THE UNIVERSITY OF DANANG
DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY

SPECIALTY: SOFTWARE ENGINEERING

PROJECT TITLE:

BUILDING A SYSTEM FOR NETWORK SURVEILLANCE AND ANOMALY DETECTION

Instructor: **Dr. LÊ THỊ MỸ HẠNH**
Student: **THÁI DUY VIỆT**
Student ID: **102210088**
Class**:** **21TCLC_DT1**

Da Nang, 06/2026

THE UNIVERSITY OF DANANG
DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY

SPECIALTY: SOFTWARE ENGINEERING

PROJECT TITLE:

BUILDING A SYSTEM FOR NETWORK SURVEILLANCE AND ANOMALY DETECTION

Instructor: **Dr. LÊ THỊ MỸ HẠNH**
Student: **THÁI DUY VIỆT**
Student ID: **102210088**
Class**:** **21TCLC_DT1**

Da Nang, 06/2026

SUMMARY

Topic title: Building a System for Network Surveillance and Anomaly Detection
Student name: Thái Duy Việt
Student ID: 102210088
Class: 21TCLC_DT1

In the era of increasing cyber threats, traditional signature-based Intrusion Detection Systems often struggle to keep pace with sophisticated, evolving network attacks. This thesis introduces a system for network surveillance and anomaly detection designed to provide proactive, automated network security monitoring.

The system architecture leverages Zabbix for real-time network metric collection, which is then processed by an XGBoost machine learning model to analyze traffic patterns and accurately detect anomalous activities—such as ICMP Floods, SYN Flood and UDP Flood. Data integrity is maintained via MongoDB, ensuring scalable and secure storage for both real-time status and historical logs.

The frontend provides a real-time dashboard for visualizing key network metrics and AI prediction results, enabling administrators to monitor system status in a clear and efficient way. By integrating AI-based analysis with a modular and secure architecture, the system supports real-time network monitoring and improves the detection of abnormal traffic in a practical and scalable manner.

The trained XGBoost model achieves an overall classification accuracy of 99.19%, with a macro-averaged F1-Score of 0.99. The system satisfies practical real-time monitoring requirements, achieving an average inference latency of less than 0.0022 milliseconds and a processing throughput of one complete monitoring cycle every 5 seconds. Under attack simulations, the end-to-end detection delay spans from 18 to 21 seconds.

DA NANG UNIVERSITY
UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY INFORMATION TECHNOLOGY

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom - Happiness

GRADUATION PROJECT REQUIREMENTS

Student Name: THAI DUY VIET Student ID: 102210088

Class: 21TCLC_DT1 Faculty: Information Technology

Major: Information Technology

1. *Topic title: BUILDING A SYSTEM FOR NETWORK SURVEILLANCE AND ANOMALY DETECTION*
2. *Project topic: ☐has signed an intellectual property agreement for final result*
3. *Initial figure and data: None*

Content of the explanations and calculations:

- **Introduction:** This chapter gives information about the context and purpose of the project, as well as the scope of the problems that will be focused on in the thesis.
- **Chapter 1:** Theories and Technologies**.**
- **Chapter 2:** System Analysis and Design**.**
- **Chapter 3:** Implementation and Evaluation**.**
- **Conclusion**.

4. *Drawings, charts (specify the types and sizes of drawings):*

- Use case diagram
- Activity diagram
- Entity diagram

5. *Name of instructor:* Lê Thị Mỹ Hạnh
6. *Date of assignment : 06/03/2026*
7. *Date of completion : ..../06/2026*

Da Nang, date month year 2026

Head of Division……. **Instructor**

Lê Thị Mỹ Hạnh

PREFACE

Upon completing the academic curriculum at the University of Science and Technology – The University of Danang, I chose the thesis titled "Building a System for Network Surveillance and Anomaly Detection" as my final graduation project. This research serves as a pivotal opportunity not only to apply the technical knowledge I have accumulated over the past four years but also to explore, research, and integrate modern technologies, thereby enhancing my professional capabilities for my future career.

I would like to express my sincere gratitude to the Faculty of Information Technology for providing the favorable conditions necessary for me to conduct this graduation project. Throughout the development process, I have received invaluable encouragement and support from my lecturers, family, and friends. Their companionship has been a tremendous source of motivation, enabling me to overcome technical challenges and successfully fulfill my research objectives.

In particular, I would like to extend my deepest appreciation to Dr. Le Thi My Hanh, my supervisor, who has consistently provided dedicated guidance and strategic direction at every stage of this research. His enthusiasm and insightful professional feedback have been crucial to the successful completion and refinement of this thesis.

Although I have dedicated my best efforts to ensuring this project is as comprehensive and robust as possible, given the inevitable limitations in practical experience and research time, some shortcomings may remain. I sincerely look forward to receiving constructive feedback and suggestions from the faculty, which will allow me to further improve this work and better prepare myself.

ASSURANCE

I hereby declare that this thesis is the result of my own research, conducted under the supervision of **Dr. Le Thi My Hanh**. All content and findings presented here are original and honest. The data used for analysis and assessment have been collected from various sources and are clearly cited in the list of references.

Should any instance of academic dishonesty or plagiarism be discovered, I accept full responsibility for the content of my research. The University of Science and Technology – The University of Danang shall not be held liable for any copyright or intellectual property infringements caused by my actions during the implementation of this research.

Student performed

Thai Duy Viet

TABLE OF CONTENTS

INTRODUCTION 1
Chapter 1 : THEORIES AND TECHNOLOGIES 4
1.1. System Architecture Overview 4
1.2. Theoretical Foundations of Intrusion Detection Systems 4
1.2.1. Concept and Architectural Classification of IDS 4
1.2.2. The Emergence of AI in Intrusion Detection 5
1.3. Analysis of Cyberattack Vectors 5
1.3.1. ICMP Flood 5
1.3.2. SYN Flood 6
1.3.3. UDP Flood 7
1.4. Theoretical Foundations of the XGBoost Classification Algorithm 7
1.4.1. Decision Tree – The Base Unit 7
1.4.2. How does XGBoost work? 7
1.4.3. Mathematics Behind the XGBoost Algorithm 8
1.4.4. Why XGBoost? 8
Chapter 2 : SYSTEM ANALYSIS AND DESIGN 11
2.1. Descriptive overview of objects 11
2.2. Functional Description 11
2.2.1. For Root (Super Administrator) 11
2.2.2. For Administrator 11
2.2.3. For Monitor (Security Analyst) 12
2.3. Use-case diagram 12
2.3.1. General 12
2.3.2. Root (Super Administrator) *13*
2.3.3. Administrator *15*
2.3.4. Monitor *15*
2.4. Activity diagram 16
2.5. Entity diagram 16
2.5.1. Table overview *16*
2.5.2. Entity diagram *17*
2.6. Network Topology and Infrastructure Design 19
2.6. Architecture diagram 20
2.7. User interface 21
2.7.1. Login *21*

2.7.2. Real-time dashboard *21*
2.7.3. Traffic Monitoring *22*
2.7.4. User Management *22*
2.8. Security & System 24
2.8.1. Potential attack vectors *24*
2.8.2. Authentication Flow *24*
Chapter 3 : IMPLEMENTATION AND EVALUATION 26
3.1. Implementation 26
3.1.1. Data Collection and Feature Optimization *26*
3.1.2. Input Feature *26*
3.1.3. Model training *27*
3.2. Overfitting Analysis 30
3.4. Comparison of Machine Learning Models 31
3.5. Evaluation 32
3.5.1. Normal traffic *32*
3.5.2. ICMP Flood Detection *33*
3.5.3. TCP SYN Flood Detection *34*
3.5.4. UDP Flood Detection *35*
3.5.5. Performance *37*
3.5.6. Scalability *38*
CONCLUSION 39
1. Achieved Results: 39
2. Limitations 40
3. Future work 40
REFERENCES 42
APPENDIX 43

LIST OF TABLES, FIGURES

TABLE

Table 1-1: Comparative Evaluation of RF, LSTM, and XGBoost 9
Table 2-1: General Use-case description.....12
Table 2-2: Use-case description for Super Administrator.....14
Table 2-3: Table overview.....16
Table 2-4: Users entity details 17
Table 2-5: Live_status entity details 18
Table 2-6: History_log entity details.....18
Table 2-7: Live_log entity details 18
Table 3-1: Description of the input features 27
Table 3-2: Distribution of classes in the dataset 27
Table 3-3: Hyperparameters used for training the XGBoost.....28
Table 3-4: Model training report.....29
Table 3-5: False Positive Rate by traffic class.....30
Table 3-6: Five-Fold Cross-Validation Results 31
Table 3-7: Comparison of classification accuracy among ML Models 32
Table 3-8: Latency and Throughput Evaluation 37

FIGURE

Figure 1-1: Ping ICMP DDoS Attack Diagram.....6
Figure 1-2: TCP Three-way handshake diagram 7
Figure 2-1: General Use-case diagram.....12
Figure 2-2: Use-case for Super Administrator's features.....13
Figure 2-3: Use-case for Admin's features.....15
Figure 2-4: Use-case for Monitor's features.....15
Figure 2-5: Activity diagram for Network Monitoring and Threat Detection.....16
Figure 2-6: Entity diagram.....17
Figure 2-7: Network topology in GNS3 19
Figure 2-8: Architecture diagram.....20
Figure 2-9: Login Page UI 21
Figure 2-10: Real-time dashboard Page 21
Figure 2-11: Network Packet Visualization in Traffic Monitoring Interface.....22
Figure 2-12: Response Time and UDP Traffic to Closed Ports Visualization 22
Figure 2-13: User Management Page.....23
Figure 2-14: Edit User Interface 23
Figure 2-15: User Update Successful Notification 23
Figure 2-16: User Creation Successful Notification 24
Figure 3-1: Confusion matrix.....29
Figure 3-2: ROC Curves.....30
Figure 3-3: The monitoring dashboard during normal network operation.....32
Figure 3-4: Network metrics during normal network operation.....33
Figure 3-5: The execution of the ICMP Flood attack from the attacker machine 34
Figure 3-6: ICMP Flood detection alert on the monitoring dashboard 34
Figure 3-7: Network metrics during the ICMP Flood attack.....34
Figure 3-8: The execution of the TCP SYN Flood attack from the attacker machine....35
Figure 3-9: TCP SYN Flood detection alert on the monitoring dashboard.....35
Figure 3-10: Network metrics during the SYN Flood attack 35

Figure 3-11: The execution of the TCP UDP Flood attack from the attacker machine ..36
Figure 3-12: TCP UDP Flood detection alert on the monitoring dashboard.....36
Figure 3-13: Network metrics during the UDP Flood attack 37

LIST OF SYMBOL, ACRONYM

SYMBOL

ACRONYM

No	Acronym	Explanation
1	UI	User Interface
2	IP	Internet Protocol
3	UDP	User Datagram Protocol
4	DOS	Denial of Service
5	ICMP	Internet Control Message Protocol
6	RBAC	Role-Based Access Control
7	SNMP	Simple Network Management Protocol
8	IDS	Intrusion Detection System
9	RF	Random Forest
10	LSTM	Long Short-Term Memory

INTRODUCTION

1. Purpose of doing thesis

In the current era of digital transformation, network infrastructures have become the fundamental backbone for organizations. However, as network connectivity expands, cybersecurity threats have evolved from simplistic, signature-based attacks to highly sophisticated, evolving, and unpredictable threats. Traditional intrusion detection systems often struggle to keep pace with these challenges.

This thesis was initiated to address these critical security gaps. By leveraging artificial intelligence, the project aims to move from traditional, reactive monitoring to proactive, automated network security. The core motivation is to provide a solution that not only monitors network integrity in real-time but also employs intelligent inference to detect anomalies, thus reducing the burden on human administrators.

2. Scope and Objectives

The primary objective of this project is to build an intelligent, integrated Intrusion Detection System that provides automated network surveillance.

The scope includes:

- **Infrastructure Simulation:** Establishing a controlled network environment using GNS3.
- **Data Ingestion:** Developing an automated pipeline to collect and process real-time network metrics via Zabbix.
- **AI-Powered Analysis:** Implementing an XGBoost machine learning model to accurately classify traffic patterns and identify anomalies.
- **System Management:** Designing a backend with FastAPI to handle core logic, data integrity, and secure authentication.
- **User Management:** Implementing a comprehensive system for administrators to manage jobs, user roles, and security protocols, supported by a dynamic, real-time dashboard.

Research Scope

The research focuses on network anomaly detection in a simulated environment using GNS3. The study is limited to the detection of ICMP Flood, SYN Flood and UDP Flood based on network telemetry data collected through Zabbix. Intrusion prevention mechanisms and application-layer attacks are beyond the scope of this project.

Research Subject

The research subject of this thesis is an AI-based Intrusion Detection System for real-time network traffic monitoring and anomaly detection. The study investigates the application of the XGBoost machine learning algorithm for classifying network traffic based on telemetry metrics collected from network devices.

3. Research Methodology

Literature Review Method: Research papers, technical documentation, and related studies on intrusion detection systems, network monitoring, and machine learning were reviewed to establish the theoretical foundation of the project.

Analytical Method: Network traffic characteristics, attack behaviors, and system requirements were analyzed to determine suitable features and system architecture for anomaly detection.

Experimental Method: The proposed system was implemented and evaluated in a simulated GNS3 environment. Different attack scenarios were conducted to assess the effectiveness of the AI-based detection mechanism.

Modeling Method: System architecture diagrams and database schemas were designed to represent the interaction between system components and data processing pipelines.

4. Graduation project structure

This thesis is organized into four chapters as follows:

- **Introduction:** Presents the background, objectives, significance of the study, technologies and tools used, and the expected outcomes of the proposed system.
- **Chapter 1** Theories and Technologies: Reviews the theoretical foundations related to intrusion detection systems, network security threats, network telemetry, and the XGBoost machine learning algorithm applied in this study.

- **Chapter 2** System Analysis and Design: Analyzes the functional and non-functional requirements of the system and presents the overall architecture, database design, network topology, user interfaces, and system design diagrams.
- **Chapter 3** Implementation and Evaluation: Describes the implementation environment, data collection process, model training procedure, and system deployment. This chapter also presents the experimental results and evaluation of the proposed system under different network traffic scenarios.
- **Conclusion**: Summarizes the achievements and limitations of the project and discusses potential directions for future improvements and research.
- **References:** Lists all resources referenced throughout the thesis.

Chapter 1 : THEORIES AND TECHNOLOGIES

Chapter 1 presents the fundamental theoretical background required for this study, including IDS concepts, common network attack types, and the XGBoost machine learning algorithm used for network traffic classification.

1.1. System Architecture Overview

The system architecture is structured as a modern layered architecture. Establishing a seamless integration between a simulated network infrastructure, real-time monitoring utilities, a machine learning inference core, and a web application architecture.

- **GNS3:** Serving as the primary network emulation platform for constructing and simulating the Local Area Network topology used in this study.
- **Zabbix Server:** Operates as the centralized telemetry system. Zabbix proactively polls network metrics from the Cisco Core Router via SNMP at strict real-time intervals.
- **XGBoost:** Programmed in Python, this algorithm serves as the primary machine learning inference core.
- **FastAPI:** A modern Python web framework used to manage backend operations, handle API requests, and coordinate communication between system components.
- **JSON Web Tokens & bcrypt:** Used to implement secure user authentication, password protection, and RBAC for administrative functions.
- **MongoDB:** A NoSQL database used to store network monitoring data, security alerts, and user account information.
- **Interactive Web Interface (HTML/JS + CSS):** A web-based dashboard that provides user authentication and real-time security alert monitoring.

1.2. Theoretical Foundations of Intrusion Detection Systems

1.2.1. Concept and Architectural Classification of IDS

A system for network surveillance and anomaly detection is a critical component in modern cybersecurity architecture. Unlike a Firewall, which primarily acts as a barrier controlling traffic based on static rules (e.g., blocking a specific port or IP range), this operates as an intelligent surveillance system. It performs Deep Packet Inspection and analyzes traffic behavior to detect malicious activities, security policy violations, or unauthorized access attempts.

In network environments, IDS architecture is generally divided into two main deployment types:

HIDS (Host-based IDS): Inspects data originating from the host system and audit sources, such as operating system logs, Windows event logs, firewall logs, application audit logs, and database logs. HIDS can detect insider attacks that do not involve network traffic. [1]

NIDS (Network-based IDS): Monitors network traffic collected through packet capture, NetFlow, or other network data sources to detect malicious activities before they spread across the network. [1]

1.2.2. The Emergence of AI in Intrusion Detection

The limitations of traditional signature-based and threshold-based intrusion detection systems have driven the adoption of AI and ML techniques in modern cybersecurity. Unlike conventional approaches that rely on manually defined rules, ML-based intrusion detection systems automatically learn patterns and relationships from historical network data, enabling them to identify both known attacks and previously unseen anomalies.

The effectiveness of machine learning-based intrusion detection systems has been widely demonstrated on benchmark datasets such as KDD Cup 99 **and** CICIDS2017, which provide labeled network traffic containing both normal behavior and various attack categories including DoS, Probe, and Brute Force attacks. These datasets are commonly used as standard benchmarks for evaluating anomaly detection models in cybersecurity research. [2][3]

By analyzing the correlation among multiple network features simultaneously, these models can effectively distinguish legitimate traffic fluctuations from malicious activities such as Denial-of-Service attacks. In addition, previous studies have shown that anomaly detection techniques play a crucial role in identifying unknown or zero-day attacks that cannot be detected using rule-based systems. [4]

1.3. Analysis of Cyberattack Vectors

1.3.1. ICMP Flood

The Internet Control Message Protocol (ICMP), which is utilized in a Ping Flood attack, is an Internet layer protocol used by network devices to communicate. The network diagnostic tools traceroute and ping both operate using ICMP. [5]

An ICMP Flood, also known as a Ping Flood attack, is a type of Denial-of-Service (DoS) attack that exploits this functionality by sending an excessive

number of ICMP Echo Request packets to a target system. For every request received, the target must allocate processing resources and network bandwidth to generate and transmit a corresponding Echo Reply packet. When the volume of requests exceeds the system's processing capacity, CPU utilization, memory consumption, and network bandwidth may become saturated, significantly degrading service performance or rendering the target unavailable to legitimate users.

The DDoS form of a Ping (ICMP) Flood can be broken down into 2 repeating steps:

- The attacker sends many ICMP echo request packets to the targeted server using multiple devices.
- The targeted server then sends an ICMP echo reply packet to each requesting device's IP address as a response.

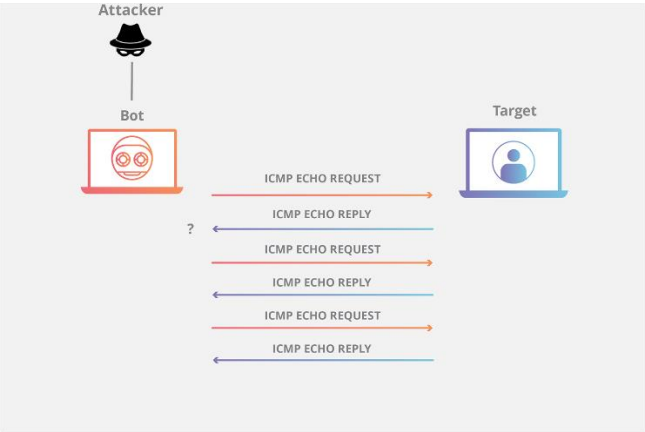


Figure 1-1: Ping ICMP DDoS Attack Diagram

1.3.2. SYN Flood

A SYN flood (half-open attack) is a type of denial-of-service (DDoS) attack that aims to make a server unavailable to legitimate traffic by consuming all available server resources. By repeatedly sending initial connection requests (SYN) packets, the attacker is able to overwhelm all available ports on a targeted server machine, causing the targeted device to respond to legitimate traffic sluggishly or not at all. [6]

- First, the client sends a SYN packet to the server in order to initiate the connection.
- The server then responds to that initial packet with a SYN/ACK packet in order to acknowledge the communication.

- Finally, the client returns an ACK packet to acknowledge the receipt of the packet from the server. After completing this sequence of packet sending and receiving, the TCP connection is open and able to send and receive data.

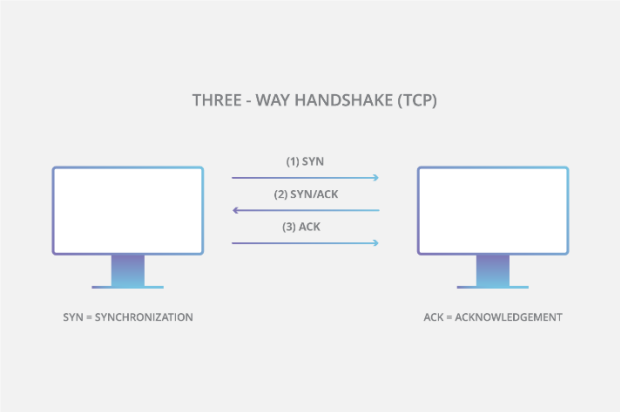


Figure 1-2: TCP Three-way handshake diagram

1.3.3. UDP Flood

A UDP flood is a type of denial-of-service attack in which a large number of User Datagram Protocol (UDP) packets are sent to a targeted server with the aim of overwhelming that device's ability to process and respond. The firewall protecting the targeted server can also become exhausted as a result of UDP flooding, resulting in a denial-of-service to legitimate traffic. [7]

1.4. Theoretical Foundations of the XGBoost Classification Algorithm

Traditional models like decision trees and random forests are easy to interpret but may lack accuracy on complex data. XGBoost (eXtreme Gradient Boosting) is an optimized gradient boosting algorithm that combines multiple weak models into a stronger, high-performance model. [8]

1.4.1. Decision Tree – The Base Unit

XGBoost uses decision trees as base learners, building them sequentially so each tree corrects errors from the previous one, and it is known as boosting.

1.4.2. How does XGBoost work?

It builds decision trees sequentially, with each tree attempting to correct the mistakes made by the previous one. The process can be broken down as follows:

1. Start with a base learner: The first model decision tree is trained on the data. In regression tasks, this base model simply predicts the average of the target variable.

2. Calculate the errors: After training the first tree, the errors between the predicted and actual values are calculated.

3. Train the next tree: The next tree is trained on the errors of the previous tree. This step attempts to correct the errors made by the first tree.

4. Repeat the process: This process continues with each new tree, trying to correct the errors of the previous trees until a stopping criterion is met.

5. Combine the predictions: The final prediction is the sum of the predictions from all the trees.

1.4.3. Mathematics Behind the XGBoost Algorithm

It can be viewed as an iterative process where we start with an initial prediction often set to zero. After which, each tree is added to reduce errors. Mathematically, the model can be represented as [9]:

$$\hat{\gamma}_i = \sum_{\kappa=1}^{\mathrm{K}} f_\kappa (x_i) \qquad \textit{(1.1) Formula for representing the model}$$

Where:

- $\hat{y}_i$ is the final predicted value for the i^{th} data point
- K is the number of trees in the ensemble
- $f_k(x_i)$ represents the prediction of the K^{th}tree for the i^{th} data point.

The general form of the loss function is:

$$obj(\theta) = \sum_i^n l\,(y_i, \hat{y}_i) + \sum_{\kappa=1}^{\mathrm{K}} \Omega(f_k) \quad \textit{(1.2) Formula for loss function}$$

Where:

- $l(y_i, \hat{y}_i)$ is the loss function which computes the difference between the true value y_i and the predicted value $\hat{y}_i$,
- $\Omega(f_k)$is the regularization term, which discourages overly complex trees.

1.4.4. Why XGBoost?

This section presents a comparative analysis of three machine learning models—Random Forest, Long Short-Term Memory , and XGBoost—based on experiments conducted on the same dataset and feature set extracted from network telemetry data. Each model was trained and evaluated under identical conditions using an 80/20 train-test split to ensure a fair comparison in terms of predictive performance and computational efficiency.

In addition to classification accuracy, a critical evaluation metric considered in this study is the Average Inference Time (ms/sample)**,** which represents the average time required for a trained model to process a single input sample and generate a prediction. This metric is particularly important for real-time intrusion detection systems, where the model must operate within strict time constraints.

The Average Inference Time is computed as:

- First, the total inference time is measured by recording the execution time required for the model to predict all samples in the test set.
- Then, this total time is divided by the number of test samples to obtain the per-sample inference latency.
- Finally, the result is converted into milliseconds (ms) for interpretability:

$$\text{Average Inference Time} = \frac{T_{total}}{N} \times 1000 \text{ (ms/sample)} \quad (1.3)$$

Formula for Average Inference Time

where:

- T_{total}: is the total prediction time for the test set (in seconds).
- N: is the number of test samples.

This metric reflects the efficiency of each model in real-time deployment scenarios, where lower inference latency indicates better suitability for continuous network monitoring.

Table 1-1: Comparative Evaluation of RF, LSTM, and XGBoost

Criteria	Random Forest	Long Short-Term Memory	XGBoost
Suitability for tabular data	Tree-based methods generally outperform deep learning models on tabular datasets [10].	Primarily designed for sequential data; not naturally optimized to extract features from simple tabular inputs. [11]	Widely recognized as highly effective for tabular data [9]
Accuracy (%)	99.11	98.28	99.19
Average Inference Time (ms/sample)	0.0088	0.1782	0.0020

As shown in Table 1.1, the selection of the machine learning model depends on both data structure and system requirements. LSTM is designed for sequential data to capture temporal dependencies, while the proposed system uses 4-dimensional tabular telemetry snapshots collected every 5 seconds. This mismatch makes LSTM less suitable, as it introduces higher computational cost without clear performance benefits for the given data format.

In addition, the experimental results show that XGBoost achieves the highest accuracy (99.19%) while also providing the lowest average inference time (0.0020 ms/sample) among the evaluated models. This makes it more suitable for

real-time intrusion detection. Therefore, XGBoost offers a better balance between accuracy and computational efficiency compared to Random Forest for the given network telemetry classification task.

In summary, this chapter presented the key theoretical foundations of network security, telemetry systems, and the XGBoost model. These concepts provide a basis for the system design and implementation discussed in the following chapters.

Chapter 2 : SYSTEM ANALYSIS AND DESIGN

Chapter 2 describes the analysis and design of the proposed AI-based IDS, including system architecture, network topology, functional requirements, database design, and user interface components. It also defines user roles, system interaction flows and network topology.

2.1. Descriptive overview of objects

To ensure strict security and compliance with network management standards, the system implements a Role-Based Access Control architecture. The system categorizes users into three main functional groups:

- **Root (Super Administrator):** The entity with the absolute highest privileges in the system. The Root account is typically created during the initial system bootstrap. This role is responsible for the overall security of the application, managing all lower-level Administrator accounts, and overseeing the entire database architecture and system configuration.
- **Administrator:** The core operator of the system. Administrators are responsible for active network surveillance. They can view real-time traffic, analyze AI inference logs and manage lower-tier monitoring accounts.
- **Monitor:** The lowest privilege tier, designed for shift analysts or guest viewers. Once authenticated, a Monitor can view the real-time streaming dashboard, observe network metrics, and read AI detection alerts.

2.2. Functional Description

2.2.1. For Root (Super Administrator)

- **Authentication:** Must securely log in using encrypted credentials (JWT and bcrypt) to access the Superadmin dashboard.
- **Global User Management:**
  - Has full CRUD (Create, Read, Update, Delete) access to all user accounts in the system.
  - Can grant, upgrade, or revoke "Admin" privileges.

2.2.2. For Administrator

- **Login & Authentication:** Must log in with a valid Admin account to access active management features.
- **Dashboard & Real-time Surveillance:**
  - Can view real-time charts displaying network metrics ingested from the Zabbix Server (e.g., bandwidth, latency, packet rates).

- **Threat Intelligence & AI Logs:**
  - Can view the historical logs of network states analyzed by the XGBoost AI Engine.
- **Sub-User Management:**
  - Can view, register, edit, or block users assigned to the "Monitor" role.

2.2.3. For Monitor (Security Analyst)

- **Login:** Must log in to access the system; unauthorized access is entirely blocked by backend middleware.
- **Profile Management:**
  - Can view and update basic personal profile information and change passwords.
- **Dashboard Observation (Read-Only):**
  - Can monitor the real-time network traffic graphs.
  - Can view the current status of the network (Normal, ICMP_Flood, SYN Flood, UDP_Flood) as inferred by the AI model.

2.3. Use-case diagram

2.3.1. General

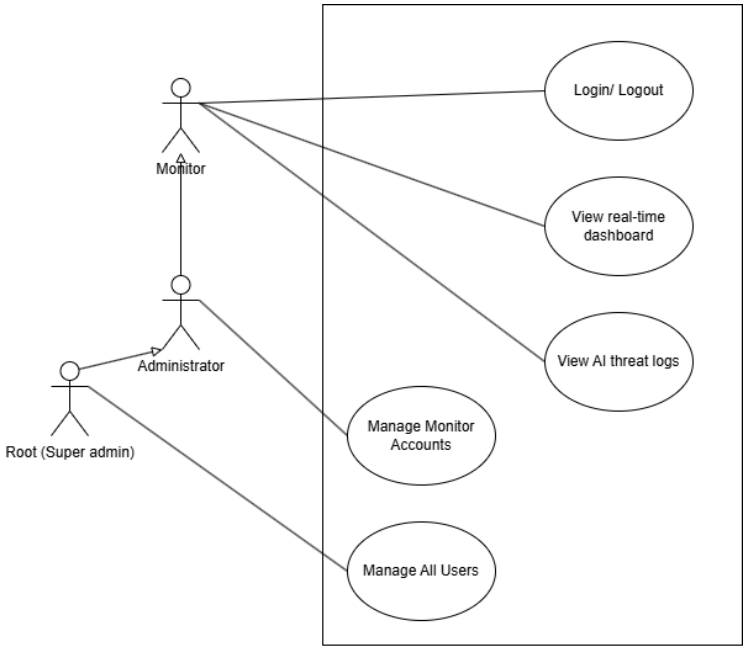


Figure 2-1: General Use-case diagram

Table 2-1: General Use-case description

Use case	Input	Output	Flow
Login /	Username and	User is	1. The user enters login

Logout	password, or an active user session	authenticated or logged out successfully	credentials or requests logout. 2. The system validates the credentials or terminates the current session. 3. The user is granted access or redirected to the login page.
View Real-time Dashboard	Authenticated user request	Real-time network telemetry dashboard	1. The user opens the dashboard. 2. The system retrieves the latest network metrics. 3. The dashboard displays updated network statistics and anomaly status.
View AI Threat Logs	Authenticated user request	Historical AI-detected anomaly logs	1. The user navigates to the threat log page. 2. The system retrieves stored detection records. 3. The logs are displayed in chronological order.
Manage All Users	User management request	User account is created, updated, or deleted	1. The Root selects the user management function. 2. The system verifies authorization. 3. The Root creates a new account, updates a user's name or role, or deletes an existing account. 4. The system saves the changes.

2.3.2. Root (Super Administrator)

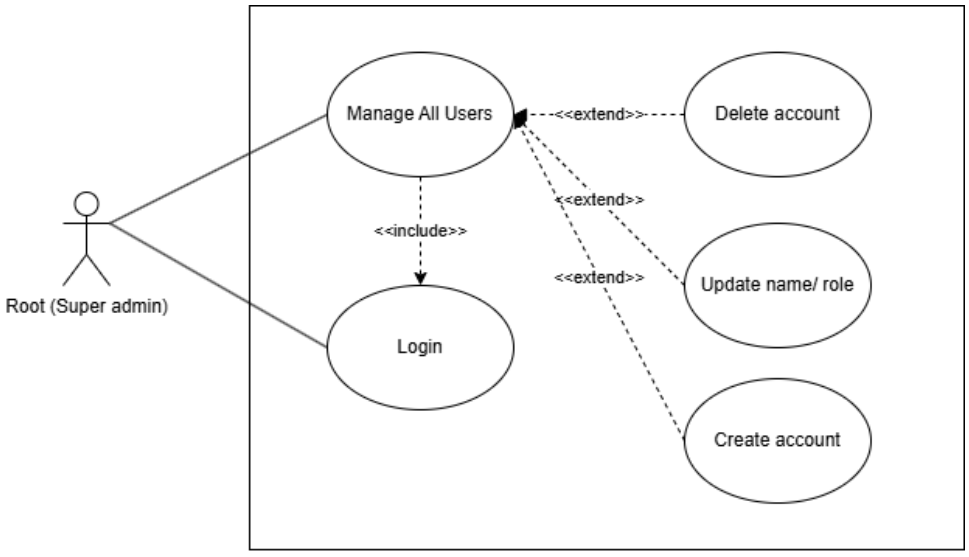


Figure 2-2: Use-case for Super Administrator's features

Table 2-2: Use-case description for Super Administrator

Use case	Input	Output	Flow
Login	Root username and password	Successful authentication or error message	1. Root enters credentials. 2. The system validates the credentials. 3. If valid, the system grants access; otherwise, an error message is displayed.
Manage All Users	User management request (create, update, or delete)	User account information is updated	1. Root selects **Manage All Users**. 2. The system requires successful login (include). 3. Root chooses to create an account, update a user's name or role, or delete an account (extend). 4. The system validates the request and updates the database.
Create Account	New user's information (username, password, role)	New user account is created	1. Root enters the new user's information. 2. The system validates the input. 3. A new account is created and stored in the database.
Update User	Existing username and updated name or role	User information is updated	1. Root selects a user account. 2. Root modifies the user's name or role. 3. The system validates and saves the changes.
Delete Account	Target user account	User account is permanently deleted	1. Root selects a user account. 2. The system requests confirmation. 3. After confirmation, the selected account, including an Administrator account if applicable, is permanently removed.

2.3.3. Administrator

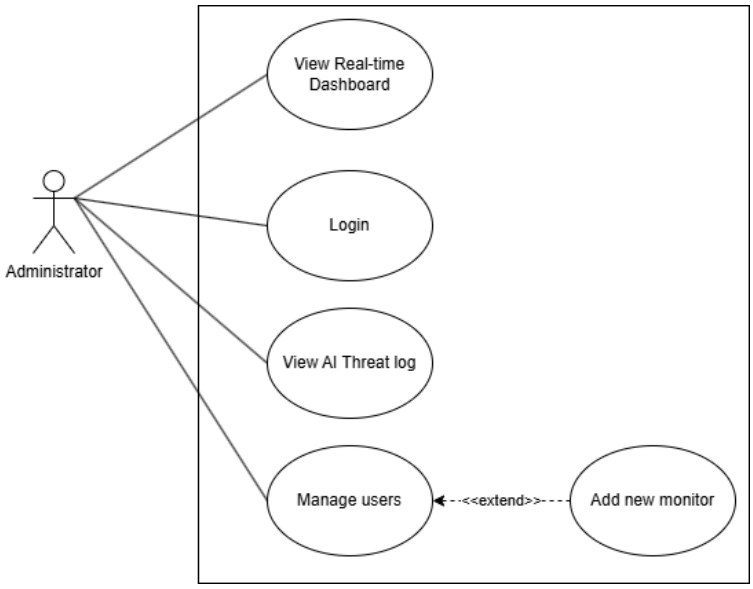


Figure 2-3: Use-case for Admin's features

The Administrator is responsible for monitoring network activities and managing Monitor accounts. To access the system, the Administrator must first log in. After successful authentication, the Administrator can view the real-time network dashboard and AI threat logs. Additionally, the Administrator can manage Monitor accounts by creating new Monitor users.

2.3.4. Monitor

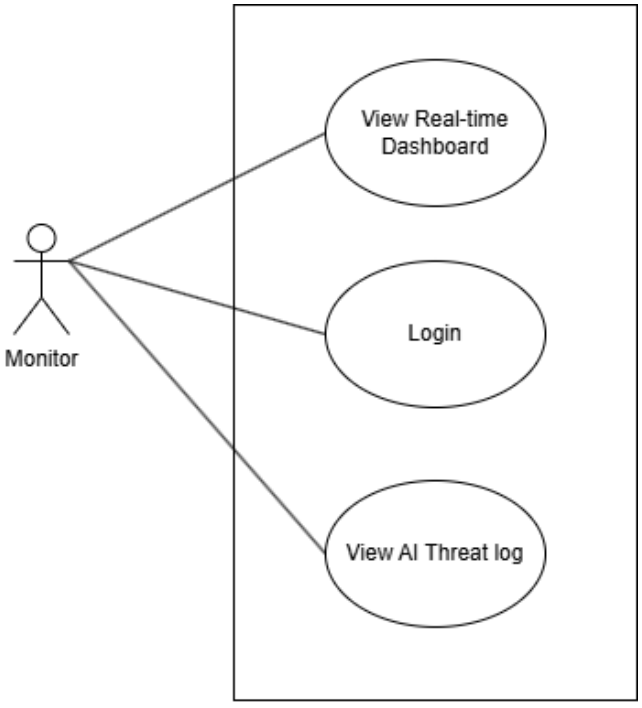


Figure 2-4: Use-case for Monitor's features

The Monitor is responsible for monitoring network activities through the platform. To access the system, the Monitor must first log in. After successful authentication, the Monitor can view the real-time network dashboard, access AI-generated threat logs, and analyze historical network data through the analytics dashboard. When monitoring tasks are completed, the Monitor can log out to end the current session.

2.4. Activity diagram

Real-Time Network Monitoring and Threat Detection

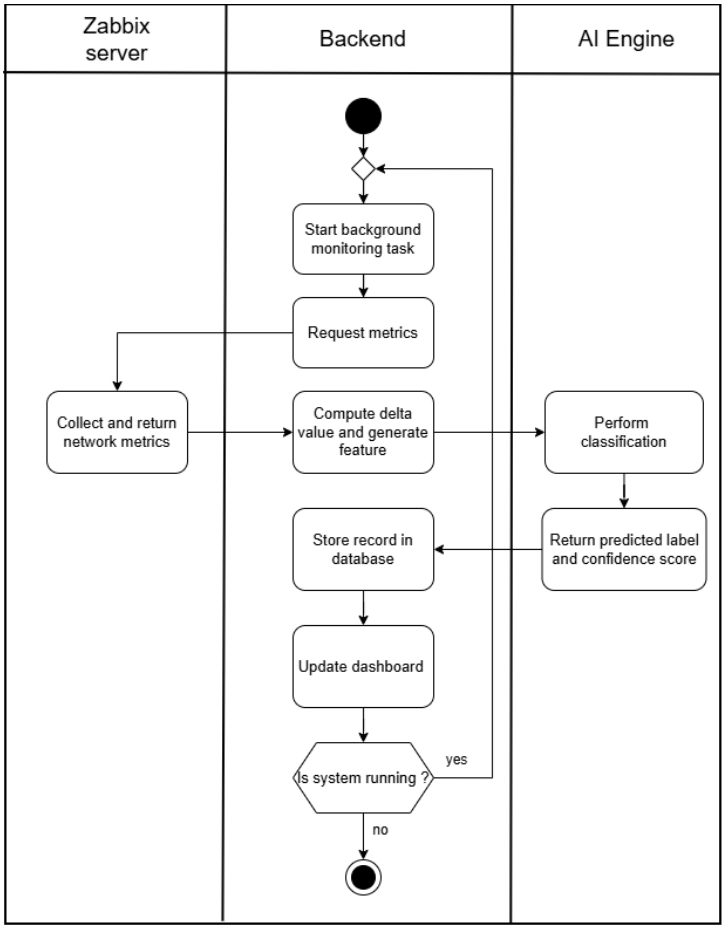


Figure 2-5: Activity diagram for Network Monitoring and Threat Detection

Figure 2-5 illustrates the activity diagram of the AI-based network monitoring and anomaly detection workflow.

2.5. Entity diagram

2.5.1. Table overview

Table 2-3: Table overview

No.	Table name	Short Description
1	Users	Stores user account information, hashed passwords,

		and role-based access control data
2	Live_status	Maintains a single record representing the current network status to support real-time dashboard synchronization
3	History_log	Stores time-series historical network traffic data and AI prediction results for analysis and traceability purposes

2.5.2. Entity diagram

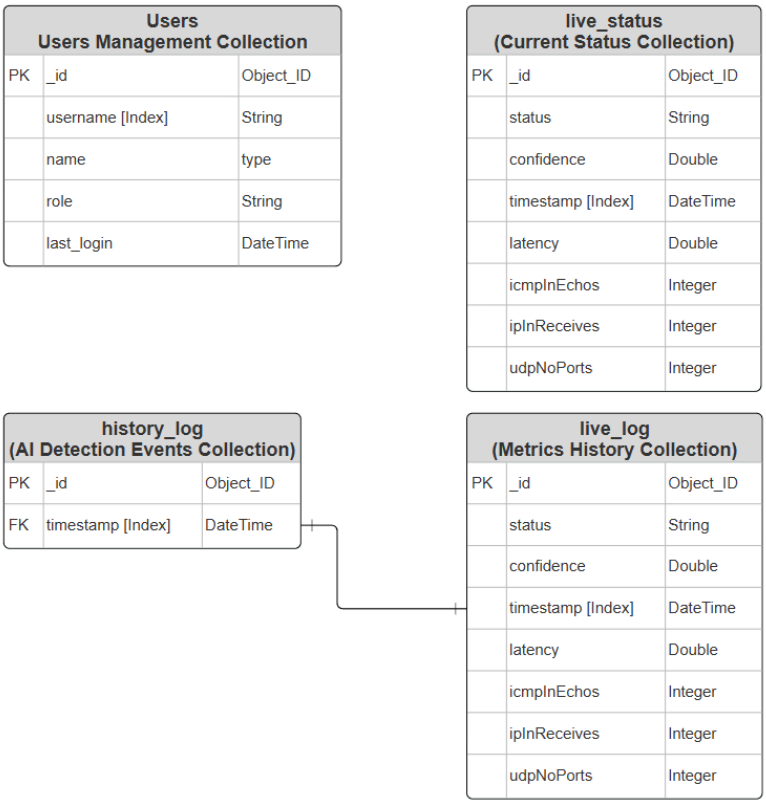


Figure 2-6: Entity diagram

2.5.3. Entity details

❖ **Users**

Table 2-4: Users entity details

Attributes	Datatype	Null	Key	Default
_id	ObjectId	No	PK	Auto
username	String	No	-	-
hashed_password	String	No	-	-
role	String	No	-	-

last_login	DateTime	No	-	-

❖ ***Live_status***

Table 2-5: Live_status entity details

Attributes	**Datatype**	**Null**	**Key**	**Default**
_id	ObjectId	No	PK	Auto
status	String	No	-	-
confidence	Double	No	-	-
timestamp	DateTime	No	-	-
latency	Double	No	-	-
icmpInEchos	Integer	No	-	-
ipInReceives	Integer	No	-	-
udpNoPorts	Integer	No	-	-

❖ ***History_log***

Table 2-6: History_log entity details

Attributes	**Datatype**	**Null**	**Key**	**Default**
_id	ObjectId	No	PK	Auto
timestamp	DateTime	No	FK	-

❖ ***Live_log***

Table 2-7: Live_log entity details

Attributes	**Datatype**	**Null**	**Key**	**Default**
_id	ObjectId	No	PK	Auto
status	String	No	-	-
confidence	Double	No	-	-
timestamp	DateTime	No	-	-

latency	Double	No	-	-
icmpInEchos	Integer	No	-	-
ipInReceives	Integer	No	-	-
udpNoPorts	Integer	No	-	-

2.6. Network Topology and Infrastructure Design

The simulated corporate network infrastructure comprises of four primary logical zones:

- Core Routing Node: This is the central network element under continuous monitoring (typically a virtualized Cisco IOS router instance). It handles all routing choices between subnets. Crucially, its operating system is configured with SNMPv2c/v3 enabled on all active interfaces. It acts as the primary data point, generating raw telemetry metrics such as packet queue lengths, inbound/outbound octet deltas, and processing delays.
- The Attacker Node: An isolated subnet hosting a specialized security testing node (Linux-based instance). This node is pre-installed with network penetration toolkits, primarily hping3 and nmap. It is programmatically controlled by the FastAPI backend to execute precise, reproducible traffic stress scripts corresponding to the four attack classes: ICMP floods, UDP floods, TCP SYN floods.
- Management & Monitoring Node (Zabbix): A centralized Linux instance running the Zabbix Server daemon. It is connected via a dedicated out-of-band management subnet to the Core Routing Node. It continuously polls the router's management information bases (MIBs) via SNMP on UDP Port 161 to log real-time interface rates.

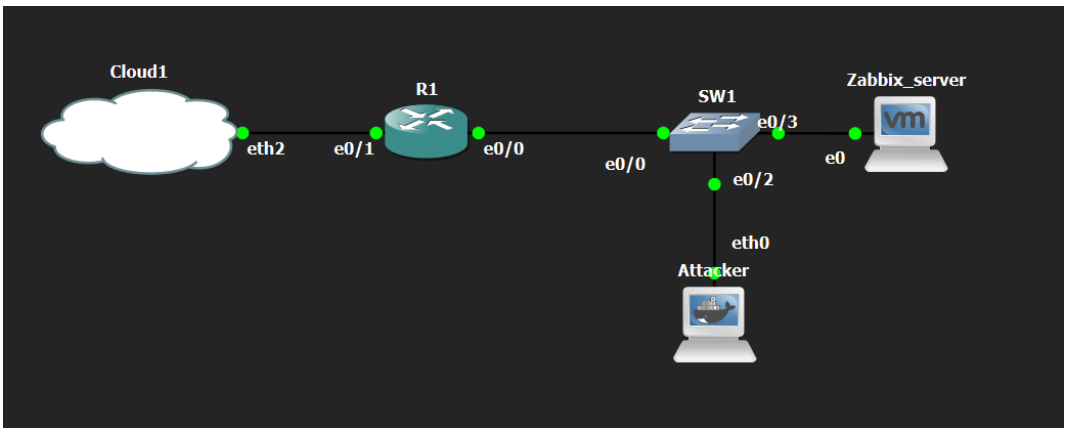


Figure 2-7: Network topology in GNS3

2.6. Architecture diagram

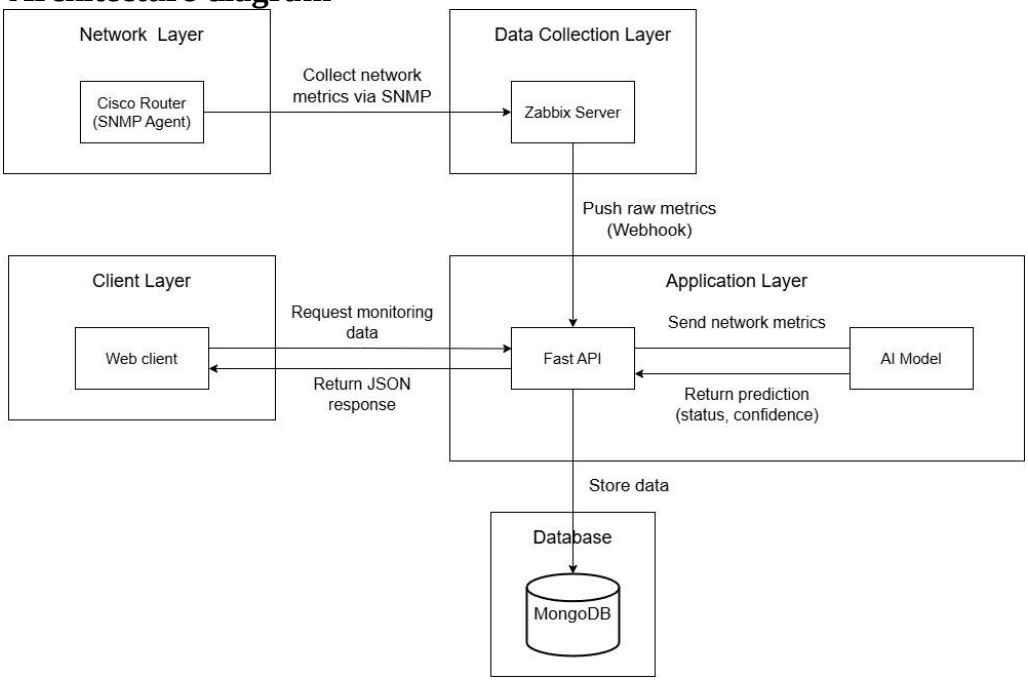


Figure 2-8: Architecture diagram

Figure 2-8 illustrates the layered architecture of the proposed system. The system is organized into five functional layers that operate in a complete end-to-end processing pipeline, as described below.

- Network Layer: This layer consists of the physical network devices, such as Cisco routers, which act as SNMP agents. These devices continuously generate network performance metrics.
- Data Collection Layer: The Zabbix Server periodically collects raw network metrics from the routers using SNMP. After data collection, Zabbix forwards the monitoring data to the backend through a Webhook mechanism.
- Application Layer: This layer serves as the core processing unit of the system. Upon receiving raw metrics from Zabbix, the FastAPI backend forwards the data to the AI Model for anomaly detection. The AI model analyzes the network data, computes the confidence score, and returns the prediction results, including the network status and confidence value. The backend then processes these results and prepares them for storage and client requests.
- Database Layer: After the anomaly detection process is completed, the FastAPI backend stores both the collected network metrics and the prediction results in the MongoDB database. The database maintains real-time monitoring information, historical detection records, and logging data for future analysis and visualization.
- Client Layer: The web client communicates with the FastAPI backend through HTTP APIs to request monitoring data. The backend responds

with JSON-formatted data, which the frontend uses to render interactive dashboards, charts, and real-time network status for system administrators.

2.7. User interface

2.7.1. Login

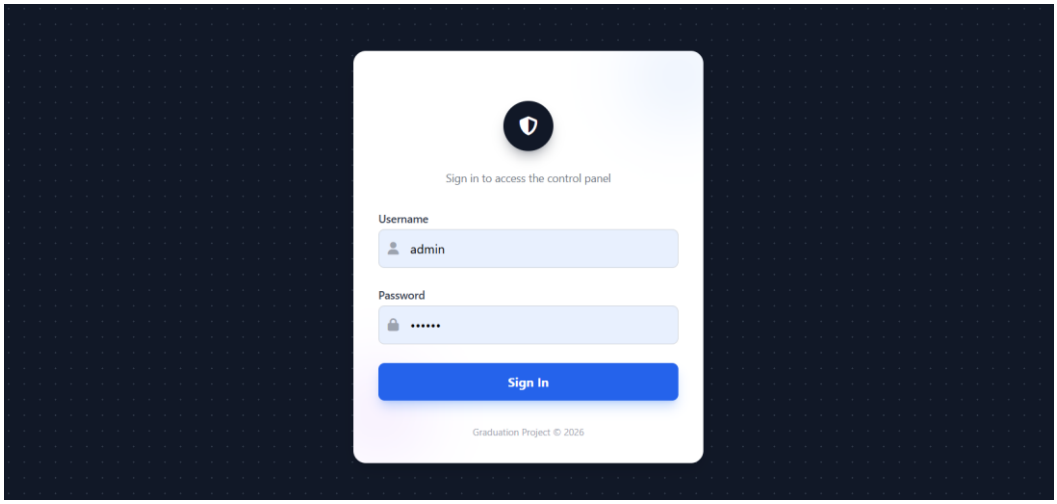


Figure 2-9: Login Page UI

The login interface allows users to authenticate into the system by entering their credentials before accessing the monitoring dashboard and system functionalities.

2.7.2. Real-time dashboard

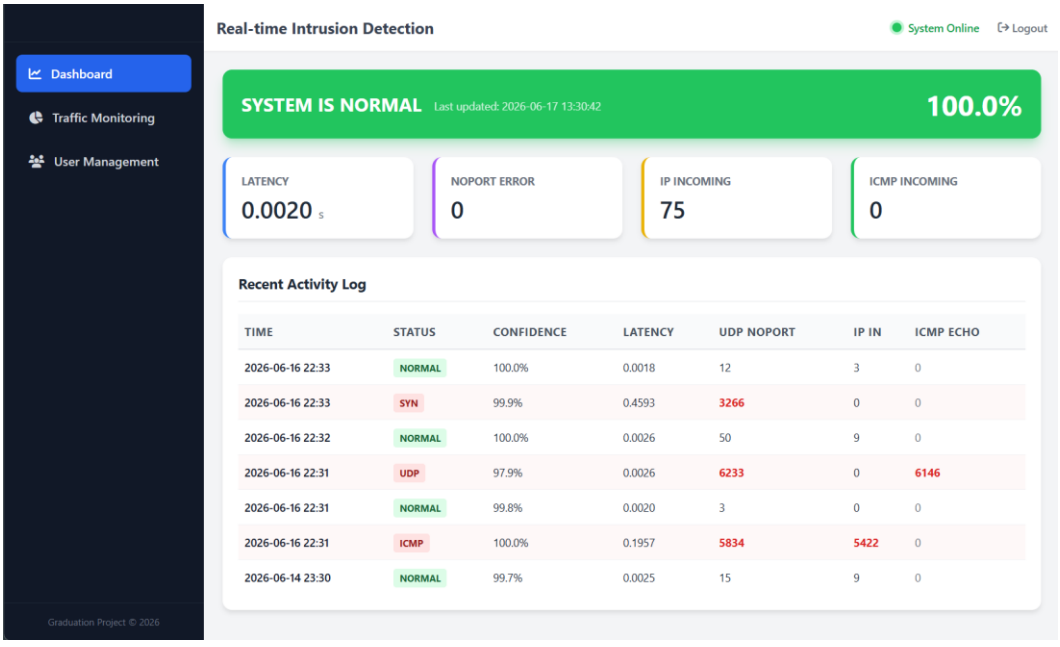


Figure 2-10: Real-time dashboard Page

The figure illustrates the main real-time dashboard of the system. From this interface, users can view AI-based traffic predictions, system logs, and historical

detection results. The navigation panel on the right provides access to additional modules, including Traffic Monitoring and User Management.

2.7.3. Traffic Monitoring

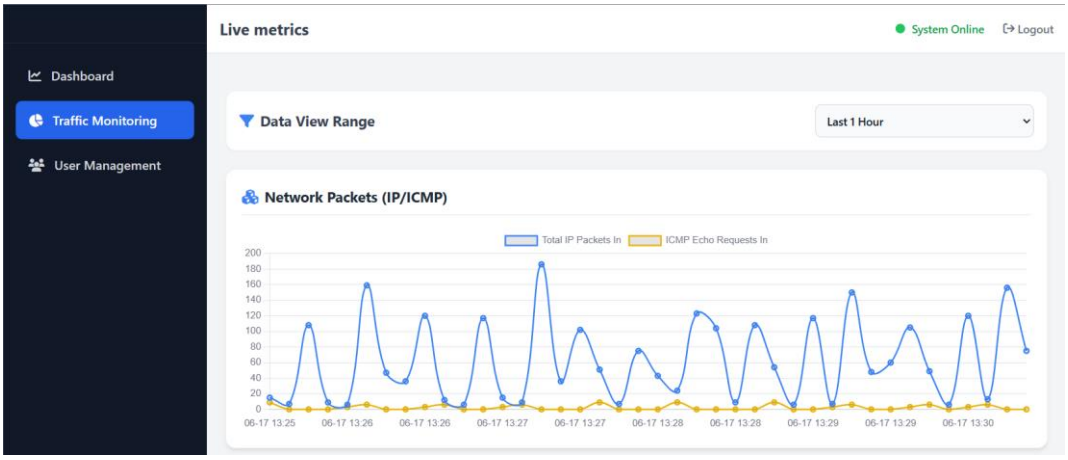


Figure 2-11: Network Packet Visualization in Traffic Monitoring Interface

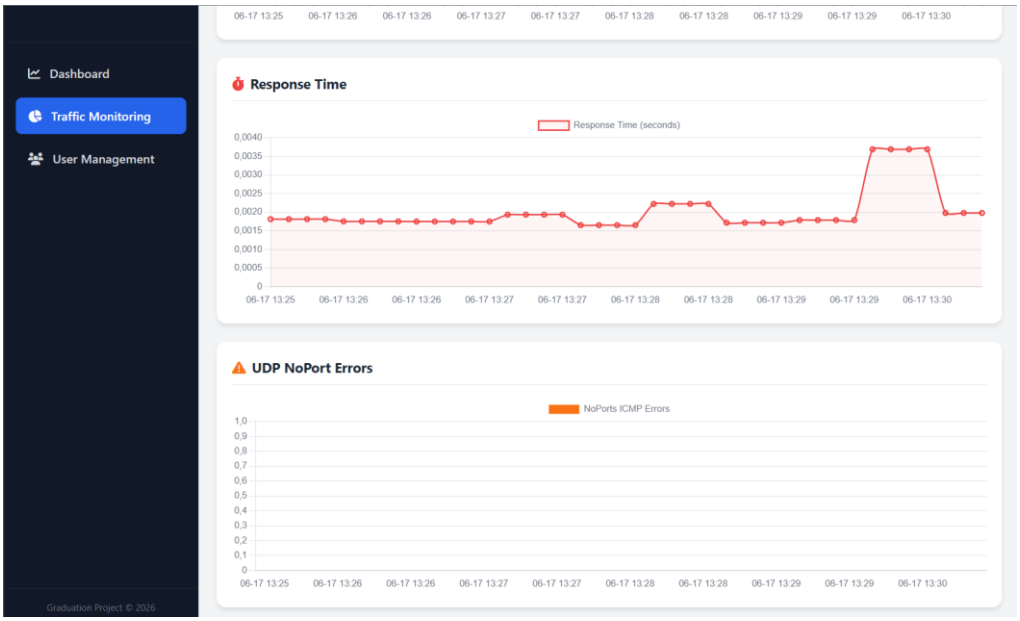


Figure 2-12: Response Time and UDP Traffic to Closed Ports Visualization

Figures 2-11 and 2-12 illustrate the traffic monitoring interface of the system, which visualizes network packet statistics, including IP and ICMP traffic to the router, UDP packets sent to closed ports, and the router response time (latency).

2.7.4. User Management

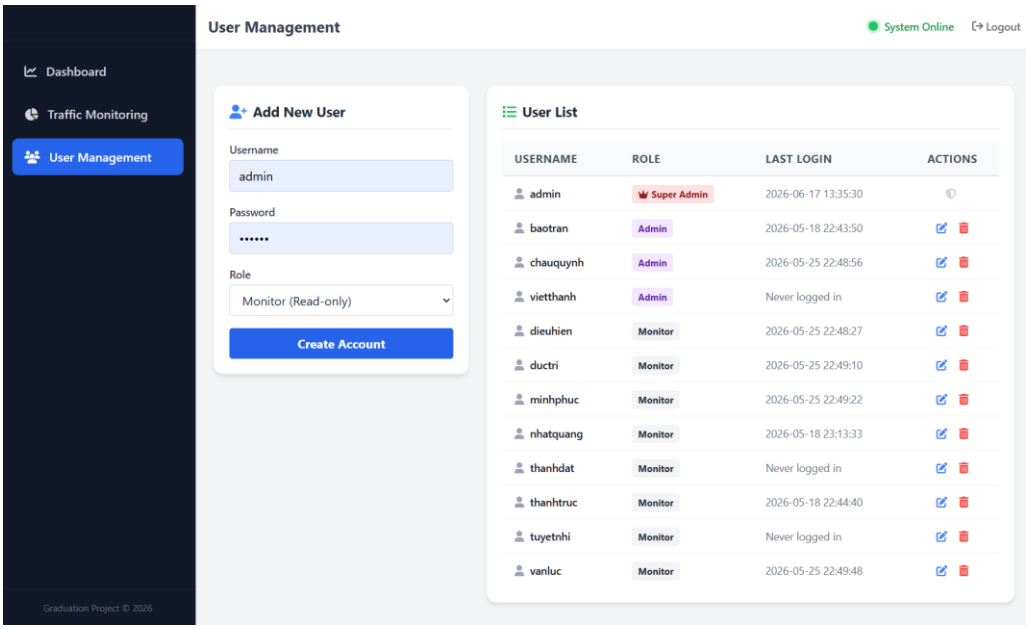


Figure 2-13: User Management Page

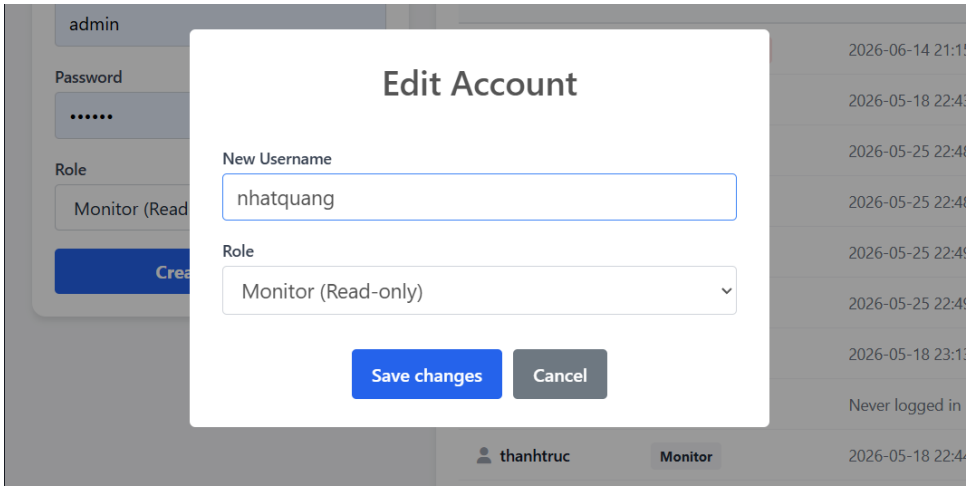


Figure 2-14: Edit User Interface

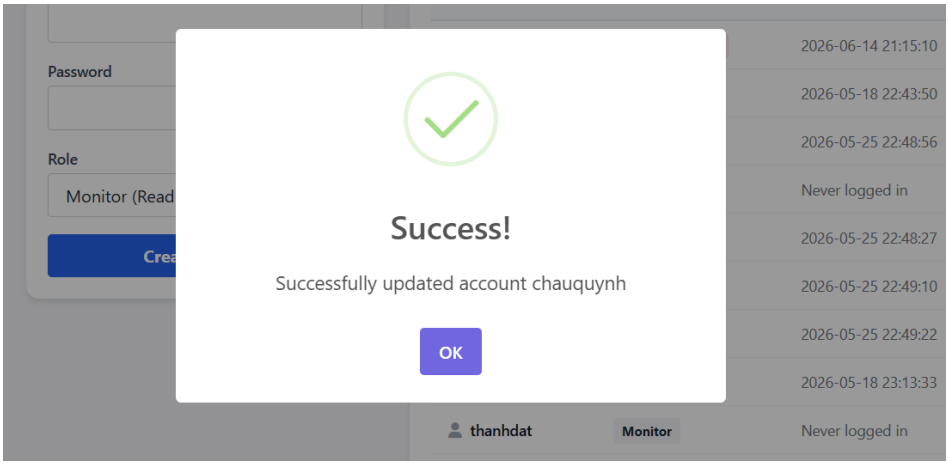


Figure 2-15: User Update Successful Notification

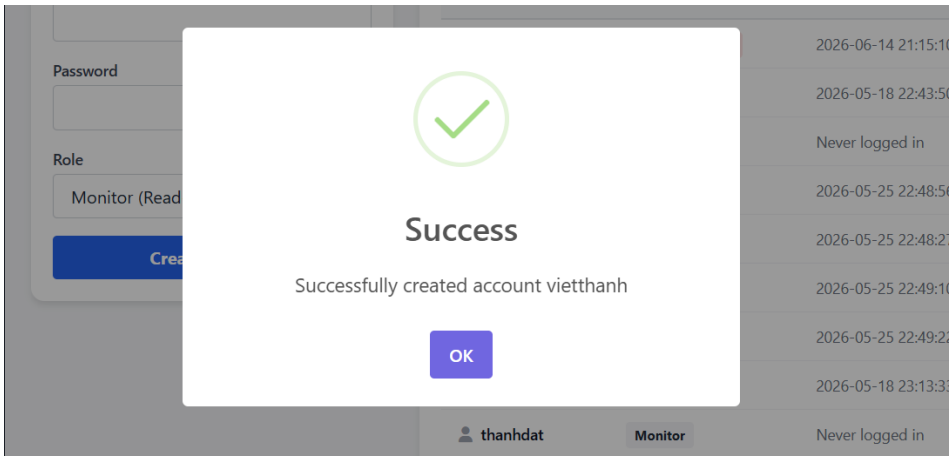


Figure 2-16 : User Creation Successful Notification

The User Management interface allows administrators to manage system users, including creating new accounts, updating existing user information, and viewing user details. This module provides full control over user access and roles within the system.

2.8. Security & System

2.8.1. Potential attack vectors

To ensure that the proposed network monitoring system does not become a target for cyber attacks, two critical communication channels were analyzed to identify potential security risks.

- Manipulation of monitoring data: Since Zabbix collects network metrics from Cisco routers using the unencrypted SNMPv2c protocol, an attacker with access to the internal network may intercept packets to obtain the SNMP community string. The attacker could then inject spoofed monitoring data, causing the AI model to produce incorrect anomaly detection results.
- Webhook and API Attacks: If the FastAPI endpoints are not adequately protected, attackers may exploit them by launching DDoS attacks, performing brute-force login attempts, or submitting malicious JSON payloads to compromise the MongoDB database or disrupt backend services.

2.8.2. Authentication Flow

The system employs a stateless authentication mechanism based on JSON Web Tokens (JWT) combined with the HTTP Bearer authentication scheme to implement Role-Based Access Control (RBAC).

The authentication process consists of the following four steps:

- **User Login:** The user enters a username and password through the login interface. The credentials are then sent to the backend via an HTTP POST request to the /login endpoint.
- **Credential Verification:** The FastAPI backend retrieves the corresponding user account from MongoDB and verifies the submitted password using the Bcrypt hashing algorithm.
- **JWT Generation:** If the authentication is successful, the system generates a JWT containing user information such as the username, assigned role, and token expiration time. The token is digitally signed using the HS256 algorithm and a secret key stored securely on the server.
- **Authorized Access:** For subsequent requests to protected API endpoints the client must include the JWT in the Authorization header using the format Bearer. Before processing the request, the FastAPI backend validates the token signature and expiration through the verify_token dependency. Only requests with valid tokens are authorized to access protected resources.

In summary, this chapter presents a complete system design covering all functional and structural components. This architecture serves as a blueprint for the implementation and evaluation phase in Chapter 3.

Chapter 3 : IMPLEMENTATION AND EVALUATION

Chapter 3 focuses on the implementation of the proposed system, including data collection, feature engineering, model training using XGBoost, and network infrastructure setup using GNS3. It also presents evaluation scenarios for different attack types.

3.1. Implementation

3.1.1. Data Collection and Feature Optimization

To construct a highly reliable dataset capable of training the machine learning model under realistic network conditions, a proactive data generation and extraction framework was established:

- **Traffic Generation & Simulation:** Controlled network environments were modeled inside **GNS3**. Specialized network attack scripts utilizing tools such as hping3 and nmap were executed from an attacker node directed at the target virtual routers. Each specific network state was maintained for distinct intervals to allow stable metric logging.
- **Telemetry Aggregation via SNMP:** The virtual routers, configured with Simple Network Management Protocol (SNMP enabled), continuously pushed performance and traffic metrics to the centralized **Zabbix Server**.
- **Automated API Polling:** A custom data pipeline developed in Python engaged the Zabbix API via JSON-RPC requests. The pipeline fetched network metrics sequentially. To capture structural patterns rather than static metrics, the system computed the delta values (rate of change) between consecutive polling intervals.
- **Data Balancing:** To prevent the machine learning model from developing a class-imbalance bias, raw data collection was strictly metered. The sampling process concluded once exactly 4,500 valid samples were logged for each network state, establishing a perfectly balanced dataset containing a total of 18,000 observations.

3.1.2. Input Feature

During the data collection phase, four network features were selected from the Zabbix telemetry logs: latency, icmpInEchos, ipInReceives, and udpNoPorts. These features were chosen because they provide useful information for distinguishing between normal traffic and different attack types. Limiting the dataset to four features also helps reduce computational overhead and simplifies the model training process while maintaining classification performance.

Table 3-1: Description of the input features

No.	Attribute Name	Data Type	Description
1	latency	Float	**Network Latency (Seconds):** Measures the round-trip time deviation. This metric escalates rapidly when a router CPU or interface queue is overwhelmed by floods of traffic.
2	icmpInEchos	Integer	**ICMP Echo Request Packets:** Counts the inbound ping requests. It serves as the primary indicator for isolating **ICMP Flood** attack vectors.
3	ipInReceives	Integer	**Total Inbound IP Packets:** Monitors the global packet intake rate. Essential for calculating volumetric traffic thresholds across all layers.
4	udpNoPorts	Integer	**Inbound UDP Packets to Closed Ports:** Tracks packets destined for unassigned ports. A sudden spike indicates a systematic infrastructure footprint scan via UDP.

3.1.3. Model training

After data preprocessing and feature selection, the XGBoost algorithm was used to train the network traffic classification model.

- The XGBoost model receives network traffic data in tabular format consisting of four input features: latency, icmpInEchos, ipInReceives, and udpNoPorts.
- The output layer was designed as a multi-class classification module. The target variables were one-hot encoded to map the continuous input metrics into one of four distinct categorical network states:

Table 3-2: Distribution of classes in the dataset

Class Label	Network State	Sample Count	Description
NORMAL	Normal	4500	Legitimate network traffic operating under normal conditions
ICMP	ICMP Flood	4,500	Denial-of-Service traffic is generated by a large volume of ICMP Echo Request packets.

SYN	TCP SYN Flood	4,500	TCP SYN Flood traffic that exploits the connection establishment process by creating numerous half-open connections.
UDP	UDP Flood	4,500	Denial-of-Service traffic generated by a high volume of UDP packets sent to target hosts or services.
Total	05 Classes	18,000	

The dataset was divided into training and testing sets using an 80/20 split ratio, ensuring a baseline for evaluating the supervised learning architecture [12]. The training set was used to build the XGBoost model for classifying network traffic into five categories, while the testing set was used to evaluate the model's performance.

The optimized hyperparameters used for training the XGBoost classifier are summarized in Table 3.3.

Table 3-3: Hyperparameters used for training the XGBoost

Parameter	Value	Description
n_estimators	100	Number of boosting trees used to build the ensemble model. A larger value generally improves learning capability but increases training time.
max_depth	6	Maximum depth of each decision tree.
learning_rate	0.1	Step size used to update model weights during boosting.
random_state	42	Random seed used to ensure reproducibility of the experimental results.
reg_alpha	0.1	L1 regularization term that encourages sparse feature weights and helps reduce overfitting.
reg_lambda	1.0	L2 regularization term that penalizes large model weights, improving model stability and generalization.
subsample	0.8	Fraction of training samples randomly selected for each boosting iteration, reducing variance and

		mitigating overfitting.
eval_metric	mlogloss	Multi-class loss function for model evaluation.

The XGBoost model achieved an overall accuracy of 99.19% on the testing set. To further evaluate its classification performance, precision, recall, and F1-score were calculated for each class, as shown in Table 3.4.

Table 3-4: Model training report

Class Label	Precision	Recall	F1-Score
icmp	1.00	0.98	0.99
normal	1.00	1.00	1.00
syn	0.99	0.99	0.99
udp	0.99	0.99	0.99
Accuracy			*0.99*

As shown in Figure 3-1, most samples are correctly classified along the main diagonal, indicating excellent classification performance. Only a small number of samples are misclassified between similar attack categories, demonstrating that the proposed model can effectively distinguish normal traffic from ICMP Flood, TCP SYN Flood, and UDP Flood attacks.

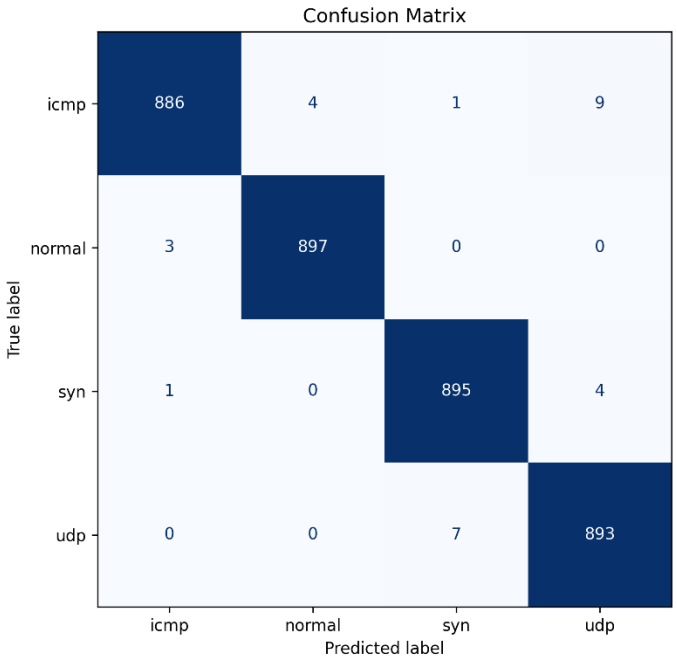


Figure 3-1: Confusion matrix

As illustrated in Figure 3-2, the ROC curves are located close to the upper-left corner of the plot, and the AUC values for all classes are close to 1.0. These results indicate that the XGBoost model achieves good discrimination between normal traffic and different attack types while maintaining a very low classification error rate.

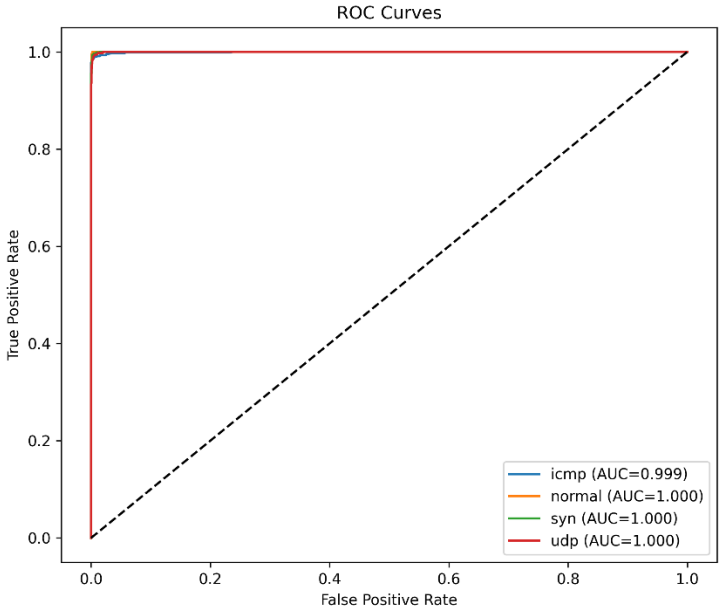


Figure 3-2: ROC Curves

The results in table 3-5 show that the false positive rates of all classes remain very low, confirming that the proposed model rarely misclassifies legitimate traffic or other attack categories. This characteristic is particularly important in practical network monitoring systems, where minimizing false alarms improves operational efficiency and reduces unnecessary administrative intervention.

Table 3-5: False Positive Rate by traffic class

Class	False Positive Rate
Normal	0.001481
ICMP Flood	0.001481
SYN Flood	0.002963
UDP Flood	0.004815

3.2. Overfitting Analysis

The high classification accuracy obtained by the proposed XGBoost model may raise concerns regarding potential overfitting, particularly because the dataset was generated in a controlled experimental environment. Consequently,

additional validation experiments were conducted to assess the model's generalization capability.

- K-Fold Cross-Validation: A 5-fold cross-validation procedure was performed using the complete dataset. The classification accuracy achieved in each fold is presented in Table 3-6. The model obtained a mean cross-validation accuracy of 99.07% with a standard deviation of 0.14%, indicating that the performance remained consistent across different data partitions. Although these results suggest that the model generalizes well within the collected dataset, they cannot completely eliminate the possibility of overfitting to the characteristics of the experimental environment.

Table 3-6: Five-Fold Cross-Validation Results

Fold	Accuracy
Fold 1	98.81%
Fold 2	99.11%
Fold 3	99.08%
Fold 4	99.14%
Fold 5	99.19%
Mean accuracy	99.07%
Standard Deviation	0.14%

- Built-in Anti-Overfitting Regularization: The XGBoost architecture inherently integrates L1 (Lasso) and L2 (Ridge) regularizationto penalize complex model structures and control capacity boundaries [13]. Furthermore, hyperparameters such as max_depth and subsample were used to constrain the model's complexity and prevent over-indexing on specific noise.

Overall, the cross-validation results and the consistent performance metrics indicate that the proposed model exhibits stable predictive capability on the available dataset. Nevertheless, further validation using real-world network traffic or publicly available intrusion detection datasets would be necessary to comprehensively assess the model's generalization performance.

3.4. Comparison of Machine Learning Models

To further assess the effectiveness of the proposed XGBoost model, a comparative evaluation was conducted against two widely used machine learning algorithms, namely Random Forest and Support Vector Machine (SVM). All three models were trained and evaluated using the same dataset, feature set, and

train-test split ratio (80:20) to ensure a fair and consistent comparison. The evaluation primarily focuses on classification accuracy. The comparison results are summarized in Table 3-7.

Table 3-7: Comparison of classification accuracy among ML Models

Model	Accuracy
SVM	96.78%
Random Forest	99.11%
XGBoost	99.19%

As shown in Table 3.7, XGBoost achieved the highest classification accuracy of **99.19%**, followed by Random Forest (**99.11%**) and SVM (**96.78%**). Based on these results, XGBoost was selected as the final model for deployment in the proposed network anomaly detection system due to its superior classification performance.

3.5. Evaluation

The proposed system was validated in a GNS3-based network environment under four traffic scenarios, including normal traffic, ICMP Flood, TCP SYN Flood and UDP Flood. For each scenario, network metrics were collected by Zabbix and analyzed by the XGBoost model. The prediction results were then displayed on the monitoring dashboard and stored in the database for further analysis.

3.5.1. Normal traffic

Before launching attack scenarios, the network was operated under normal conditions to establish a baseline traffic profile. During this period, no malicious activities were performed, and only regular communication between network devices was observed.

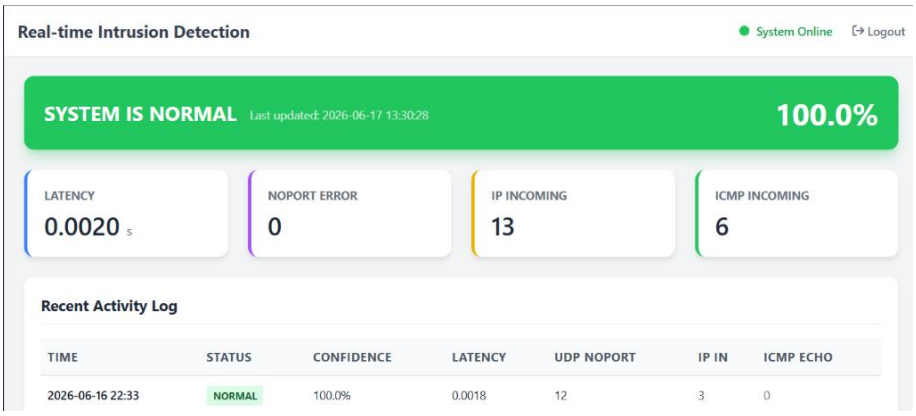


Figure 3-3: The monitoring dashboard during normal network operation



Figure 3-4: Network metrics during normal network operation

3.5.2. ICMP Flood Detection

To evaluate the system's ability to detect ICMP Flood attacks, an ICMP flooding scenario was conducted in the GNS3 environment. The attack traffic was generated from the attacker node and directed toward the target router using the following command:

```
hping3 -1 --flood --rand-source 192.168.10.1
```

Where:

- hping3: A command-line packet crafting tool used to generate and send custom network packets.
- -1: Sets ICMP mode (ICMP Echo Request packets), used for ICMP-based traffic generation.
- --flood: Sends packets as fast as possible without waiting for replies, generating high traffic volume.
- --rand-source: Randomizes the source IP address to simulate distributed attack behavior and avoid simple source-based filtering.
- 192.168.10.1: Target IP address receiving the ICMP flood traffic.

```
root@Attacker:/# hping3 -1 --flood --rand-source 192.168.10.1
HPING 192.168.10.1 (eth0 192.168.10.1): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 3-5: The execution of the ICMP Flood attack from the attacker machine

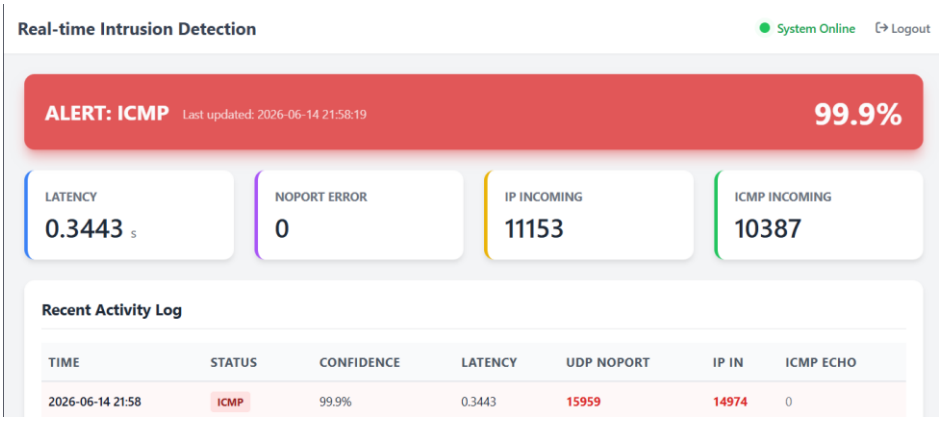


Figure 3-6: ICMP Flood detection alert on the monitoring dashboard

As shown in Figure 3.7, the dashboard displayed noticeable increases in latency, ICMP requests, and IP packet incoming during the attack. Based on the collected feature values, the XGBoost model classified the traffic as ICMP Flood.

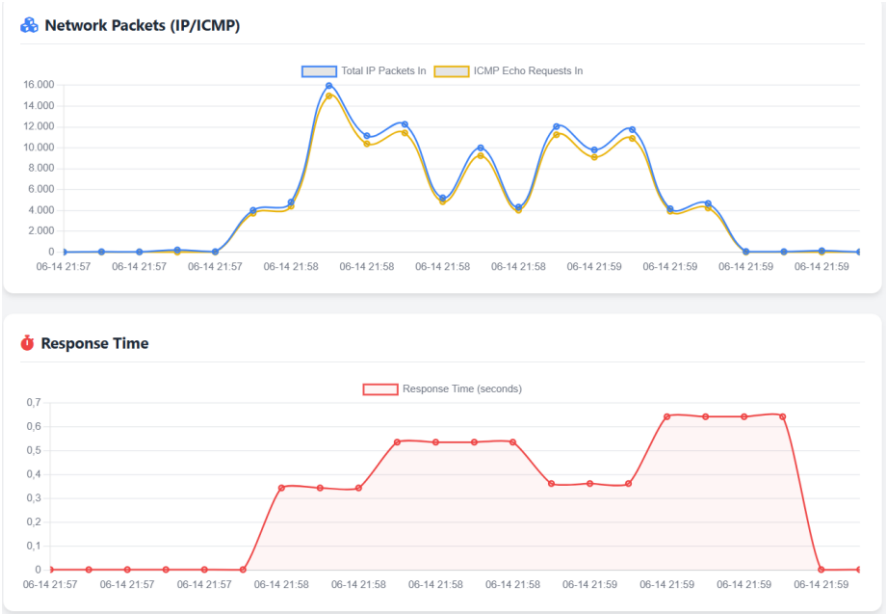


Figure 3-7: Network metrics during the ICMP Flood attack

3.5.3. TCP SYN Flood Detection

A TCP SYN Flood attack was launched from the attacker node against the target router using the following command:

hping3 -S --flood --rand-source -p 80 192.168.10.1

Where:

- -S: TCP SYN flag
- --flood: high-rate sending

- --rand-source: spoofed IPs
- -p 80: target port
- 192.168.10.1: target IP

```
root@Attacker:/# hping3 -S --flood --rand-source -p 80 192.168.10.1
HPING 192.168.10.1 (eth0 192.168.10.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 3-8: The execution of the TCP SYN Flood attack from the attacker machine

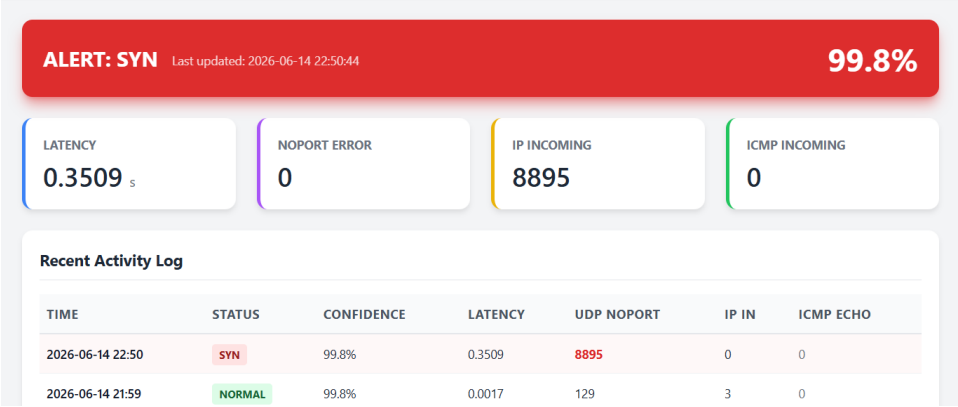


Figure 3-9: TCP SYN Flood detection alert on the monitoring dashboard

As shown in Figure 3.9, the dashboard displayed noticeable increases in latency, and IP packet incoming during the attack. Based on the collected feature values, the XGBoost model classified the traffic as SYN Flood.

Based on the monitored metrics, the XGBoost model classified the traffic as SYN Flood.

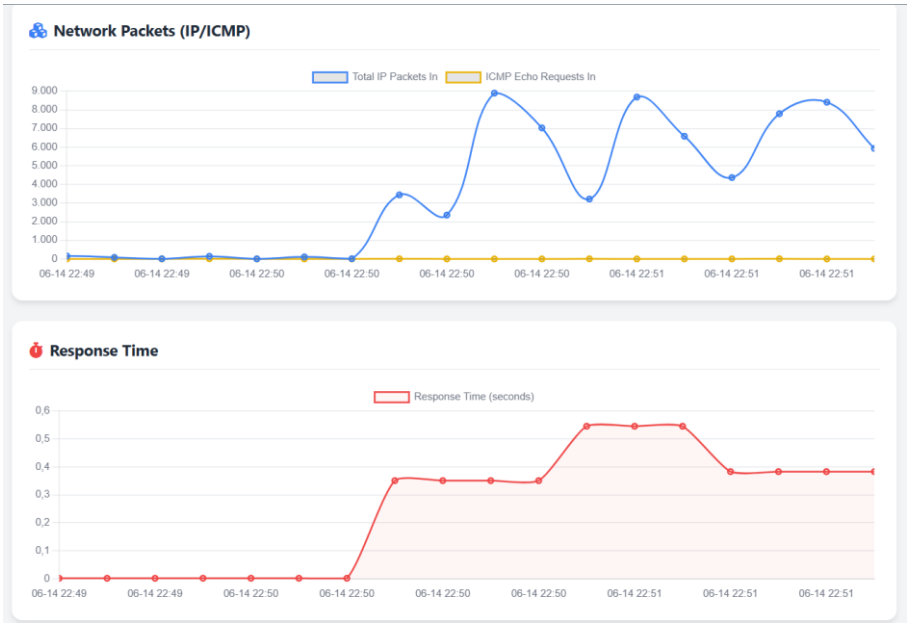


Figure 3-10: Network metrics during the SYN Flood attack

3.5.4. UDP Flood Detection

A UDP Flood attack was launched from the attacker node against the target router using the following command:

hping3 -2 --flood -p 80 192.168.10.1

Where:

- -2: UDP mode
- --flood: high-rate packet sending
- -p 80: destination port
- 192.168.10.1: target IP address

```
root@Attacker:/# hping3 -2 --flood -p 80 192.168.10.1
HPING 192.168.10.1 (eth0 192.168.10.1): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 3-11: The execution of the TCP UDP Flood attack from the attacker machine

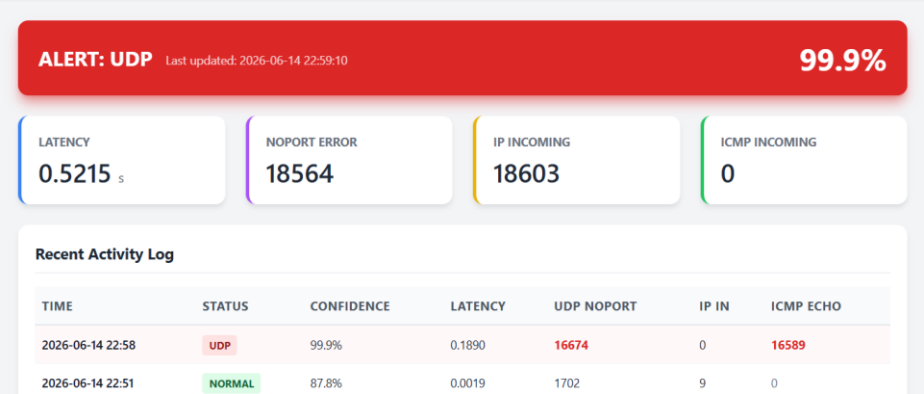


Figure 3-12: TCP UDP Flood detection alert on the monitoring dashboard

As shown in Figure 3.13, the dashboard displayed noticeable increases in latency and IP packet incoming and UDP noport error during the attack. Based on the collected feature values, the XGBoost model classified the traffic as UDP Flood.

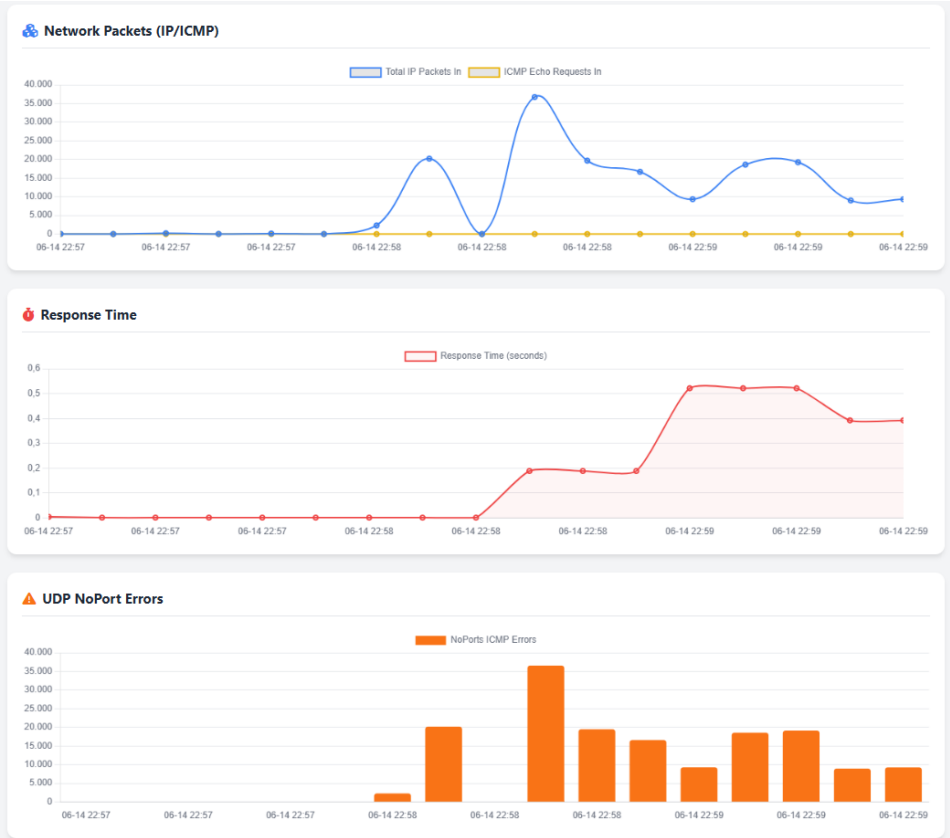


Figure 3-13: Network metrics during the UDP Flood attack

3.5.5. Performance

To evaluate the operational capability of the system, stress tests were conducted within the GNS3 simulation environment. System performance was profiled based on two core technical metrics:

- End-to-End Latency (s): The total elapsed time from the initiation of the attack to its final detection and logging by the backend system.
- Attack Throughput (packet/s): The intensity of the malicious traffic generated by the attacking tool per second.

Table 3-8: Latency and Throughput Evaluation

Attack Scenario	End-to-End Latency (s)	Throughput (packet/s)
SYN Flood	19.00 *s*	≈ 88,156
UDP Flood	21.07 *s*	≈ 90,016
ICMP Flood	18.69 *s*	≈ 75,867

3.5.6. Scalability

The scalability of the proposed architecture was evaluated to determine its capability to monitor multiple network infrastructure components simultaneously. Due to the physical hardware limitations of the host machine (8 GB RAM and localized CPU topologies), running a large-scale live network environment within GNS3 is strictly constrained. Empirically, the host system experiences heavy latency and performance degradation when operating more than 10 virtual Cisco routers concurrently alongside the Zabbix Server, FastAPI backend, and MongoDB instances.

In summary, this chapter demonstrates the successful implementation and testing of the AI-based IDS. The system is evaluated under multiple attack scenarios.

CONCLUSION

1. Achieved Results:

Through the completion of this project, I have gained both theoretical knowledge and practical experience in the fields of Network Security and Artificial Intelligence.

Theoretical Knowledge

- **Machine Learning:** Improved understanding of machine learning techniques, particularly tree-based ensemble models such as XGBoost, and their application in network traffic classification using structured tabular data.
- **Network Security:** Acquired knowledge of network protocols and traffic analysis, including the behavior of different attack types such as ICMP Flood, UDP Flood, and SYN Flood, as well as network monitoring using SNMP and Zabbix.
- **System Architecture:** Learned the principles of designing scalable system architectures, including RESTful API development, asynchronous processing, and NoSQL database integration.

Practical Achievements

- **System Development:** Developed an AI-based intrusion detection system capable of real-time network traffic monitoring and anomaly detection.
- **Network Simulation:** Built and tested a network environment using GNS3 to simulate realistic traffic and attack scenarios.
- **Dashboard Implementation:** Implemented a real-time monitoring dashboard using FastAPI, enabling visualization of network metrics, system logs, and AI prediction results in a web interface.

Contributions

- **System Implementation:** A real-time network monitoring and anomaly detection system was successfully developed by integrating GNS3, Cisco routers, Zabbix, FastAPI, MongoDB, and an XGBoost-based machine learning model. The system continuously collects SNMP telemetry, performs real-time inference, and visualizes detection results through a web dashboard.
- **Dataset Construction:** A balanced dataset containing 18,000 labeled samples was constructed under controlled network conditions. The dataset

includes four traffic classes (Normal, ICMP Flood, SYN Flood, and UDP Flood), with 4,500 samples for each class, providing training data for the classification model.

2. Limitations

- **Limited network topology scale:** The simultaneous operation of GNS3 (with Cisco router images), Docker, Zabbix Server, and the AI model requires significant computational resources (CPU and RAM). Due to hardware limitations of a personal computer, the current network topology is implemented on a small scale with a limited number of nodes. As a result, it does not fully represent the complexity and high traffic volume of a real enterprise network.
- **Limited diversity of attack scenarios and dataset:** The current XGBoost model is trained mainly on basic volumetric DoS attacks, including ICMP Flood, SYN Flood and UDP Flood. It has not yet been extended to detect more sophisticated application-layer attacks, such as HTTP Flood or SQL Injection, nor long-term stealth attacks such as Advanced Persistent Threats (APT).
- **In-Band Telemetry Bottleneck**: The current architecture relies on in-band management, meaning SNMP monitoring traffic shares the same physical/logical channel as regular user traffic. Under extreme volumetric saturation (such as the simulated ICMP Flood), the network channel becomes congested, restricting the Zabbix server from retrieving fresh metrics and forcing the system into a temporary blind spot.

3. Future work

In future development, several improvements can be made to enhance the performance and applicability of the system.

Firstly, the network topology can be expanded to simulate a larger and more complex enterprise-scale environment. This would require optimizing system resources or deploying the system on cloud infrastructure to support higher traffic loads and more network nodes.

Secondly, the dataset can be extended to include more diverse and sophisticated attack types, particularly application-layer attacks such as HTTP Flood, SQL Injection, and more advanced persistent threats (APT). This would improve the robustness and generalization ability of the AI model.

Finally, the system architecture can be improved by deploying a dedicated Out-of-Band management network. This separates SNMP monitoring traffic

from normal network traffic, reducing telemetry data loss during high-volume attacks.

REFERENCES

[1] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges" IEEE, vol. 2, no. 20, 2019.

[2] KDDCup99, S. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009.

[3] CICIDS2017, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," ICISSP, 2018.

[4] A. L. Buczak and E. Guven. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection". IEEE Communications Surveys & Tutorials, , vol. 18, no. 2, 2016.

[5] Cloudflare, "Ping (ICMP) flood DDoS attack". Available: https://www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack/

[6] Cloudflare, "SYN flood attack". Available: https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/

[7] Cloudflare, "UDP flood attack". Available: https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/

[8] GeeksForGeek, "XGBoost". Available: https://www.geeksforgeeks.org/machine-learning/xgboost/

[9] Tianqi Chen, Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System", 2016.

[10] Shwartz-Ziv, R., & Armon, A. (2022). "Tabular data: Deep learning is not all you need". Information Fusion, 81, 84-90.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[12] A. Géron, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*",* 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2022.

[13] C. M. Bishop, "Pattern Recognition and Machine Learning". New York, NY, USA: Springer, 2006.

APPENDIX

Link github: https://github.com/V1etDuy/AI-IDS.git