systems in Internet of Things: a systematic literature review," *Human-centric Computing and Information Sciences,* Bd. 9, Nr. 21, 2019.

[10] Jorge Eduardo Rivadeneira, Oscar Torres Sánchez, Moisés Dias, André Rodrigues, Fernando Boavida, Jorge Sá Silva,, „"CONFLUENCE: An Integration Model for Human-in-the-Loop IoT Privacy-Preserving Solutions Toward Sustainability in a Smart City," *IEEE Internet of Things,* Bd. 11, Nr. 5, pp. 8690-8714, 2021.

[11] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas, „Communication-Efficient Learning of Deep Networks from Decentralized Data," Cornell University, 2016.

[12] Kai Hu, Sheng Gong, Qi Zhang, Shanshan Jiang, Min Xia, Chaowen Seng, „An overview of implementing security and privacy in federated learning," *Artificial Intelligence Review,* Bd. 57, Nr. 10.1007/s10462-024-10846-8, 2024.

[13] Thien Duc Nguyen, Phillip Rieger, Markus Miettinen, Ahmad-Reza Sadeghi, „Poisoning Attacks on Federated Learning-based IoT Intrusion Detection System," Darmstadt, Germany.

[14] Floribert Katembo Vuseghesa, Mohamed-Lamine Messai, „Study on Poisoning Attacks: Application through an IoT Temperature Dataset," in *IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Bron, France, 2023.

[15] Peiming Xu, Peilin Luo, Yixin Jiang, Haojin Zhu, „Backdoor Detection in Digital Twin-Driven Smart Grid via Contribution Analysis," in *IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, China, 2024.

[16] Corey Dunn, Nour Moustafa, Benjamin Turnbull, „Robustness Evaluations of Sustainable Machine Learning Models against Data Poisoning Attacks in the Internet of Things," 10 August 2020.

[17] Baracaldo, Nathalie and Chen, Bryant and Ludwig, Heiko and Safavi, Amir and Zhang, Rui, „Detecting Poisoning Attacks on Machine Learning in IoT Environments," in *IEEE International Congress on Internet of Things (ICIOT)*, San Jose, CA, United States, 2018.

[18] Tomoki Chiba, Yuichi Sei, Yasuyuki Tahara, Akihiko Ohsuga, „A Defense Method against Poisoning Attacks on IoT Machine Learning Using Poisonous Data," in *IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering*, Tokyo, Japan, 2020.

[19] Miguel A. Ramirez, Song-Kyoo Kim, Hussam Al Hamadi, Ernesto Damiani, Young-Ji Byon, Tae-Yeon Kim, Chung-Suk Cho, Yeob Yeun, „Poisoning Attacks and Defenses on Artificial Intelligence: A Survey," Abu Dhabi, UAE, 2022.

[20] Yanxu Zhu, Hong Wen, Jinsong Wu, Runhui Zhao, „Online data poisoning attack against edge AI paradigm for IoT enabled smart city," *Mathematical Biosciences and Engineering,* pp. 17726-17746, 15 September 2023.

[21] Firoz Khan, R. Lakshmana Kumar,Mustufa Haider Abidi,Seifedine Kadry,Hisham Alkhalefah,Mohamed K. Aboudaif, „Federated Split Learning Model for Industry 5.0: A Data Poisoning Defense for Edge Computing".

[22] Aljanabi, Mohammad, „Safeguarding connected health: leveraging trustworthy AI techniques to harden Intrusion Detection systems against data poisioning threats in IoMT environments," *Babylonion Journal of Internet of Things,* pp. 31-37, 17 May 2023.

[23] S. García, D. Brumley, and M. Pechoucek, „IoT-23: A labeled dataset with malicious and benign IoT network traffic," in *Stratosphere IPS Project, Czech Technical University*, Czech Republic, 2020.

[24] Nour, Moustafa, „A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets," *Sustainable Cities and Society,* Bd. 72, Nr. 2021, p. 103046, 2021.

[25] Moustafa, Nour, and Jill Slay, „The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set.," in *Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015*, 2015.

[26] Moustafa, Nour, and Jill Slay, „UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set).," in *Military Communications and Information Systems Conference (MilCIS), 2015. IEEE*, 2015.

[27] Jasjeet Dhaliwal, Saurabh Shintre, „Gradient Similarity: An Explainable Approach to Detect Adversarial Attacks against Deep Learning," California, 2018.

[28] Xiaokuan Zhang, Haizhong Zheng, Xiaolong Li, Suguo Du, Haojin Zhu, „You are Where You Have Been: Sybil Detection via Geo-location Analysis in OSNs," Globecom 2014 - Communication and Information System Security Symposium, Shanghai, 2014.

[29] Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, Sridha Sridharan, „Correlation-aware Adversarial Domain Adaptation and Generalization," 2019.

[30] Natalie Baracaldo, Bryant Chen, Heiko Ludwig, Amir Safavi, Rui Zhang, „Detecting Poisoning Attacks on Machine Learning in IoT Environments," in *IEEE Internation Congress on Internet of Things*, San Jose, CA, US, 2018.

[31] Liu, Yu and Lin, Yujie and Hu, Hong and Luo, Xiapu, „ET-BERT: A Pre-Trained Model for Encrypted Traffic Classification Using BERT," in *2021 IEEE International Conference on Communications (ICC)*, IEEE, 2021, pp. 1--6.

[32] Xu, Xiaoqing and Liu, Zeyu and Xu, Yuan and Liu, Yu and Wu, Jie, „NetMamba: A Lightweight Pretrained Encoder for Network Traffic Representation," *arXiv preprint arXiv:2302.02924,* Nr. https://arxiv.org/abs/2302.02924, 2023.

[33] Saghir Ahmed, Basit Raza, Lal Hussain, Muhammad AMIN Nadim, „The Deep Learning ResNet101 and Ensemble XGBoost Algorithm with Hyperparameters Optimization Accurately Predict the Lung Cancer," *Applied Artificial Intelligience,* Bd. 37, Nr. 10.1080/08839514.2023.2166222. , 2023.

[34] „https://scapy.readthedocs.io/en/latest/".

[35] Ju Hyeon Lee, Jiho Shin, Jung Taek Seo, „Solar Power Plant Network Packet-Based Anomaly Detection System for Cybersecurity," *Computers, Materials & Continua,* 10.32604/cmc.2023.039461.

[36] Murat Kuzlu, Corinne Fair, Ozgur Guler, „Role of Artificial Intelligence in IoT CyberSecurity," *Discover Internet of Things,* 30 Novemnber 2020.

[37] Jenny Nadar, „Master Thesis Git Repository," *https://github.com/jennadar/Master_Thesis*"