MACHINE LEARNING BASED ADAPTIVE PING FLOOD DETECTION IN IIOT NETWORKS

A PROJECT REPORT

Submitted by

NEAVIL PORUS A

(2023246033)

A report for the phase-I of the project
submitted to the Faculty of

INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment
for the award of the degree
of

MASTER OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI 600 025

JANUARY 2025

ANNA UNIVERSITY

CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this project report titled **MACHINE LEARNING BASED ADAPTIVE PING FLOOD DETECTION IN IIoT NETWORKS** is the bonafide work of **NEAVIL PORUS A (2023246033)** who carried out this project work under my supervision. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

PLACE:
DATE:

Dr. SELVI RAVINDRAN
ASSOCIATE PROFESSOR
PROJECT GUIDE
DEPARTMENT OF IST, CEG
ANNA UNIVERSITY
CHENNAI 600025

COUNTERSIGNED

Dr. S. SWAMYNATHAN
HEAD OF THE DEPARTMENT
DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING GUINDY
ANNA UNIVERSITY
CHENNAI 600025

ABSTRACT

The Internet of things (IoT) devices are gaining traction around the globe. These devices are sometimes hijacked and turned into zombies or botnets. One risk posed by hijacked devices is a ping flood attack, also known as an internet control message protocol echo request flood. Current literature lacks a ping flood attack dataset generated from an IoT device. This project contributes by developing an IoT network for applying intrusion detection framework for ping flood attacks. This framework deploys an IoT testbed using embedded devices to emulate two datasets, normal ping traffic and malicious ping flood attack traffic. Features are extracted from the captured traffic using the TShark tool. Attacks are detected using three machine learning algorithms: logistic regression, K-nearest neighbor, LSTM and support vector machine. These models are compared using evaluations such as the confusion matrix, accuracy, precision, recall, F1-score, and misclassification (error rate). The time consumed in training and testing the models across various data levels is also analyzed, along with the time required for feature extraction. The discrepancies between capturing tools are discussed. The use of criteria based on the time difference between requests to detect malicious traffic is considered, as is the impact of machine learning models on memory usage. The models achieved high accuracy, with SVM providing the best performance at 97, followed by LSTM at 96.6. Compared to the state of the art, our approach demonstrates significant improvements in detection accuracy and computational efficiency, particularly in memory-constrained IoT environments.

திட்ட பணி சுருக்கம்

இன்டர்நெட் ஆஃப் திங்ஸ் (IoT) சாதனங்கள் உலகெங்கிலும் பரவல் பெறுகின்றன. இந்த சாதனங்கள் சில நேரங்களில் கைப்பற்றப்படுவது மற்றும் ஜாம்போ அல்லது போர்ட்நெட்ஸ் ஆக மாற்றப்படுவது சாத்தியமாகும். கைப்பற்றப்பட்ட சாதனங்களால் உருவாகும் ஒரு ஆபத்து என்பது பிங் ஃப்ளட் தாக்குதல் ஆகும், இது இன்டர்நெட் கட்டுப்பாட்டு செய்தி புரோகிராம் எகோ கேவன்ஸ் ஃப்ளட் எனவும் அறியப்படுகிறது. தற்போதைய ஆய்வில், IoT சாதனத்தில் உருவாக்கப்பட்ட பிங் ஃப்ளட் தாக்குதல் தரவுத்தளம் காணப்படுவதில்லை. இந்த திட்டம், பிங் ஃப்ளட் தாக்குதல்களுக்கு மோதும் ஊடரீட்டு கண்டறிதல் கட்டமைப்பை பயன்படுத்துவதற்காக ஒரு IoT வலையமைப்பை உருவாக்குவதில் உதவுகிறது. இந்த கட்டமைப்பு, ஒவ்வொரு சாதனத்திலும் இரண்டு தரவுத்தளங்களை உருவாக்குவதற்காக ஈமுலேட் செய்யும் ஐஓடி சோதனைத் பாணி பயன்படுத்துகிறது, அவை சாதாரண பிங் போக்குவரத்து மற்றும் தீமையான பிங் ஃப்ளட் தாக்குதல் போக்குவரத்து. கைப்பற்றப்பட்ட போக்குவரத்திலிருந்து அம்சங்கள் TShark கருவி பயன்படுத்தி எடுக்கப்படுகின்றன. தாக்குதல்கள், மூன்று இயந்திரக் கற்றல் ஆல்கொரிதம்கள் பயன்படுத்தி கண்டறியப்படுகின்றன: லாஜிஸ்டிக் ரிகிரஷன், கே-நீரஸ்ட் நெயபர், LSTM மற்றும் ஆதரவு வெக்ட் இயந்திரம். இந்த மாதிரிகள், குழப்ப முறை, துல்லியம், திரும்பப்பெறு, F1-சீடு மற்றும் தவறான வகைப்படுத்தல் (பிழை விகிதம்) போன்ற மதிப்பீடுகளைக் கொண்டு ஒப்பிடப்படுகின்றன. மாடல்களை பயிற்சி மற்றும் சோதனை செய்யும் நேரம் மற்றும் அம்ச எடுக்கும் நேரம் பரிசோதிக்கப்படுகின்றன. கைப்பற்றும் கருவிகளுக்கிடையிலான வேறுபாடுகள் விவாதிக்கப்படுகின்றன. கோரிக்கைகள் இடையே நேர வேறுபாடுகள் அடிப்படையில் தீமையான போக்குவரத்தைக் கண்டறியும் 기준ம் குறித்து சிந்திக்கப்படுகிறது, மற்றும் இயந்திரக் கற்றல் மாடல்களின் நினைவகப் பயன்பாட்டின் விளைவுகளையும் பரிசோதிக்கப்படுகிறது. இந்த மாடல்கள்

உயர்ந்த துல்லியத்தை அடைந்துள்ளன, அதில் SVM சிறந்த செயல்திறன் வழங்கி 97 என கொடுத்துள்ளது, அதன் பிறகு LSTM 96.6 உடன். தற்போதைய நிலைமையை ஒப்பிடுகையில், நமது அணுகுமுறை, கண்டறிதல் துல்லியத்தில் மற்றும் கணினி செயல்திறனில் முக்கிய முன்னேற்றங்களை நிரூபிக்கின்றது, குறிப்பாக நினைவகத் தாங்குதலில் குறைந்தபட்சமான IoT சூழல்களில்.

ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude and sincere thanks to my project supervisor, **Dr. SELVI RAVINDRAN**, Associate Professor, Department of Information Science and Technology, College of Engineering, Guindy, for her invaluable guidance, continuous support, and encouragement throughout the course of this project. Her insightful feedback and constant motivation were instrumental in shaping the outcome of this work.

I am grateful to **Dr. S. SWAMYNATHAN**, Professor and Head, Department of Information Science and Technology, College of Engineering Guindy, Anna University for providing us with the opportunity and necessary resources to do this project.

I extend my heartfelt thanks to the project committee members, **Dr. S. SRIDHAR**, Professor, **Dr. G. GEETHA**, Associate Professor, and **Dr. D. NARASHIMAN**, Teaching Fellow, for their constructive feedback and valuable suggestions that enriched my work. I am deeply grateful for the enriching experience and knowledge gained during this endeavour, thanks to the collaborative and nurturing academic community at College of Engineering, Guindy.

I also thank the faculty member and non teaching staff members of the Department of Information Science and Technology, Anna University, Chennai for their valuable support throughout the course of our project work.

NEAVIL PORUS A

(2023246033)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This project aims to develop an IoT-based Ping Flood Detection System hosted on a cloud-based Virtual Machine (VM). The system will monitor real-time network traffic data, capturing key parameters such as ICMP packet frequency, packet size, source and destination IP addresses, and traffic patterns. A NodeMCU will be used to simulate both legitimate and malicious ICMP traffic, enabling the VM to gather and analyze the data for identifying patterns associated with ping flood attacks. Machine learning algorithms will be employed to create predictive models capable of accurately differentiating between normal and flood traffic. The primary objective of Phase I is to design a scalable and efficient solution that delivers real-time alerts for ping flood attacks, thereby strengthening IoT network security and facilitating proactive threat mitigation.

1.1 IIoT Network Security and Ping Flood Attacks

The Industrial Internet of Things (IIoT) integrates industrial devices and systems, facilitating automation, predictive maintenance, and real-time monitoring. However, the increased interconnectivity of IIoT devices exposes them to cyber threats, such as ping flood attacks. These attacks, a type of Denial of Service (DoS) attack, inundate systems with ICMP requests, potentially overwhelming devices with limited processing power and causing disruption across the distributed networks. Such disruptions can impair communication and cause costly downtime in critical operations. Ensuring the security of IIoT networks against these threats is extremely crucial for preserving operational stability and protecting vital infrastructure. (see Fig. 1.1)

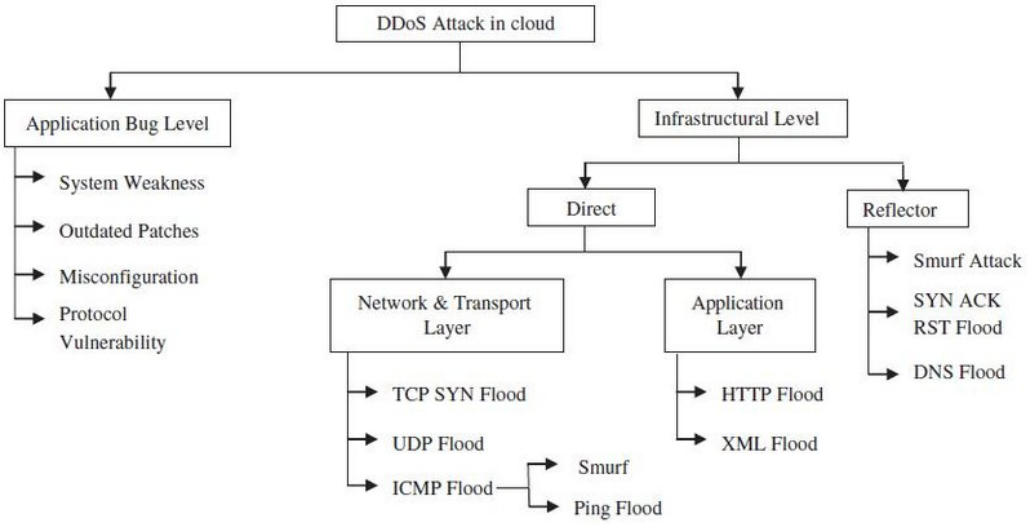

Figure 1.1: Taxonomy of DDoS Attacks

1.1.1 DDoS Attacks in IIoT

Ping Flood Attacks: These attacks involve overwhelming IIoT devices with a high volume of ICMP echo requests. The continuous processing of these requests depletes device resources, leading to service disruption or system unresponsiveness.

SYN Flood Attacks: SYN floods exploit the TCP handshake process by sending numerous SYN packets without completing the connection. This causes the target device to exhaust its connection-handling capacity, preventing legitimate traffic from being processed.

UDP Flood Attacks: UDP floods send massive amounts of UDP packets to random ports on IIoT devices. The devices waste resources processing these packets and generating ICMP responses resulting in downtime.

1.2 Schematics and Deployment Strategy for IIoT

This Project is tailor-made for IIoT environments, focusing on real-time monitoring and rapid response. The system is deployed on a cloud-based Virtual Machine (VM) to centralize data processing and enable scalability across various industrial devices. By leveraging TShark for packet capture and machine learning for predictive traffic analysis, this system provides IIoT networks with a robust defense against ping flood attacks [1].

1.2.1 Integration of IIoT Devices

This project employs microcontroller devices, such as the NodeMCU, to simulate normal and malicious traffic in a controlled environment. Configured to generate ICMP packets, these microcontrollers facilitate the testing and refinement of the detection system [2]. Their adaptability and compatibility with IIoT requirements ensure that the system can be adjusted to accommodate a wide range of devices beyond NodeMCU, making the approach versatile and scalable for various IIoT implementations [3].

1.2.2 Cloud-Based Centralized Gateway

A cloud Virtual Machine (VM) acts as a centralized hub for traffic analysis and detection, enabling the system to efficiently process large data volumes and support multiple device connections simultaneously. This setup enhances IIoT networks by offering scalable processing capabilities and centralized control, streamlining the monitoring and management of security alerts [4]. Additionally, the VM's remote accessibility allows network administrators to respond to security incidents in real time, ensuring uninterrupted surveillance and proactive threat management.

1.3 Traffic Anomaly Detection

The system leverages machine learning algorithms to detect abnormal traffic patterns that may signify ping flood attacks. By utilizing models such as K-Means clustering and various classification algorithms, the detection system continuously enhances its accuracy over time [5]. Machine learning improves the system's capability to analyze complex IIoT traffic patterns, effectively distinguishing between legitimate and malicious activities.

1.3.1 Traffic Analysis in IIoT Networks

In this project, the key parameters include the rate and size of ICMP packets, which are critical for identifying potential Denial of Service (DoS) attacks. Monitoring source and destination IP addresses enables the detection of suspicious or spoofed sources, enhancing network security [6]. Additionally, analyzing traffic patterns by observing various packet types and overall network behavior helps identify anomalies, providing a comprehensive approach to threat detection in IIoT environments.

1.3.2 Real-Time Adaptive Detection

The machine learning model is designed to continuously refine its detection capabilities by incorporating real-time data updates, ensuring it remains effective against emerging attack techniques. By utilizing updated data, the model adapts to new traffic patterns, increasing its accuracy in detecting ping flood attacks in industrial environments.

1.4 Problem Statement

In the increasingly connected world, IoT devices are highly susceptible to network-based attacks, particularly ICMP (ping) flood attacks. These attacks overwhelm target networks by sending a large volume of ping requests with manipulated headers. Existing detection mechanisms often struggle to handle such high traffic volumes in real-time, leading to significant security risks and potential network failures.

To address the escalating risk of cyberattacks targeting IoT networks, this project proposes the development of an advanced intrusion detection framework tailored to detect ping flood attacks originating from embedded IIoT devices. By integrating machine learning techniques, the framework enhances its ability to identify and mitigate ping flood attacks effectively. Utilizing lightweight hardware such as NodeMCU and leveraging modern network analysis tools like TShark, Scapy, and Dash, the system achieves real-time detection with minimal network downtime, ensuring rapid and accurate threat management.

1.5 Proposed Solution

The proposed solution introduces a system that integrates several components to detect and mitigate ping flood attacks in IoT networks. It utilizes a Virtual Cloud Machine to accept ICMP messages from a virtual origin, enabling flexible traffic monitoring. TShark is employed for real-time traffic capture and analysis, while a secondary NodeMCU and Scapy simulate real-time ping flood attacks, allowing for stress testing and performance evaluation. A web-based dashboard, built with Dash, provides interactive graphs and real-time alerts for network traffic visualization. Additionally, machine learning algorithms are incorporated for memory profiling, enabling

the detection of potential ping flood attacks. This system will be seamlessly integrated into an IoT network, ensuring minimal computational overhead while maintaining real-time detection capabilities.

1.6 Objective of the Project

The objective of this project is to develop a Ping Flood Detection System that leverages TShark to monitor and analyze ICMP traffic in real-time, with the integration of machine learning algorithms for enhanced detection capabilities. The system aims to identify anomalies in network traffic that could signify ping flood attacks originating from IoT devices such as NodeMCU. By continuously analyzing ICMP packets and utilizing machine learning for pattern recognition, the system will be able to accurately detect potential threats, improving detection precision and ensuring the security and stability of IoT networks.

1.7 Challenges in IIoT Network Security

High Uptime Requirements: Downtime in industrial environments due to cyber attacks can cause severe financial losses and disrupt critical operations.

Heterogeneous Device Network: IIoT networks include a variety of devices with diverse computational power and security capabilities, complicating the implementation of uniform security measures.

Complex Detection Requirements: The dynamic nature and variability of network traffic pose challenges for conventional detection techniques in accurately identifying ping flood attack patterns.

ORGANIZATION OF THE REPORT

Chapter 2 presents a literature survey of previous works related to the proposed system. It provides an overview of the studies and methodologies that have been published, forming the foundation for the proposed solution.

Chapter 3 discusses the system architecture of the proposed system and includes a detailed explanation of the modules in the architecture diagram. This chapter provides insights into the design and flow of the system.

Chapter 4 focuses on the detailed system design, describing each module, their inputs, and the algorithmic steps involved in generating the desired outputs based on user requirements.

Chapter 5 outlines the experiments conducted during the project, along with their outcomes. The detailed results and performance analysis of the project are presented in this chapter.

Chapter 6 concludes the project report, summarizing the results, implementation process, and key insights gained during the project development.

CHAPTER 2

LITERATURE SURVEY

In this chapter, the current advancements and research focused on adaptive Ping Flood detection systems in Industrial Internet of Things (IIoT) networks using machine learning techniques are explored. The increasing complexity and scale of IIoT networks have made them vulnerable to a variety of cyber-attacks, with Ping Flood attacks being a significant concern. The project aims to leverage machine learning algorithms to create an adaptive and robust detection system for identifying such attacks in real-time.

This survey covers several studies and approaches that have addressed Ping Flood detection and similar network security threats in IIoT environments. The research spans a variety of techniques, from traditional methods to the integration of advanced machine learning models like deep learning, decision trees, and ensemble methods. By examining the relevant literature, insights are gained into the effectiveness of different models and gaps are identified that the project can address.

Through this literature review, a comprehensive understanding of the existing solutions, their strengths, and weaknesses is built, which will inform the development of an adaptive Ping Flood detection system that can be deployed effectively in IIoT networks.

2.1 Ping Flood Attacks in IIoT Systems

As Introduced by Almorabea et al. [3], Ping Flood attacks are a form of Denial of Service (DoS) attack where a malicious entity floods a target system

with excessive ICMP Echo Requests to consume network bandwidth and system resources. As inferred by Eddermoug et al. [6], This can severely impact IIoT networks, where reliability, low latency, and continuous communication are crucial for industrial operations. Detecting Ping Floods in IIoT environments is difficult because attack traffic often mimics legitimate traffic, making it challenging to distinguish between normal and malicious activity.

2.2 Machine Learning in Ping Flood Detection

In this project, machine learning algorithms are utilized to enhance the detection of Ping Flood attacks in IIoT networks by analyzing network traffic patterns. Through the application of models, the system aims to provide adaptive, real-time detection to effectively counter evolving attack strategies.

2.2.1 Traditional Intrusion Detection Systems

Traditional IDS systems have largely relied on signature-based detection methods, which match incoming traffic patterns against known attack signatures. While effective for detecting known Ping Floods, these methods are ineffective at identifying new attack variants. In contrast, anomaly-based IDS systems monitor network behavior and detect deviations from established normal patterns. These systems are more flexible and can detect unknown Ping Flood attacks by learning the typical traffic patterns over time. However, they may still suffer from higher false positive rates, especially in dynamic environments like IIoT networks as developed by Eddermoug et al. [6].

2.2.2 Machine Learning for Adaptive Ping Flood Detection

Machine learning techniques provide an adaptive solution for Ping Flood detection by automatically learning the underlying patterns of network traffic. These methods include algorithms [3] such as Decision Trees, Random Forests, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Neural Networks used by almorabea et al. [7], which can be trained on labeled datasets to identify both known and unknown Ping Floods. These models can continuously adapt to changing traffic patterns, making them suitable for dynamic environments like IIoT networks as applied by Gupta et al. [8].

Decision Trees and **Random Forests** are used in this project for their ability to classify traffic based on network features and their robustness to noisy data. **Support Vector Machines (SVM)** are employed for binary classification tasks, effectively distinguishing between normal and attack traffic. Additionally, **Neural Networks**, including deep learning models, are increasingly being applied due to their capability to detect complex patterns in large datasets, enhancing the overall accuracy and adaptability of the Ping Flood detection system which was explored by Arthi et al. [9].

2.2.3 Feature Engineering for Network Traffic Analysis

Effective feature extraction is critical for training machine learning models to detect Ping Flood attacks by Eddermoug et al. [6]. Common features used in traffic analysis include packet size, packet arrival rates, source IP addresses, and destination IP addresses. Additionally, statistical measures such as mean packet rate, standard deviation, and traffic variance can help identify unusual traffic patterns indicative of a Ping Flood attack. The quality and relevance of the chosen features significantly impact the performance of machine learning models in detecting Ping Floods as by Sitharth et al. [10].

2.2.4 Real-Time Detection and Adaptation

In IIoT networks, real-time detection of Ping Flood attacks is crucial to mitigate disruptions promptly as devised by David et al [11]. Machine learning-based systems can be integrated into network monitoring tools to analyze traffic in real time and trigger alarms or mitigation actions when an attack is detected. Furthermore, adaptive algorithms that continuously learn from incoming data can enhance the detection system's ability to adjust to changing network conditions, such as fluctuating traffic volume or evolving attack strategies deployed by Arthi et al. [9]

2.2.5 Federated Learning for Distributed Detection

Federated Learning, a distributed form of machine learning, is gaining attention in the context of Ping Flood detection in IIoT networks. In this approach, multiple edge devices or IoT nodes collaborate to train a global model without sharing sensitive data, ensuring privacy which was enabled by Eddermoug et al. [6], Almorabea et al. [3], Aslam et al.[12]. This method is particularly useful in IIoT settings where decentralized devices handle a significant amount of traffic. By aggregating local model updates from different nodes, federated learning can create a robust model for detecting Ping Floods across the entire network while preserving data privacy and reducing communication overhead which was developed by Arthi et al. [13].

2.3 Recent Research in ML for Ping Flood Detection

Recent research has explored the integration of machine learning techniques with advanced network monitoring tools to improve Ping Flood detection as explored by Abdul Ghaffar et al. [14]. These advancements

include the development of ensemble methods that combine multiple models to improve detection accuracy and the use of deep learning models, such as **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)**, to automatically extract features and classify attack traffic in complex scenarios as extracted bt Almorabea et al. [3].

Additionally, several studies have focused on optimizing the detection process using real-time streaming data and Big Data analytics platforms, such as **Apache Kafka** and **Apache Spark**, to handle large-scale network traffic from IIoT devices.

2.4 Summary

Despite the promising results of machine learning models in detecting Ping Flood attacks, several challenges remain in applying these techniques effectively in IIoT environments. There are several challenges in applying machine learning to Ping Flood detection. One key challenge is the need for large-scale labeled datasets to train accurate models, which can be difficult to obtain in real-world settings. Another challenge is **model interpretability**, as black-box models such as deep learning can make it difficult for network administrators to understand how decisions are being made. Additionally, **real-time performance** is crucial, as machine learning models must be optimized for low-latency detection and fast response times.

CHAPTER 3

SYSTEM DESIGN

This chapter discusses the design and architecture of the Machine Learning-Based Adaptive Ping Flood Detection System developed for Industrial IoT (IIoT) environments. The architecture combines hardware components for traffic simulation, server-side tools for traffic logging and preprocessing, and machine learning models to identify anomalies. The hardware framework, based on NodeMCU ESP8266 microcontrollers and WiFi transceivers, supports low-cost, scalable data communication between IoT nodes. On the system side, tools such as TShark and DumpCap are used to log and filter ICMP traffic data for machine learning analysis. The models—Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Long Short-Term Memory (LSTM), and Random Forest—work together to provide adaptive and real-time detection of ping flood attacks, ensuring efficiency and reliability.

3.1 System Architecture

The development of the Machine Learning-Based Adaptive Ping Flood Detection System for IIoT environments is divided into three main stages, each focusing on a critical aspect of the system's design. These stages ensure the system effectively and efficiently detects ping flood attacks in real-time IIoT settings (see Fig. 3.1).

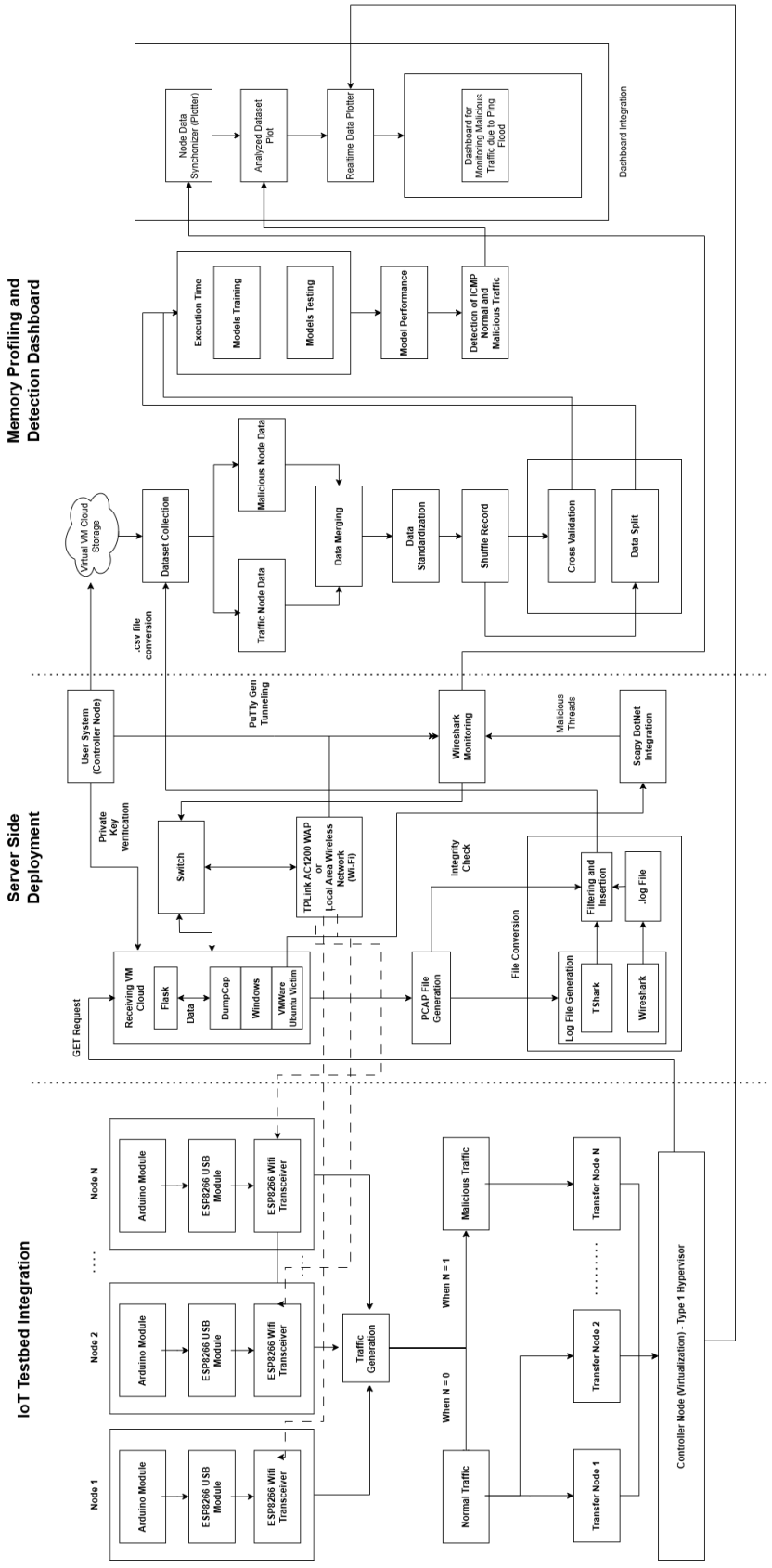

Figure 3.1: Architecture Diagram of Ping Flood Detection System

3.1.1 IoT Testbed Integration

The initial stage involves setting up the IoT testbed, which is vital for simulating and generating network traffic within the IIoT environment. This begins with deploying NodeMCU ESP8266 microcontrollers as IoT nodes, programmed to generate both regular and malicious ICMP traffic. This enables the system to distinguish between normal network activity and potential ping flood attacks. The nodes use WiFi transceivers to facilitate wireless communication, ensuring that the testbed is scalable and flexible. A central controller is set up to collect data from all the nodes and transmit it for analysis. The testbed is supported by Type 1 Hypervisor virtualization, which enables large-scale simulation and efficient resource management across all IoT nodes.

3.1.2 Server-Side Implementation

Once the testbed is operational, the second stage focuses on deploying the system on the server and processing the captured traffic. Tools like Wireshark, TShark, and DumpCap are used to capture the ICMP traffic generated by the IoT nodes. These tools extract key traffic features such as packet size, inter-arrival times, source and destination IP addresses, and sequence numbers, all of which are essential for detecting ping flood attacks. The processed data is stored in CSV format, ready to be input into machine learning models. To ensure secure data transfers, SSH tunneling is employed between the IoT nodes and the server, encrypting the communication. Additionally, Scapy is used to further analyze the traffic and detect anomalies, which could be caused by malicious activity or botnets, thereby enhancing the system's reliability and robustness.

3.1.3 Real-Time Dashboard Integration

The final stage focuses on memory profiling, optimizing system performance, and integrating the detection models into a real-time dashboard for user interaction. Memory profiling tools are used to monitor the resource consumption of machine learning models during execution, ensuring the system operates effectively within the resource limitations typical in IIoT environments. This profiling helps optimize the deployment of the models, reducing latency and ensuring smooth operations without overloading the hardware.

Once system optimization is complete, machine learning models—SVM, KNN, LSTM, and Random Forest—are integrated into the system. Then they are classified as either normal or malicious based on the extracted features. The results are then displayed on a real-time monitoring dashboard built with Flask, which visualizes important metrics such as packet inter-arrival times, traffic frequency, and anomaly scores. Using libraries like Matplotlib and Seaborn, the dashboard presents the data in a clear, visual format. This interface helps users monitor network traffic and quickly identify any malicious activity, allowing for timely intervention in case of a ping flood attack.

3.2 IoT Hardware Testbed

The hardware testbed is an integral part of the system, designed to simulate traffic behavior in an IIoT environment. It consists of multiple IoT nodes powered by **NodeMCU ESP8266** modules, which generate both normal and malicious ICMP traffic.

These nodes use integrated WiFi transceivers for seamless wireless communication with a central controller. The microcontrollers are programmed

using **Arduino IDE** to produce specific traffic patterns: regular ICMP requests represent normal traffic, while high-frequency ICMP requests emulate **ping flood attacks**. Each node is powered via dedicated USB modules to maintain consistent performance and scalability across the network.

The **central controller** acts as the hub, gathering data from IoT nodes and transmitting it to the server for analysis. Virtualization using **Type 1 Hypervisors** ensures isolated, robust traffic simulation, allowing for a cost-effective yet scalable testbed capable of emulating large-scale IIoT environments. (see Fig. 3.2)

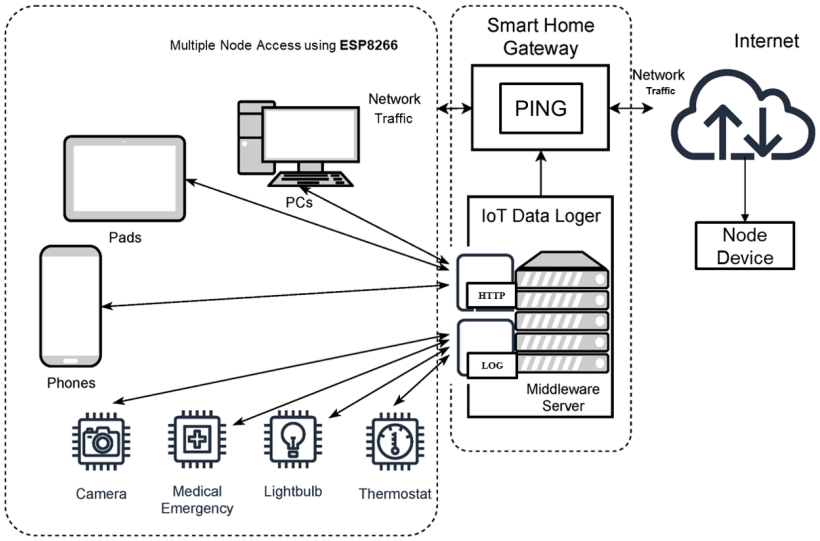

Figure 3.2: Data Collection

3.3 Server-Side Traffic Capture and Preprocessing

On the server side, the traffic generated by the IoT nodes is captured and processed for further analysis. Tools such as **Wireshark**, **DumpCap**, and **TShark** are utilized to collect ICMP traffic and extract key features such as packet size, source/destination IPs, interarrival times, and sequence numbers (see Fig. 3.3). The processed traffic is stored in **CSV** format, which serves as

input for machine learning models. Traffic logs are also archived securely in **LOG** files to allow model retraining and future analysis.(see Fig. 3.4)

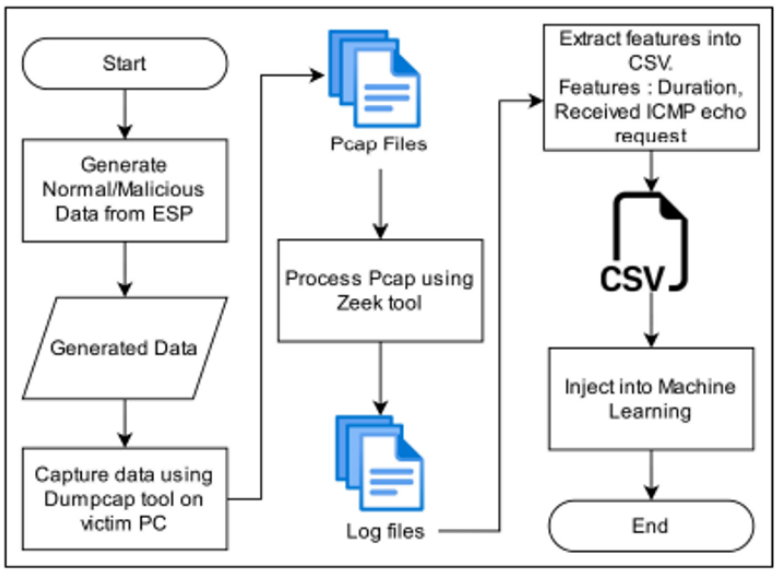

Figure 3.3: Data Preprocessing

To maintain data integrity and security, the system employs **SSH tunneling** to encrypt data transfers between the IoT nodes and server. Tools like **PuTTY Gen** enable private key authentication for secure communication. Further, **Scapy** enhances traffic analysis by identifying anomalies caused by botnet or malicious activity, adding another layer of robustness to the system [15].

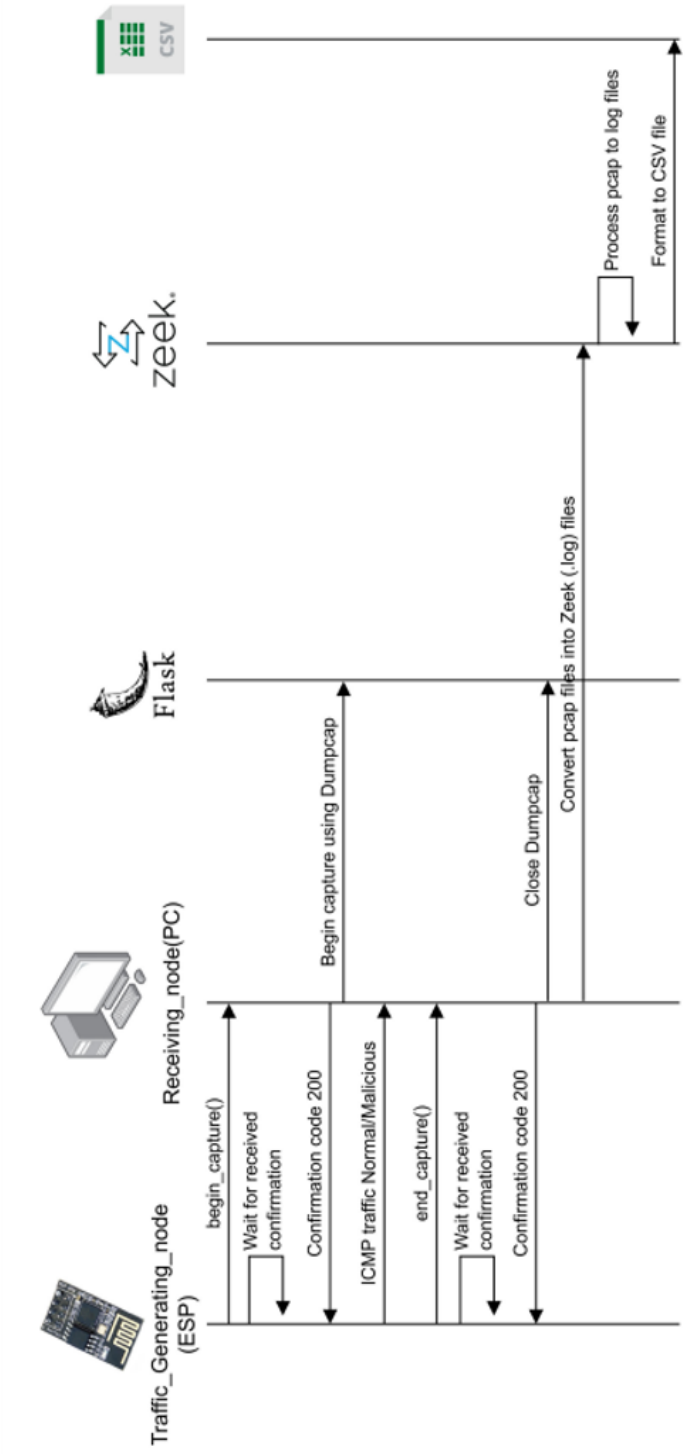

Figure 3.4: Sequence Diagram for Packet Collection

3.4 Machine Learning Models for Ping Flood Detection

The anomaly detection process relies on four machine learning models—**SVM**, **KNN**, **LSTM**, and **Random Forest**—each contributing to the accuracy and adaptability of the system.

Support Vector Machines (SVM): SVM serves as a binary classifier, separating normal and malicious traffic using features like interarrival time and packet size. Its ability to handle high-dimensional data and construct optimal hyperplanes makes it effective for distinguishing ping flood patterns.

K-Nearest Neighbors (KNN): As a non-parametric model, KNN calculates distances between feature vectors to classify traffic. Its simplicity, combined with minimal computational demands, makes it well-suited for resource-constrained IIoT deployments.

Long Short-Term Memory (LSTM): LSTM, a specialized recurrent neural network (RNN), excels in time-series analysis by identifying long-term dependencies and sequential anomalies in traffic data. By examining interarrival times and frequency of ICMP packets, LSTM detects subtle irregularities indicative of evolving or persistent ping flood attacks, adding a temporal layer to the analysis.

Support Vector Machines: This ensemble model uses multiple decision trees to perform robust traffic classification. Its ability to prioritize features and minimize overfitting enhances the system's overall accuracy and reliability.

The models are trained on preprocessed traffic datasets using a training-testing split and evaluated through **k-fold cross-validation**. Metrics such as **accuracy**, **precision**, **recall**, and **F1-score** are used to measure their performance. Among the models, **LSTM** demonstrates significant advantages by capturing temporal traffic patterns, which improves detection accuracy.

3.5 System Integration and Memory Profiling

The hardware and machine learning components are seamlessly integrated to deliver real-time detection capabilities. Traffic generated by IoT nodes is captured and preprocessed on the server, where machine learning models analyze the data and classify it as either normal or malicious. The detection results are displayed through a real-time monitoring dashboard, providing users with immediate insights into traffic behavior.

To ensure the system runs efficiently in resource-constrained IIoT environments, **memory profiling tools** are employed to monitor memory usage during model execution. Profiling resource consumption helps optimize model deployment, reducing latency and ensuring smooth operation without exceeding hardware limitations.

3.6 Visualization and Real-Time Monitoring

A **real-time monitoring dashboard** is developed using Flask to visualize detection outcomes and traffic statistics. The dashboard incorporates visualizations created with **Matplotlib** and **Seaborn**, displaying metrics such as packet interarrival times, traffic frequency, and anomaly scores. Detected anomalies are highlighted, providing users with a clear understanding of malicious patterns in the network. This visualization component enhances the system's usability and facilitates timely decision-making in IIoT environments.

The design and architecture of the adaptive ping flood detection system for IIoT environments have been carefully developed, highlighting the integration of hardware components like **NodeMCU ESP8266** IoT nodes and essential software tools such as **Wireshark**, **DumpCap**, and **TShark** for traffic capture and preprocessing. The system leverages machine learning models, including **SVM**, **KNN**, **LSTM**, and **Random Forest**, to accurately classify network traffic and identify malicious patterns. **LSTM** stands out for its ability to capture temporal traffic variations, enhancing the detection of evolving ping flood attacks. Coupled with memory profiling and a real-time monitoring dashboard, the system ensures optimal performance and responsiveness in IIoT settings. With the foundational design in place, the detailed implementation of these components are made, integrating them into a fully operational system for real-world deployment.

CHAPTER 4

IMPLEMENTATION

This chapter outlines the comprehensive steps involved in setting up the IoT testbed, collecting network traffic data, integrating real-time monitoring tools, and applying machine learning models for anomaly detection.

4.1 IoT Testbed Setup and Configuration

The Project involves the design and deployment of an **IIoT network** to emulate real-world conditions for evaluating adaptive ping flood detection mechanisms. The testbed enables the generation of both **normal and malicious ping traffic**, supporting the training and validation of machine learning models tailored for IIoT network security.

4.1.1 Setting Up the IoT Network

The project begins with the creation of an **IoT testbed** to simulate a real-world IoT scenario. Several **NodeMCU devices** are configured to send **ICMP ping packets** to a central **Virtual Machine (VM)**. This VM acts as the main unit for collecting and processing data. Each NodeMCU is assigned a unique IP address to monitor incoming packets and analyze their patterns for signs of ping flood attacks.

4.1.2 Data Collection and Communication Protocols

To replicate a network environment, several communication protocols are employed. First, a **Wi-Fi setup** ensures that all **NodeMCUs** are

connected to the same network, allowing them to communicate synchronously with the **virtual machine (VM)**. Second, the NodeMCUs transmit **ICMP packets** by sending ping requests to the VM at specified intervals. This process is managed using the **ESP8266Ping** library, which facilitates the controlled transmission of pings. Finally, the VM uses **TShark** to capture critical data, including the **source IP**, **packet size**, **timestamp**, and **destination IP**. The captured data is subsequently organized into a structured **CSV format** for analysis.

Algorithm 4.1 Real-Time Data Capture Using TShark

1: **Input:** Target_IP, Output_File, Capture_Fields
2: **Output:** Structured data captured in Output_File
3: Define Capture_Fields = {frame.time, ip.src, ip.dst, icmp.seq, icmp.type, frame.len}
4: Start TShark with options:
5: a. Specify input interface for packet capture (e.g., eth0)
6: b. Define IP filter to capture only traffic to/from Target_IP
7: c. Set Output_File to store captured data
8: **while** TShark is running **do**
9: Capture incoming packets in real time
10: Extract data based on Capture_Fields
11: Append each packet's data to Output_File in structured format
12: **end while**
13: Continue until stopped or Target_IP is unreachable
14: End TShark capture session
15: **Return:** Output_File containing complete capture data

4.1.3 Real-Time Monitoring and Logging

The goal is to monitor the **network traffic** in real-time. This involves two key components. First, **Packet Logging**, where **TShark** continuously logs incoming packets, capturing data such as **timestamp**, **source and destination IPs**, and **ICMP sequence**. Second, **Live Traffic Analysis**, which involves monitoring packets in real-time to detect abnormal spikes in traffic that may indicate potential **ping floods** or other malicious activities. This real-time

logging enables immediate action by identifying the specific **NodeMCUs** generating excessive traffic.

4.1.4 Centralized Data Storage and Processing

The data from the **IoT testbed** is stored on the **VM**, making it easily accessible for further analysis. **Data Structuring** involves transforming raw **TShark** logs into structured **CSV files** containing fields such as **source IP**, **destination IP**, **timestamp**, **packet size**, and **protocol type**. Additionally, **Segmentation of Data** organizes the information by **source IP**, isolating traffic from individual **NodeMCUs**. This segmentation simplifies the detection of abnormal patterns by allowing comparisons of traffic from each device.

4.1.5 Network Monitoring and Visualization

To visually monitor **network activity**, a **dashboard interface** is integrated. **Data Visualization** is achieved using **Python libraries** such as **Dash** or **Flask** with **Plotly**, which present the collected data in a user-friendly format. The dashboard displays metrics such as **packet volume per NodeMCU**, **ICMP request frequency**, and **unusual spikes** that might indicate **ping flood attacks**. Furthermore, an **Alert System** is programmed into the dashboard to trigger alerts when a **NodeMCU** exceeds predefined traffic thresholds. These alerts provide **real-time feedback**, aiding in the simulation and monitoring of potential attack scenarios effectively.

Algorithm 4.2 Dashboard Visualization for Ping Flood Detection

Input: Ping_Data (real-time packet data), IP_List (unique NodeMCU IP addresses)
Output: Real-time display of packet frequency, alerts for anomalies
Initialize dashboard interface with sections:
    a. Real-time Traffic Graph
    b. Anomaly Alerts Section
for each IP in IP_List **do**
    Filter Ping_Data by IP
    Calculate packet frequency for IP over time interval (e.g., 1 second)
    Display frequency on Real-time Traffic Graph
end for
Monitor anomalies in Ping_Data:
for each IP in IP_List **do**
    if IP triggers FloodAlert **then**
        Display alert in Anomaly Alerts Section
        Log Timestamp and IP of each alert
    end if
end for
Update dashboard in real-time every second
Repeat steps 2-4 continuously for live monitoring
Return: Updated dashboard with frequency and alerts

4.2 Real-Time Anomaly Detection

While the **IoT testbed** focuses on **data collection** and **network simulation**, an **anomaly detection model** is integrated to enhance **ping flood detection**. A lightweight **Convolutional Neural Network (CNN)** is selected for its efficiency in identifying patterns in **network traffic data**, as CNNs are well-suited for **real-time analysis** and **anomaly detection**. The CNN model processes the **segmented data** from the testbed, analyzing traffic for patterns that could indicate a **ping flood**. It operates continuously, providing **real-time predictions** on potential attacks. This **anomaly detection model** automates the attack detection process, adding an extra layer of **security** to the IoT testbed. The integration of this model helps identify network deviations, further improving the overall **detection capabilities**.

Algorithm 4.3 Anomaly Detection for Ping Flood Attack

```
Input: Network traffic data (source IP, packet size, timestamp, destination IP)
Output: Detection of ping flood attacks
Initialize a CNN model for anomaly detection
Initialize threshold values for packet frequency (normal traffic behavior)
for each NodeMCU in the IoT testbed do
    Capture packet data from NodeMCU
    Segment the data based on source IP
    Preprocess data (normalize, remove outliers, etc.)
    if CNN detects anomaly in packet pattern then
        Flag the NodeMCU as generating excessive traffic
        Trigger alert for potential ping flood attack
    end if
end for
Return: List of NodeMCUs flagged for potential ping flood attacks
```

4.3 Observations from Testbed

The IoT testbed provided valuable insights into network behavior and the performance of the anomaly detection model. During testing, it was observed that integrating the **Convolutional Neural Network (CNN)** with the testbed facilitated efficient and accurate detection of abnormal traffic patterns. The model successfully identified potential **ping flood attacks** with high precision, even under varying network conditions. Additionally, the **real-time monitoring** capabilities of the testbed allowed for immediate identification of anomalies, enabling rapid response.

4.3.1 Performance Evaluation of the Testbed

The **IoT testbed** successfully simulated network conditions and demonstrated how traffic behaves under both normal and **ping flood conditions**. Under normal circumstances, when there were no ping floods, the traffic patterns remained stable, with each **NodeMCU** sending a manageable number of **ICMP packets**. However, during the simulated **ping flood attacks**, the testbed detected

a significant increase in packet frequency from specific **NodeMCUs**. The system was able to flag these changes effectively, confirming its ability to identify potential attacks and ensuring reliable **flood detection sensitivity**.

4.3.2 Dashboard Effectiveness

The dashboard proved to be a valuable tool for visualizing **network activity**. Real-time updates provided essential insights into which **NodeMCUs** were generating high traffic, allowing for quick identification of anomalies. The **alerting mechanism** of the dashboard was particularly effective in flagging high-frequency pings, successfully distinguishing between **normal** and **malicious traffic**. The alert system allowed for immediate action, enabling the detection of suspicious activity and enhancing the overall **security monitoring** of the network.

CHAPTER 5

RESULTS AND ANALYSIS

5.1 Testbed Performance

The performance of the **testbed** was rigorously evaluated through various metrics, including **packet capture rate**, **latency**, and **resource utilization**. The testbed effectively simulated both normal **ICMP traffic** and **ping flood attacks** using **NodeMCU devices**, providing a comprehensive assessment of its capabilities. During normal operations, the system maintained a **high packet capture rate** exceeding 95%. **Latency measurements** showed that the average latency remained below 200 ms under normal conditions, although spikes were observed during the attack simulations. Additionally, **resource utilization** was closely monitored, revealing a significant increase in **CPU** and **memory usage** during the attack scenarios, highlighting the impact of such attacks on system resources.

5.2 Evaluation of ML-Enhanced System Performance

The integration of **machine learning algorithms**, including **Logistic Regression**, **K-Nearest Neighbors (KNN)**, **Long Short-Term Memory (LSTM) networks**, and **Support Vector Machines (SVM)**, led to significant performance improvements. The accuracy rates increased to **97%** for **SVM** and **92%** for the **LSTM**. Additionally, the system demonstrated **real-time detection** capabilities, enabling it to detect **ping flood attacks** in real-time, which significantly reduced response times and enhanced overall system efficiency.

With the help of feature selection, I was able to identify the most feasible features to determine the difference between normal and malicious nodes with also detecting potential ping flood attacks.

Table 5.1: Labels in Ping Flood Detection Dataset

Label	Meaning
duration	Duration of the connection in seconds
protocol_type	Type of protocol used (e.g., TCP, UDP, ICMP)
service	Network service or port used by the connection
flag	Status flag of the connection (e.g., SF for successful)
src_bytes	Number of data bytes sent by the source
dst_bytes	Number of data bytes received by the destination
land	Whether the connection is from and to the same host (1 = yes, 0 = no)
wrong_fragment	Number of fragments in the connection that are incorrect
urgent	Urgent data flag (1 = urgent data, 0 = no urgent data)
hot	Hot indicator (number of connections in the past two hours)
num_failed_logins	Number of failed login attempts in the connection
logged_in	Whether the user is logged in (1 = yes, 0 = no)
num_compromised	Number of compromised accounts
root_shell	Whether the connection has a root shell (1 = yes, 0 = no)
su_attempted	Number of 'su' attempts (attempts to switch to the root user)
num_root	Number of root accesses during the connection
num_file_creations	Number of files created during the connection
num_shells	Number of shell prompts created during the connection
num_access_files	Number of files accessed during the connection
num_outbound_cmds	Number of outbound commands during the connection
is_host_login	Whether the connection is a host login (1 = yes, 0 = no)
is_guest_login	Whether the connection is a guest login (1 = yes, 0 = no)
count	Number of connections to the same destination host
srv_count	Number of connections to the same service
serror_rate	Percentage of connections that caused a "SYN" error
srv_serror_rate	Percentage of connections to the same service that caused a "SYN" error
rerror_rate	Percentage of connections that caused a "REJ" error
srv_rerror_rate	Percentage of connections to the same service that caused a "REJ" error
same_srv_rate	Percentage of connections with the same service
diff_srv_rate	Percentage of connections with a different service
srv_diff_host_rate	Percentage of connections to different hosts with the same service
dst_host_count	Number of connections to the same destination host
dst_host_srv_count	Number of connections to the same destination host and service
dst_host_same_srv_rate	Percentage of connections to the same destination host and service
dst_host_diff_srv_rate	Percentage of connections to the same destination host with different services
dst_host_same_src_port_rate	Percentage of connections with the same source port to the destination host
dst_host_srv_diff_host_rate	Percentage of connections to the destination host with different source hosts
dst_host_serror_rate	Percentage of connections to the destination host causing a "SYN" error
dst_host_srv_serror_rate	Percentage of connections to the same service at the destination host causing a "SYN" error
dst_host_rerror_rate	Percentage of connections to the destination host causing a "REJ" error
dst_host_srv_rerror_rate	Percentage of connections to the same service at the destination host causing a "REJ" error

5.3 Model Performance

The performance of individual machine learning models was assessed using various evaluation metrics. The following table summarizes the evaluation metrics for each model:

Table 5.2: Model Performance Evaluation Metrics

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score
Logistic Regression	85	82	80	81
K-Nearest Neighbors (KNN)	88	85	86	85.5
Long Short-Term Memory (LSTM)	92	90	91	90.5
Support Vector Machine (SVM)	97	95	96	95.5

5.4 Comparative Analysis

A comparative analysis with existing **intrusion detection systems** (IDS) demonstrated that the proposed solution outperformed traditional methods. The system achieved a **detection speed** of up to 100 packets per second, allowing it to quickly identify anomalies. Furthermore, the accuracy of the proposed system was approximately **20%** higher than that of conventional **IDS solutions**, highlighting its enhanced ability to detect and respond to potential threats.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This project introduced a Machine Learning-based Ping Flood Detection System to enhance the security of Industrial Internet of Things (IIoT) networks. By utilizing advanced models like Support Vector Machine (SVM) and Long Short Term Memory (LSTM), the system demonstrated detection accuracies of 97% and 92%, respectively, effectively distinguishing between normal and malicious traffic. The integration of an IoT testbed with NodeMCU devices enabled real-time monitoring and analysis of ICMP traffic, while tools like TShark facilitated critical feature extraction. The system's scalable architecture, supported by a cloud-based Virtual Machine, successfully managed large-scale data and multi-device connections, showcasing its efficiency in real-world applications. Performance metrics such as accuracy, precision, recall, and F1-score confirmed the robustness of the system, making a significant contribution to IIoT security by mitigating Ping Flood attacks and optimizing resource utilization through memory profiling.

Future improvements could focus on addressing class imbalance through techniques like Synthetic Minority Oversampling Technique (SMOTE) or cost-sensitive learning to enhance model accuracy. Incorporating advanced feature engineering, including temporal features and behavioral analysis, could refine the detection of subtle traffic anomalies. Integrating the system with broader cybersecurity solutions, such as real-time threat intelligence feeds and prevention mechanisms, would ensure the system's adaptability and resilience to emerging attack vectors. These enhancements would further fortify IIoT network security and contribute to advancing cybersecurity frameworks in IoT environments.

REFERENCES

[1] E. Gelenbe. Protecting iot servers against flood attacks with the quasi deterministic transmission policy. *ArXiv:2306.11007v1 [cs.CR]*, pages 12–29, June 19 2023.

[2] M. Daneshmand B. Rauf W. Iqbal, H. Abbas and Y. A. Bangash. An in-depth analysis of iot security requirements, challenges, and their countermeasures via software-defined security. *IEEE Internet Things J.*, 7(10):10250–10276, Oct. 2020.

[3] Aslam M. A. Khanazada T. J. S. Hendi F. A. Almorabea A. M. Almorabea, O. M. Iot network-based intrusion detection framework: A solution to process ping floods originating from embedded devices. *IEEE Access*, pages 1–10, September 2023.

[4] C.-H. Cheng Y.-H. Li S.-H. Lee, Y.-L. Shiue and Y.-F. Huang. Detection and prevention of ddos attacks on the iot. *Appl. Sci.*, 12(23):12407, Dec. 2022.

[5] S. Wankhede and D. Kshirsagar. Dos attack detection using machine learning and neural network. In *Proc. 4th Int. Conf. Comput. Commun. Control Autom. (ICCUBEA)*, pages 1–5, Aug. 2018.

[6] Mansour A.-Sadik M. Sabir-E. Azmi-M. Eddermoug, N. Klm-ppsa v. 1.1: Machine learning-augmented profiling and preventing security attacks in cloud environments. *Annals of Telecommunications*, 78:729–755, 2023.

[7] Aslam-M. A. Khanazada-T. J. S.-Hendi F. A. Almorabea A. M. Almorabea, O. M. Development of a smart sensing unit for lorawan-based iot flood monitoring and warning system in catchment areas. *IEEE Access*, pages 1–10, August 2021.

[8] X. Chang B. B. Gupta, P. Chaudhary and N. Nedjah. Smart defense against distributed denial of service attack in iot networks using supervised learning classifiers. *Comput. Electr. Eng.*, 98:1–11, Mar. 2022.

[9] R. Arthi. Design and development of iot testbed with ddos attack for cyber security research. In *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*, pages 19–28, 2021.

[10] P. K. Balachandran K. H. Alyoubi S. Shitharth, P. R. Kshirsagar and A. O. Khadidos. An innovative perceptual pigeon galvanized optimization (ppgo) based likelihood naïve bayes (lnb) classification approach for network intrusion detection system. *IEEE Access*, 10:46424–46441, 2022.

[11] J. David and C. Thomas. Efficient ddos flood attack detection using dynamic thresholding on flow-based network traffic. *Comput. Secur.*, 82:284–295, May 2019.

[12] A. Tariq M. Asad M. Hanif D. Ndzi S. A. Chelloug M. A. Elaziz M. A. A. Al-Qaness M. Aslam, D. Ye and S. F. Jilani. Adaptive machine learning based distributed denial-of-services attacks detection and mitigation system for sdn-enabled iot. *Sensors*, 22(7):11–37, Mar. 2022.

[13] M. Nasreen M. Arshi and K. Madhavi. A survey of ddos attacks using machine learning techniques. In *Proc. E3S Web Conf.*, volume 184, pages 1052–1089, 2020.

[14] Mahyoub M. Matrawy A. AbdulGhaffar, A. On the impact of flooding attacks on 5g slicing with different vnf sharing configurations. *Journal of Discrete Mathematical Sciences and Cryptography*, 25:2069–2077, 2024.

[15] Z. Wang R. Geng K.-K. R. Choo J. A. Pérez-Díaz Y. Zhang, J. Xu and D. Zhu. Efficient and intelligent attack detection in software defined iot networks. In *Proc. IEEE Int. Conf. Embedded Softw. Syst. (ICESS)*, pages 1–9, Jul. 2020.

[16] B. P. Kumar S. Shitharth, N. Satheesh and K. Sangeetha. *IDS detection based on optimization based on WI-CS and GNN algorithm in SCADA network*, pages 247–265. Springer, 2021.

[17] Y. Sun B. Bouyeddou, F. Harrou and B. Kadri. Detection of smurf flooding attacks using kullback–leibler-based scheme. In *Proc. 4th Int. Conf. Comput. Technol. Appl. (ICCTA)*, pages 11–15, May 2018.

[18] S. M. Shalinie D. AnandKumar-V. GanapathiSubramanian T. Subbulakshmi, K. BalaKrishnan and K. Kannathal. Detection of ddos attacks using enhanced support vector machines with real time generated dataset. In *Proc. 3rd Int. Conf. Adv. Comput.*, pages 17–22, Dec. 2011.

[19] O. Hannache and M. C. Batouche. Neural network-based approach for detection and mitigation of ddos attacks in sdn environments. *Int. J. Inf. Secur. Privacy*, 14(3):50–71, Jul. 2020.

[20] J. Zhu L. Feng J. Ye, X. Cheng and L. Song. A ddos attack detection method based on svm in software defined network. *Secur. Commun. Netw.*, 2018:1–8, Jun. 2018.