The World Islamic Sciences and Education University

جامعة العلوم الاسلامية العالمية

Faculty of Information Technology

كلية تكنولوجيا المعلومات

GRADUATION PROJECT

Title:

Network Anomaly Detection System

Students:

Osama Alaa Al-deen	3210606034	Cyber Security
Abd Al-rahman Al-awartany	3210606038	Cyber Security
Ibrahim Al-halabiah	3210603072	Computer Network Systems

Supervisor:

Dr.Firas Al-zobi

SEMESTER I

2024/2025

Acknowledgments

We would like to begin by expressing our sincere gratitude to Allah for granting us the strength, wisdom, and perseverance to complete this project successfully. Our heartfelt thanks go to our beloved families, whose unwavering support, encouragement, and prayers have been a source of motivation throughout this journey. We would also like to extend our deep appreciation to our esteemed university, The World Islamic Science and Education University, for providing us with the opportunity and platform to collaborate and bring this project to fruition. Our gratitude also goes to the University President for his leadership and vision, and to the Dean of the IT College for fostering an environment that encourages innovation and academic excellence. We also acknowledge the efforts of the dedicated doctors and professors of the IT College, whose knowledge and guidance have greatly contributed to our learning and growth.

Lastly, we wish to acknowledge with deep appreciation the invaluable guidance and encouragement of our esteemed professor, Dr. Firas Al-zobi, whose insightful feedback and unwavering support have been instrumental throughout the development of this project.

Abstract

The Network Anomaly Detection System (NADS) has been developed to enhance network monitoring and security by transitioning from traditional manual detection methods to an automated, AI-driven approach. This system leverages modern technologies to ensure efficiency, accuracy, and scalability in detecting and addressing network anomalies.

NADS aims to save time, effort, and resources for network administrators and users by offering a user-friendly interface and robust functionality. Feedback on existing systems and their limitations was gathered through surveys to design a solution tailored to user needs.

The system has been developed in two main parts:

A Streamlit-based web interface for real-time interaction, allowing administrators to monitor network traffic, view anomaly alerts, and generate reports.

A backend powered by TShark for network traffic analysis, coupled with a Keras-based deep learning model to detect anomalies with high accuracy.

Data preprocessing and analysis are managed using Pandas, while Plotly is employed to visualize packet capture and anomaly patterns interactively. The integration of these tools enables the system to process large volumes of data efficiently and provide actionable insights to users.

In the future, NADS aims to expand its capabilities by incorporating advanced machine learning models and additional features to support broader network environments, ensuring comprehensive protection against evolving security threats.

Table of Contents

Acknowledgements I

Abstract II

List of Tables IV

List of Figures V

List of abbreviations V

Chapter 1: Introduction 1

1.1 Overview 2

1.2 Problem Statement 3

1.3 Project Objectives 3

1.4 Research Strategy (Framework) 4

1.5 Gantt Chart 5

1.6 Project outline 5

Chapter 2: Literature Review 7

2.1 Overview 8

2.2 Related Work 8

2.3 Summary 11

Chapter 3: Methodology 12

3.1 Overview 13

3.2 Feasibility Study 13

3.3 Tools 15

3.4 Requirements 15

3.5 Methodology Process 16

3.6 Data Collection 18

Chapter 4: Design Models 20

4.1 Overview 21

4.2 Context Diagram 21

4.3 Data Flow Diagram 22

4.4 Use Case Diagram 25

Chapter 5: Experiments and Results 26

5.1 Overview 27
5.2 Testing Methodologies 27
5.3 Discussion and Evaluation 46
Chapter 6: Conclusion and Future Works 47
6.1 Overview 48
6.2 Summary about the Project 48
6.3 Achieved Objectives 48
6.4 Main Contributions of the Work 49
6.5 Limitations 49
6.6 Future Work 50
6.7 Conclusion 50
REFERENCES 51
Appendix A 53
Appendix B 53
Appendix C 53

List of tables

Table 1: Operational Feasibility 15
Table 2: Tools 15
Table 3: Sprints 16
Table 4: Testing Results 38

List of Figures

Figure 1: Scrum Process 4
Figure 2: Gantt Chart 5
Figure 3: The Algorithm of Paper 1 9
Figure 4: The Scope of Paper 2 10
Figure 5: Context Diagram 21
Figure 6: Data Flow Diagram 22
Figure 7: Use Case Diagram 25

Figure 8: Unit Testing Results A 28
Figure 9: Unit Testing Results B 29
Figure 10: Unit Testing Results C 30
Figure 11: Unit Testing Results D 31
Figure 12: Unit Testing Results E 32
Figure 13: Unit Testing Results F 33
Figure 14: Unit Testing Results G 34
Figure 15: Unit Testing Results H 35
Figure 16: Unit Testing Results I 36
Figure 17: Unit Testing Results J 37
Figure 18: Unit Testing Results K 38

List of abbreviations

- **ANN** – Artificial Neural Network
- **LUFlow** – A flow-based network intrusion detection dataset
- **CTI** – Cyber Threat Intelligence
- **GUI** – Graphical User Interface
- **JD** – Jordanian Dinar
- **DDoS** – Distributed Denial of Service
- **NADS** – Network Anomaly Detection Systems
- **BiGAN** – Bidirectional Generative Adversarial Networks
- **PSO** – Particle Swarm Optimization
- **EQ** – Emotional Quotient

CHAPTER 1

INTRODUCTION

1.1. Overview

Modern networks are the backbone of today’s interconnected world, but their increasing complexity and volume pose significant challenges for maintaining security. Cyberattacks, malicious behaviors, and unexpected anomalies can disrupt operations, compromise data integrity, and incur financial losses. Studies have shown that traditional security measures, such as signature-based systems, are insufficient for detecting new and evolving threats. As a result, the need for advanced solutions that can proactively monitor, analyze, and identify irregular network behavior has grown significantly.

A Network Anomaly Detection System is designed to address these challenges by utilizing advanced machine learning techniques. Unlike conventional systems, it can identify previously unseen threats by learning traffic patterns and distinguishing between normal, malicious, and outlier behaviors. Tools such as Tshark allow real-time traffic capture, while datasets like LUFlow offer a foundation for training Artificial Neural Network (ANN) models, ensuring accuracy and adaptability.

Anomalies in network traffic can signify various issues, including Distributed Denial of Service (DDoS) attacks, unauthorized access attempts, or misconfigured systems. These threats not only affect individual organizations but can also compromise national infrastructures and sensitive data. Detecting anomalies early enables faster responses, reducing the impact of potential security breaches.

To make this system accessible and effective, it must be integrated into a user-friendly platform. Applications built using frameworks like Streamlit can provide real-time monitoring, detailed traffic visualizations, and immediate alerts, empowering network administrators with actionable insights. Such a system simplifies the detection process, streamlines anomaly investigation, and promotes a proactive approach to network security.

The Network Anomaly Detection System bridges the gap between traditional methods and modern security needs, ensuring a more robust and resilient network environment while mitigating the risks posed by increasingly sophisticated threats.

1.2 Problem Statement

In today's rapidly evolving digital landscape, networks are essential for the functioning of businesses, governments, and everyday communication. However, detecting and addressing security threats in these networks has become increasingly difficult due to the sheer volume and complexity of traffic. Traditional security systems often rely on predefined rules and signature-based methods, which are ineffective against new, evolving, and previously unseen threats. These systems struggle to differentiate between benign and malicious traffic, leading to high false positive rates and missed anomalies.

Network administrators face significant challenges in ensuring the security and stability of their networks. With increasing data volumes and the need for real-time analysis, manually identifying threats or irregularities is no longer practical. As a result, there is a pressing need for advanced systems that can monitor, analyze, and identify network anomalies using intelligent, adaptive techniques. The lack of such systems leaves networks vulnerable to potential breaches, unauthorized access, and other cyber threats that can disrupt operations and compromise sensitive data.

1.3 Project objectives

1. Develop and train an artificial neural network (ANN) model to detect malicious network intrusions using the LUFlow dataset.

2. Understand the challenges faced in detecting network anomalies and evaluate the current methods and datasets used in anomaly detection.

3. Develop and train an Artificial Neural Network (ANN) model to classify network traffic as benign, malicious, or anomalous.

4. Implement a user-friendly Streamlit application for real-time monitoring, visualization, and anomaly detection.

5. Validate the accuracy, reliability, and performance of the system through comprehensive testing and comparison with existing solutions.

1.4 Research Strategy (Framework)

Scrum is an agile framework designed to help teams work together on complex projects by breaking them down into manageable tasks and delivering incremental value through iterative cycles called **Sprints.**

a sprint is a short, time-boxed period during which specific tasks are completed to deliver incremental value. It involves planning, daily stand-ups, reviews, and retrospectives to ensure continuous improvement, flexibility, and regular feedback, allowing the team to stay aligned with stakeholder needs and adapt to changes effectively.

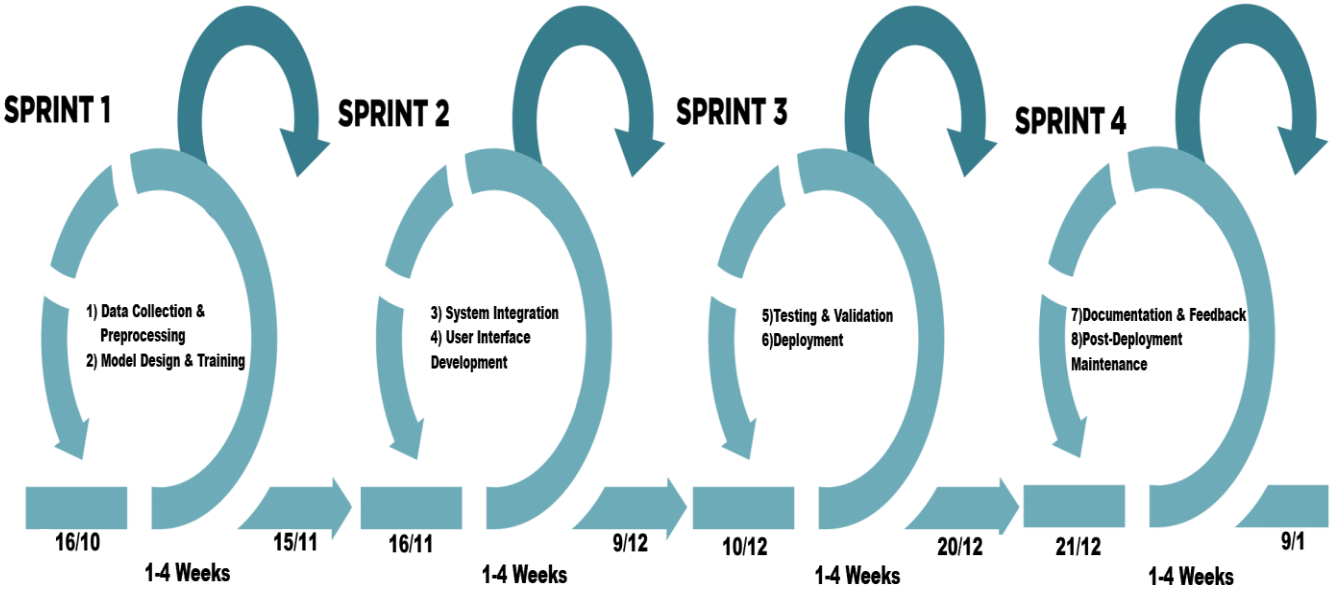


Figure 1:Scrum Process

1.5 Gantt chart: Using scrum methodology.

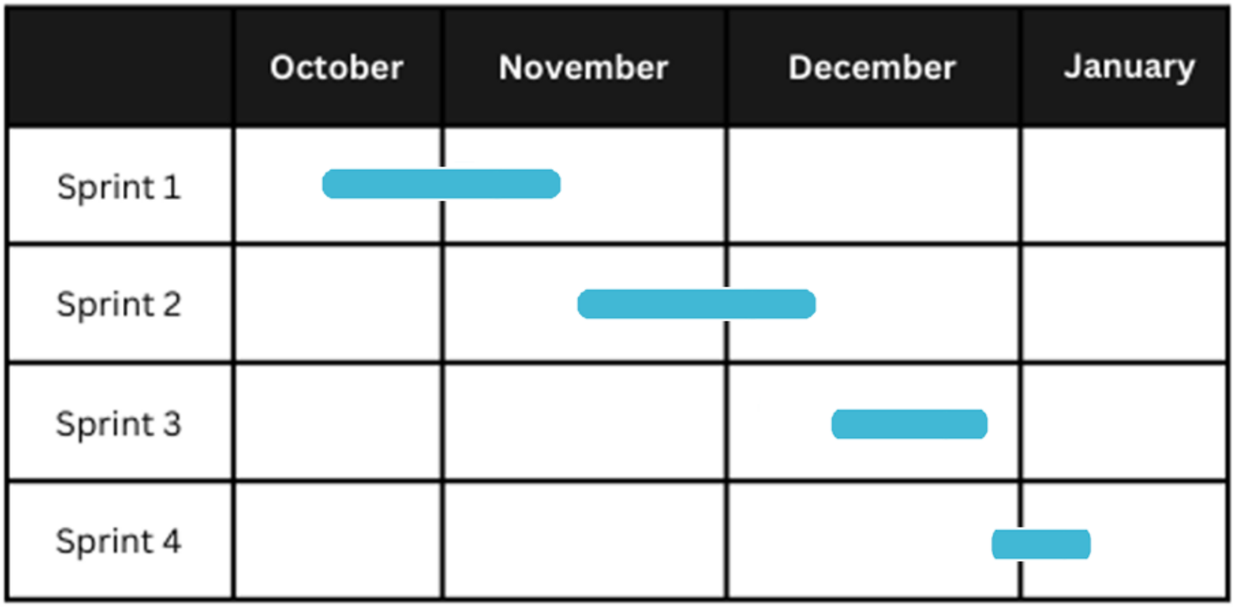


Figure 2: Gantt Chart

1.6 Project outline

- Chapter 2: This chapter examines previous studies on network anomaly detection. It reviews various techniques, including both traditional and modern machine learning methods. The strengths and weaknesses of these approaches are discussed, identifying common challenges such as detecting new threats and minimizing false positives.
- Chapter 3: Feasibility study and methodology used in the project, functional and non-functional requirements.
- Chapter 4: Show how the system works by a set of diagrams most easily and simply and show how to use the system.

- Chapter 5: The implementation and evaluation of the system. And explaining each implementation and how it was implemented.
- Chapter 6: A summary of the project and its future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

Network Anomaly Detection Systems (NADS) play a crucial role in enhancing cybersecurity by identifying malicious and anomalous activities within network traffic. Recent advancements have utilized machine learning and deep learning techniques for more accurate detection. For instance, datasets like LUFlow provide robust ground truth by correlating malicious behaviors with Cyber Threat Intelligence (CTI) sources, capturing normal traffic, malicious traffic, and outliers to promote research into emerging threats. Studies have demonstrated the effectiveness of Artificial Neural Networks (ANNs) in classifying network traffic into benign, malicious, or anomalous categories, achieving high detection accuracy. Tools like Tshark have been employed for real-time traffic capture and analysis, aiding anomaly detection systems in identifying novel threats. However, challenges remain in processing high-volume traffic, detecting sophisticated attack patterns, and minimizing false positives. NADS frameworks incorporating real-time analysis, visualization, and continuous dataset updates are vital for adapting to evolving threats.

2.2 Related Work

1- Enhanced Anomaly Traffic Detection Framework Using BiGAN and Contrastive Learning (https://cybersecurity.springeropen.com/articles/10.1186/s42400-024-00297-7#Sec1)

This paper proposes an enhanced anomaly traffic detection framework that combines BiGAN and contrastive learning to address these issues. First, the original traffic data is preprocessed through three steps: data cleaning, data normalization, and data clustering, to retain helpful information, remove redundant features, and effectively reduce the dimensionality and complexity of the data. Next, an abnormal traffic detection model was built based on BiGAN, and a scoring function was set to detect abnormal traffic. In addition, given the advantages of contrastive learning technology in improving model robustness (Tian et al. 2020; Liu et al. 2022), we combine it with BiGAN to improve the model's ability to express complex data features and improve detection Effect. Experimental results show that the method proposed in this paper outperforms existing advanced methods in terms of detection accuracy, recall rate and F1 score. In the context of processing large-scale, high-dimensional network traffic, this method significantly improves the efficiency of traffic anomaly detection and has broad practical application potential.

The overview of the algorithm in this paper:

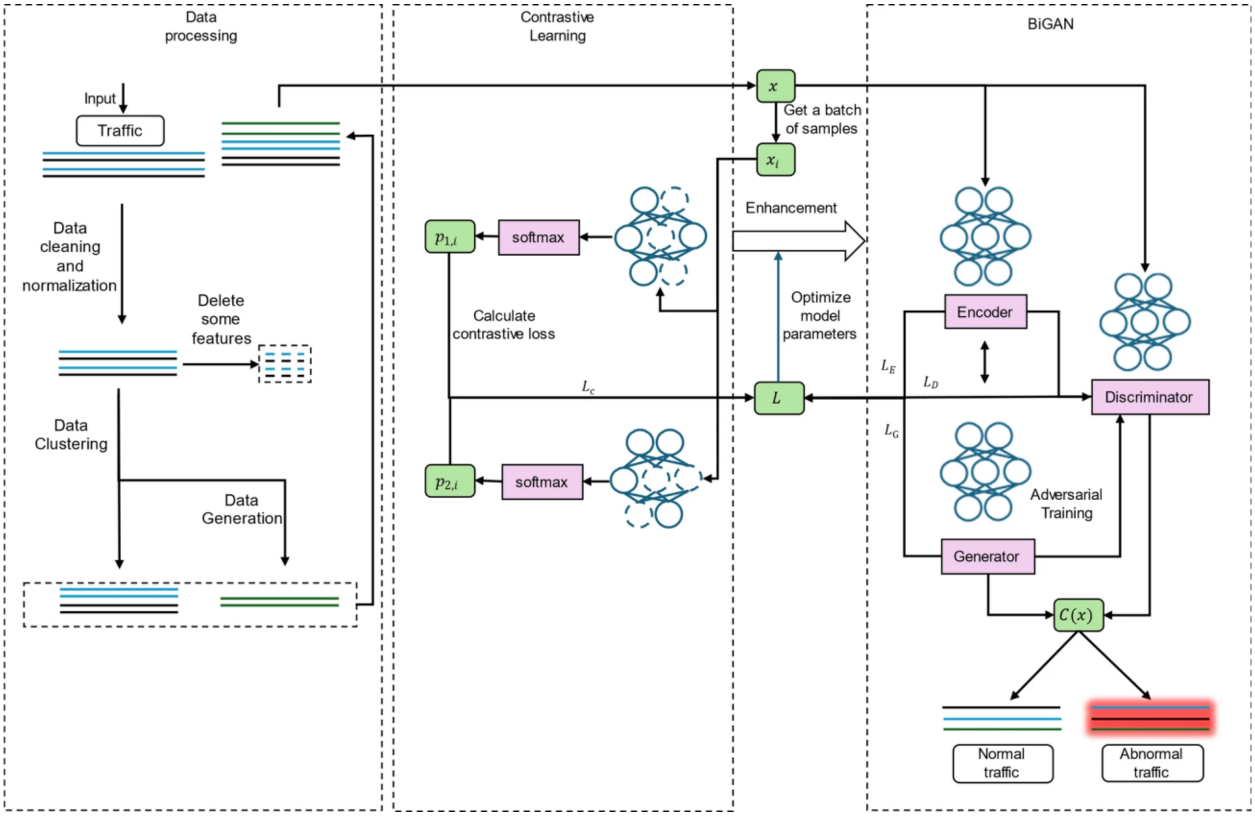


Figure 3: The Algorithm Paper 1

a. Process traffic data: data cleaning and normalization, data clustering.
b. Train the processed data based on BiGAN and establish a scoring function C(x) based on the loss function to detect anomalies.
c. While training BiGAN, use contrastive learning technology to enhance the model, continuously optimize model parameters, and improve the detection effect of the model.

2- Comprehensive Survey of Anomaly Detection Techniques for High-Dimensional Big Data (https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00320-x)

This survey reviews various anomaly detection techniques specifically for high-dimensional big data. It identifies unique challenges in this domain and categorizes various techniques, including statistical, machine learning, and deep learning approaches. The survey also discusses the application of these methods across different fields, offering insights into their effectiveness and limitations. This resource is valuable for understanding the current landscape of anomaly detection in complex data environments.

The scope of this paper:

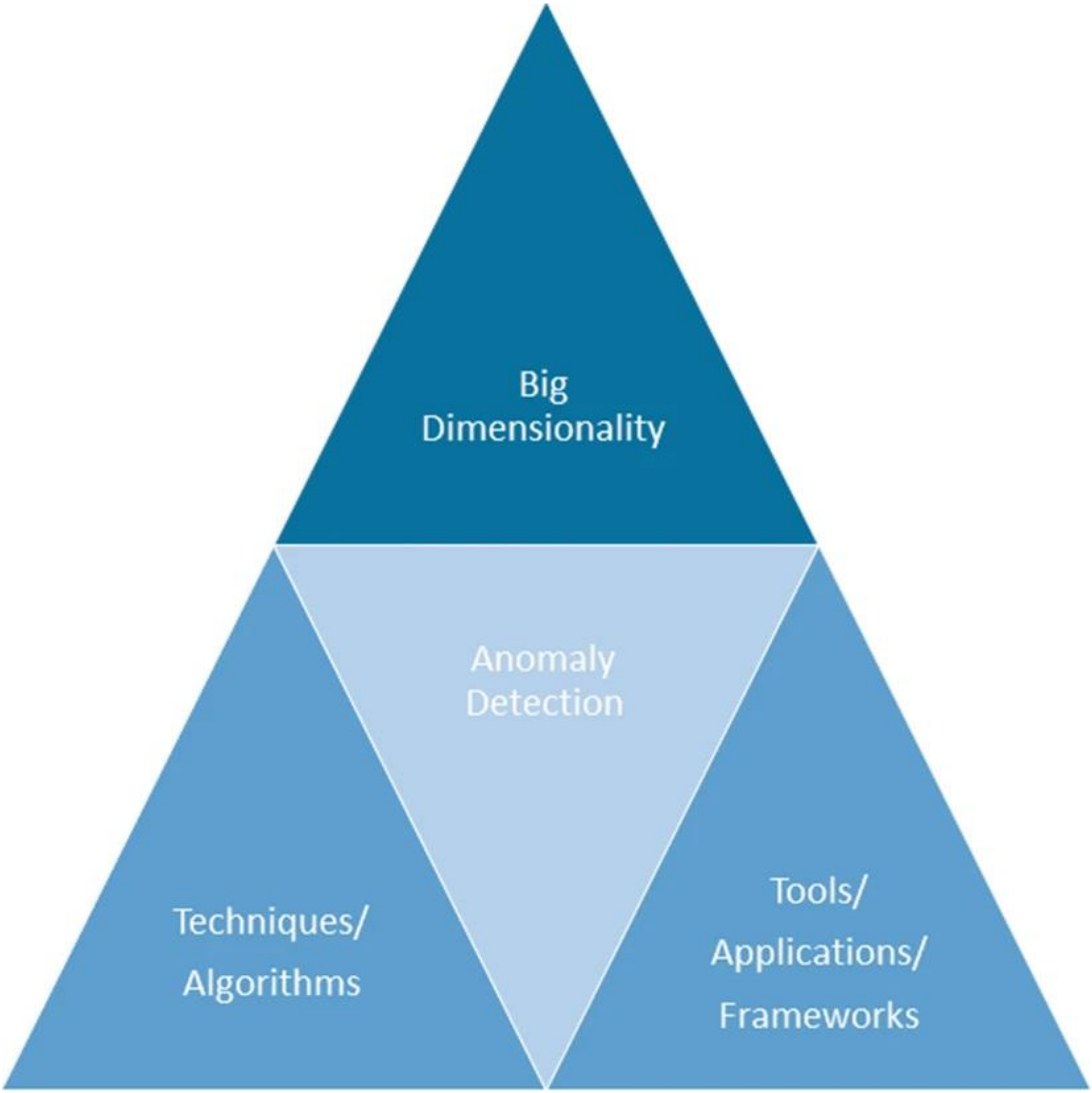


Figure 4: The Scope of Paper 2

This survey aims to document the state of anomaly detection in high dimensional big data by identifying the unique challenges using a triangular representation of vertices: the problem (big dimensionality), techniques/algorithms (anomaly detection), and tools (big data applications/frameworks). Authors' work that falls directly into any of the vertices or closely related to them are taken into consideration for review. Furthermore, the limitations of traditional approaches and current strategies of high dimensional data are discussed along with the recent techniques and applications on big data required for the optimization of anomaly detection.

Hence, the contributions of this survey are threefold:

a. Review existing literature to highlight the theoretical background and current strategies for managing the big dimensionality problem.

b. Identify unique challenges and review the techniques/algorithms of anomaly detection that address the problems of high dimensionality and big data.

c. Review the big data tools, applications and frameworks for anomaly detection in high dimensional big data.

2.2.1 Issues and challenges faced by the other project

1- Data Labeling: One of the biggest challenges is the lack of labeled data for training supervised models. Anomalies are rare events, making it difficult to obtain a sufficient amount of labeled data**.**

2- Privacy Concerns: Handling sensitive network data requires robust privacy measures to ensure that the data is protected from unauthorized access and breaches.

3- Scalability: Ensuring that the anomaly detection system can handle large-scale network traffic in real-time without significant performance degradation is a major challenge.

2.3 Summary

The review covers various network anomaly detection methods, highlighting the shift from traditional techniques to advanced machine learning models. Despite their promise, challenges like data labeling, Privacy Concerns, and Scalability….etc. This sets the stage for developing an improved detection system.

CHAPTER 3

METHODOLOGY

3.1 Overview

The methodology chapter outlines the structured approach taken to develop the Network Anomaly Detection System. This project follows an scrum framework, utilizing iterative development through sprints to ensure flexibility and continuous improvement. Each stage of the methodology is meticulously designed to address the challenges of anomaly detection in network traffic, from data collection to system deployment. The chapter begins by discussing the feasibility study, followed by detailed explanations of the requirements, tools, and technologies used. Additionally, it presents the methodology process, including data preprocessing, model design, real-time integration, and the development of a user-friendly interface for monitoring and visualization. This comprehensive methodology ensures the system's scalability, reliability, and real-world applicability, aiming to deliver a robust solution for modern network security challenges.

3.2 Feasibility Study

When businesses develop or purchase software, they face risks. To mitigate these risks, they need to assess whether the software can succeed in the ever-changing market. Feasibility studies can help businesses determine whether their software projects are likely to be successful in the future. This is crucial for all industries, but particularly for software development. By conducting feasibility studies, businesses can gain insights into potential risks and opportunities associated with their software products. The feasibility study is part of the initial design stage of any proposed project/plan. It is done to evaluate the feasibility of a proposed project or existing software used by the business. It can assist in identifying and assessing the opportunities and threats present in the natural environment, the resources needed for the project, and the chances of success.

There are typically three types of feasibility studies that can be conducted for the project, Technical Feasibility, Operational Feasibility and Economic Feasibility.

3.2.1 Technical Feasibility study:

- Infrastructure and Tools:
  The project relies on established tools and frameworks such as Keras for building the ANN model, Tshark for real-time traffic capture, and Streamlit for the user interface. These tools are readily available and well-documented.
- Data Availability:
  The LUFlow dataset provides a reliable source of labeled network traffic data for training and evaluation.
- Expertise Required:
  The team requires expertise in machine learning, networking, and software development. These skills are essential for designing, training, and deploying the system.
- Challenges:
  Addressing issues like false positives/negatives, real-time processing, and scalability will require advanced optimization techniques and model tuning.

3.2.2 Economic Feasibility

- Development Cost:
  The project requires investments in computational resources for model training and testing. Cloud platforms or local setups can be used based on budget constraints.
- Operational Costs:
  Deployment and maintenance of the system involve moderate costs, including periodic updates and monitoring infrastructure.
- Benefits:
  The system reduces the risk of cyber-attacks, saving organizations from potential financial and reputational losses. It also streamlines network monitoring processes, enhancing efficiency.

3.2.3 Operational Feasibility

- Integration:
  The system can be integrated into existing network monitoring setups, providing seamless functionality without major disruptions.
- User Training:
  Minimal training is required for network administrators to understand the interface and interpret the results. The Streamlit GUI simplifies user interaction.

- Scalability:

  The system is scalable to handle larger networks and higher traffic volumes with appropriate hardware configurations.

Item	Percentage (%)
Performance	90%
Usability	85%
Maintenance	80%
Compatibility	85%
Training	90%
Scalability	88%

Table 1: Operational Feasibility

3.3 Tools

Streamlit	**Pandas**	**Sklearn**	**Kali/Debian**	**Anaconda**	**Plotly**	
Python	Numpy	Tensorflow	Keras	Glob	Pyshark & Tshark	Jupyter-notebook

Table 2: Tools

3.4 Requirements

A. FUNCTIONAL REQUIREMENTS

1. The system must capture real-time network traffic using Tshark.
2. The system must preprocess and normalize the captured traffic data.
3. The system must classify traffic as benign, malicious, or anomalous using an ANN
4. model. The system must provide visualizations of traffic data using Streamlit.
5. The system must allow users to export anomaly detection results.
6. The system must send notifications in case of detected anomalies.

B. NON-FUNCTIONAL REQUIREMENTS

1. The system should process and classify traffic data within 5 seconds per batch.
2. The system should be scalable to handle traffic from large enterprise networks.

3. The system should provide a user-friendly and intuitive interface.
4. The system should ensure data privacy and adhere to security standards.
5. The system should operate reliably with an uptime of at least 99%.
6. The system should be compatible with major operating systems (Linux).

3.5 Methodology Process

The Scrum methodology was integral to managing the complexity of the Network Anomaly Detection System project. By dividing the work into iterative sprints, we ensured flexibility, continuous improvement, and timely delivery of objectives. Scrum ceremonies like sprint planning, daily stand-ups, and retrospectives enhanced collaboration and allowed us to address challenges, such as refining the ANN model and improving the Streamlit dashboard. This iterative approach also enabled rapid adaptation to changing requirements and stakeholder feedback. Ultimately, Scrum streamlined the development process, promoted efficiency, and ensured the delivery of a scalable and reliable anomaly detection system.

Number of Sprints	Sprint 1	Sprint 2	Sprint 3	Sprint 4
Time of Sprint	1-4 Weeks	1-4 Weeks	1-4 Weeks	1-4 Weeks
Requirement Item	FR (1): Data Collection and Preprocessing. FR (2): Model Design and Training.	FR (3): System Integration. FR (4): User Interface Development.	FR (5): Testing and Validation. FR (6): Deployment.	FR (7): Documentation and Feedback. FR (8): Post-Deployment Maintenance.
Date	16/10-15/11	16/11-9/12	10/12-20/12	21/12-9/1

Table 3: Sprnts

Sprint 1:

Step 1: Data Collection and Preprocessing

a. load LUFlow dataset \ CICFlowmeter-v4.0 \ Collected local data .

b. Preprocess the dataset:

- Handle missing or noisy data.
- Normalize numerical features for better ANN performance.
- Encode categorical data if necessary.

Step 2: Model Design and Training

a. Develop the Artificial Neural Network (ANN) using Keras:
    - Define input, hidden, and output layers.
    - Optimize the model with relevant activation functions, loss functions, and optimizers.

b. Train the ANN on the LUFlow dataset:
    - Split data into training, validation, and testing sets.
    - Monitor training metrics (accuracy, precision, recall).

c. Save the trained model for integration.

Sprint 2:

Step 3: System Integration

a. Integrate real-time traffic analysis using Tshark:
    - Parse live network traffic and format it for input into the ANN.
    - Use Python scripts to connect Tshark output to the trained model.

Step 4: User Interface Development

a. Use Streamlit to create an interactive dashboard:
    - Real-time traffic monitoring with anomaly detection results.
    - Visualizations using Pandas and Plotly (e.g., traffic distributions, anomaly timelines).
    - Filter options for IPs, ports, or traffic types.

Sprint 3:

Step 5: Testing and Validation

a. Conduct unit testing for individual components (e.g., model predictions, preprocessing pipeline).

b. Perform integration testing to ensure smooth interaction between Tshark, ANN, and the Streamlit dashboard.

c. Validate the system with labeled datasets (LUFlow or similar).

Step 6: Deployment

a. Host the system locally or on a cloud platform for broader accessibility.

b. Ensure robust system scalability for increased traffic volumes.

Sprint 4:

Step 7: Documentation and Feedback

a. Write comprehensive user and technical documentation.

b. Conduct user testing with network security experts or administrators.

c. Gather feedback for final adjustments.

Step 8: Post-Deployment Maintenance

a. Set up automated updates for the anomaly detection model using new data.

b. Monitor system performance and optimize the ANN as required.

3.6 Data Collection

Dataset contains the following:

1. LUFlow**:**

is an flow-based intrusion detection data set which contains a robust ground truth through correlation with threat intelligence service :

a. It contains telemetry containing emerging attack vectors through the composition of honeypots within Lancaster University's address space.

b. The labeling mechanism is autonomous, enabling the constant capture, labelling and publishing of telemetry to this repository.

c. Flows which were unable to be determined as malicious, but are not part of the normal telemetry profile are labeled as outliers. These are included to encourage further analysis to discover the true intent behind their actions.

d. Known normal traffic is also captured from production services, e.g. ssh and database traffic, are included in this data set.

e. This data set is constantly updated using the Citrus framework. This repository is structured into the year and month the telemetry was captured; The whole dataset was used.

f. The telemetry is captured using Cisco's Joy tool. This tool records multiple measurements associated with flows.

2) CICFlowmeter-V4.0:

is an Ethernet traffic Bi-flow generator and analyzer for anomaly detection that has been used in many Cyber-security datasets such as Android Adware-General Malware dataset. Which is a collection of multiple anonymized packet captures of real-world.

a. bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions, hence more than 80 statistical network traffic features such as Duration, Number of packets, Number of bytes, Length of packets, etc. can be calculated separately in the forward and backward directions.
b. Additional functionalities include, selecting features from the list of existing features, adding new features, and controlling the duration of flow timeout. The output of the application is the CSV format file that has six columns labeled for each flow (FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol) with more than 80 network traffic analysis features.

3) Locally captured packet :
which was analysed using multiple techniques such as using Joy-tool/ CICFlowmeter-V4.0 and wireshark for capturing the packet needed, with the help of simulating attacks based on different types ranging from ARP spoofing, Dos, ICMP as outliers, etc... but only 16 features were used as they are the needed features such as (protocol type, src IP, dst IP, src port, dest port, time stamp, duration, bytes in, bytes out).

CHAPTER 4

DESIGN MODELS

4.1 Overview

This chapter presents the design and development of key models for the Network Anomaly Detection System (NADS). It includes the representation of the Context Diagram, Use Case Diagram, and Data Flow Diagram (DFD), which collectively provide a comprehensive understanding of the system's architecture, interactions, and data flow.

4.2 Context diagram-0:

The key components and interactions within (NADS).

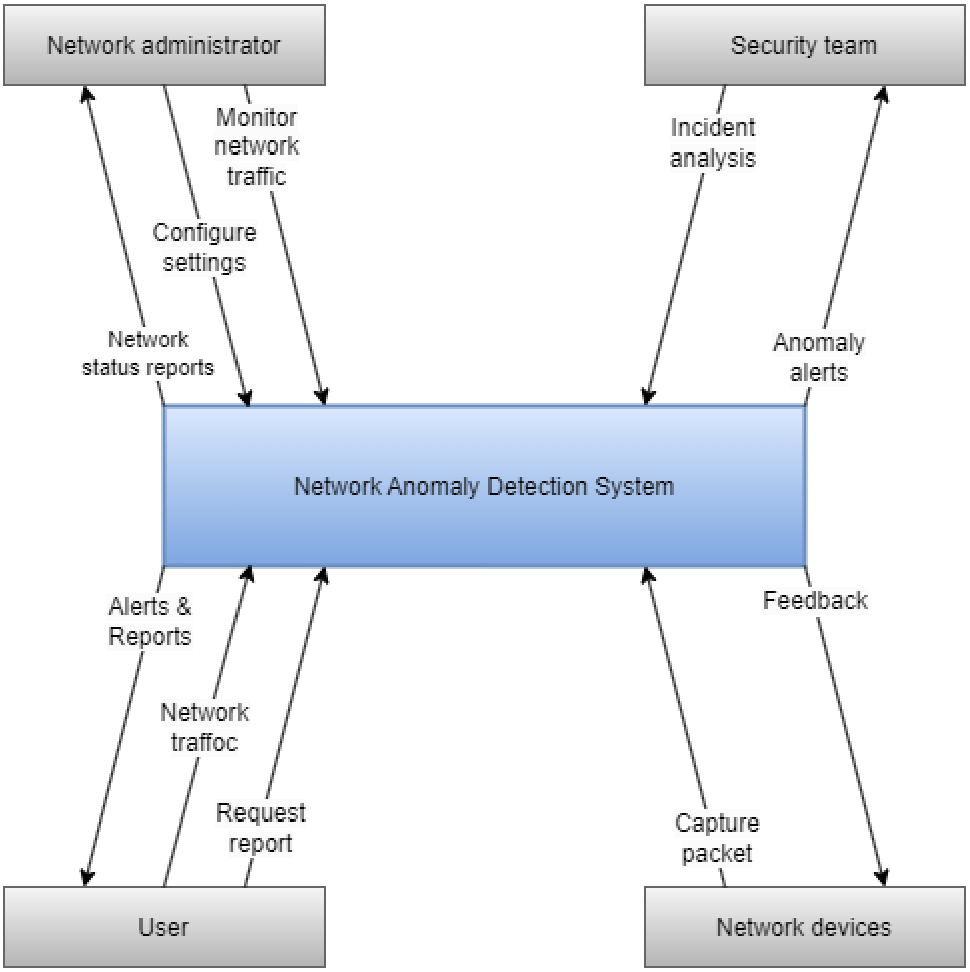


Figure 5: Context Diagram

4.3 Data flow Diagram:

The workflow within (NADS), illustrating the stages involved in capturing network traffic, filtering data, detecting anomalies, generating alerts, and providing reports to administrators and security teams.

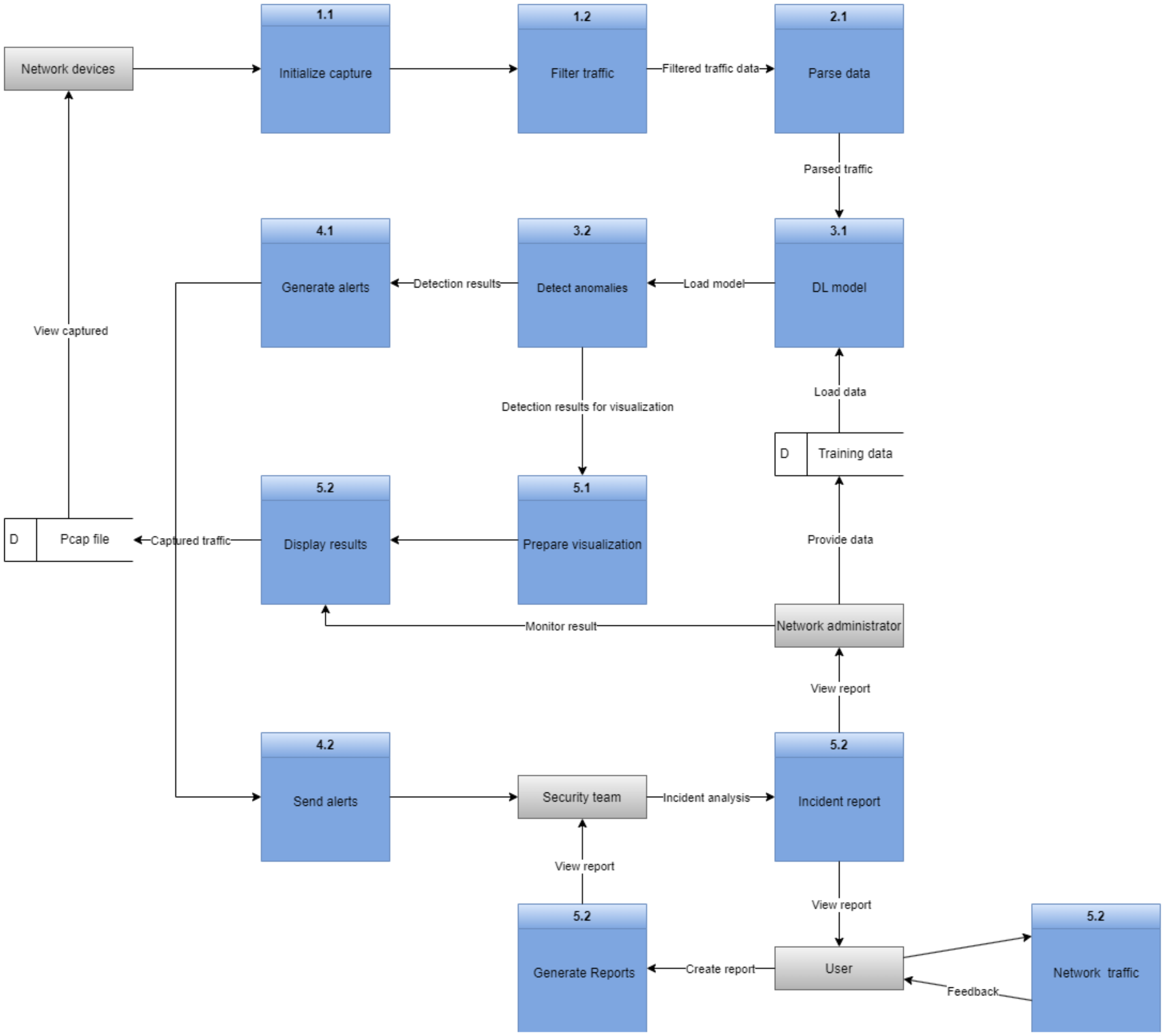


Figure 6: Data Flow Diagram

Overall Purpose:

illustrate the workflow of (NADS). It shows the various stages involved in capturing network traffic, analyzing it for potential threats, and generating alerts and reports.

Key Components and Data Flow:

1. Network Devices:
   - These are the sources of network traffic data.
   - They capture network traffic packets and send them to the system.

   a. Initialize Capture (1.1):
      - This step likely involves setting up the system to start capturing network traffic.
      - Receives network traffic packets from network devices.

   b. Filter Traffic (1.2):
      - Filters the captured traffic based on specified rules (e.g., IP addresses, ports, protocols) to focus on relevant data.
      - Receives unfiltered traffic, filters it, and passes the filtered data to the next stage.

   c. Parse Data (2.1):
      - Analyzes the filtered traffic data to extract relevant features and information.
      - Receives filtered traffic data, parses it, and generates parsed traffic data.

   d. Detect Anomalies (3.2):
      - Compares the parsed traffic data against known normal traffic patterns or predefined rules to identify potential anomalies or intrusions.
      - Receives parsed traffic data, detects anomalies, and generates detection results.

   e. DL Model (3.1):
      - This component likely involves machine learning or deep learning models trained to detect anomalies in network traffic.
      - Receives training data and generates a DL model that can be used for anomaly detection.

   f. Generate Alerts (4.1):
      - Generates alerts based on the detected anomalies.
      - Receives detection results and generates alerts.

g. Send Alerts (4.2):
    - Sends the generated alerts to the security team or other designated recipients.
    - Data Flow: Receives alerts and sends them to the security team.

h. Display Results (5.2):
    - Presents the detection results and other relevant information to the network administrator or other users.
    - Data Flow: Receives detection results and displays them.

i. Prepare Visualization (5.1):
    - Prepares visualizations (e.g., graphs, charts) to help users understand the detected anomalies and network traffic patterns.
    - Receives detection results and prepares visualizations.

j. Generate Reports (5.2):
    - Creates reports summarizing the detected anomalies, network traffic patterns, and other relevant information.
    - Receives detection results and generates reports.

2. Security Team:
    - Receives alerts, analyzes incident reports, and takes appropriate actions (e.g., investigate threats, block traffic).
    - Receives alerts and incident reports.

3. Network Administrator:
    - Monitors the system, views reports, and manages the system configuration.
    - Receives reports and provides feedback to the system.

4. User:
    - Generates network traffic through their activities.
    - Generates network traffic.
    - Feedback: Receives feedback and uses it to improve the system.

Key Points:

The diagram highlights the importance of data flow between different components of the system.

It shows how the system processes network traffic, detects anomalies, and generates alerts and reports.

The feedback loop allows for continuous improvement of the system's accuracy and effectiveness.

4.4 Use Case Diagram:

Use Case Diagram: Depicting the interactions in a Network Monitoring and Anomaly Detection System, including activities such as packet capture, anomaly detection using a Deep Learning (DL) model trained with data, generating alerts, and providing visualization tools for Network Administrators and Users.

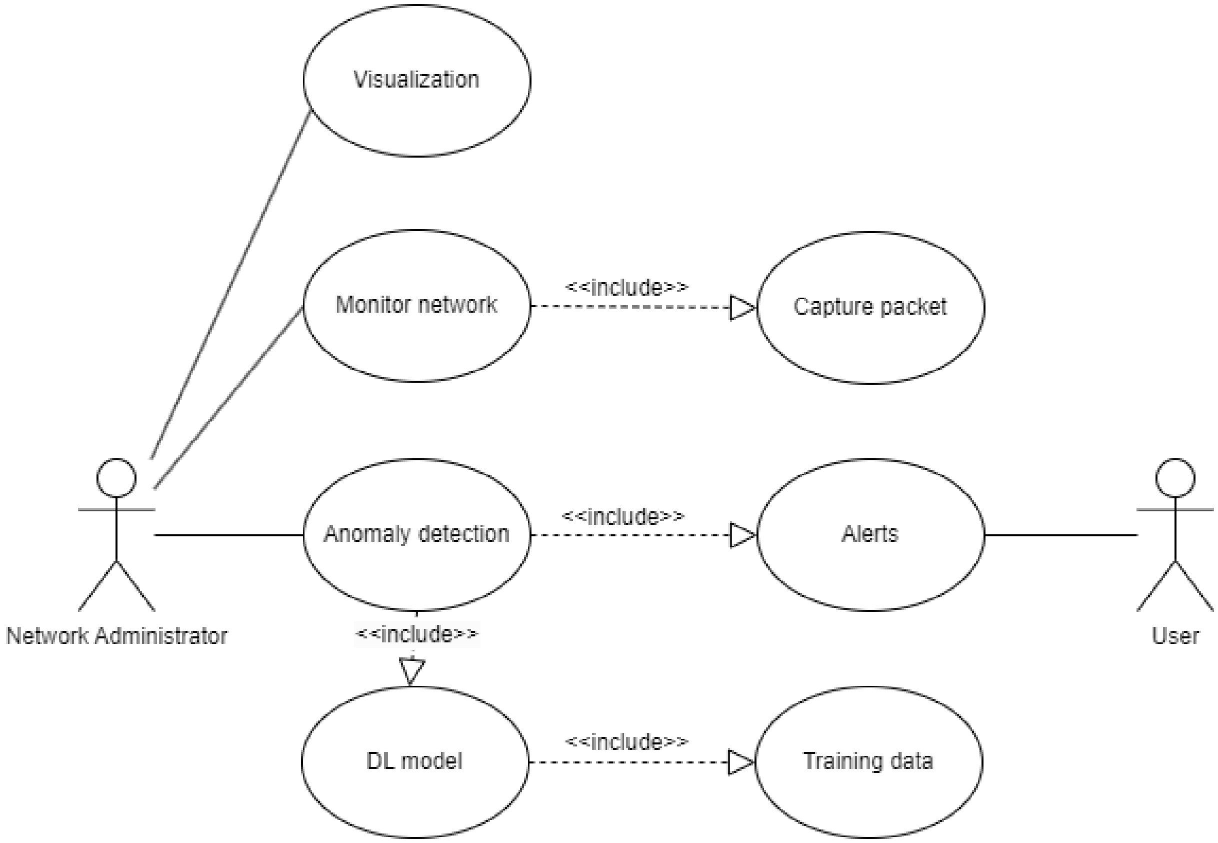


Figure 7: Use Case Diagram

CHAPTER 5

EXPERIMENTS AND RESULTS

5.1 Overview

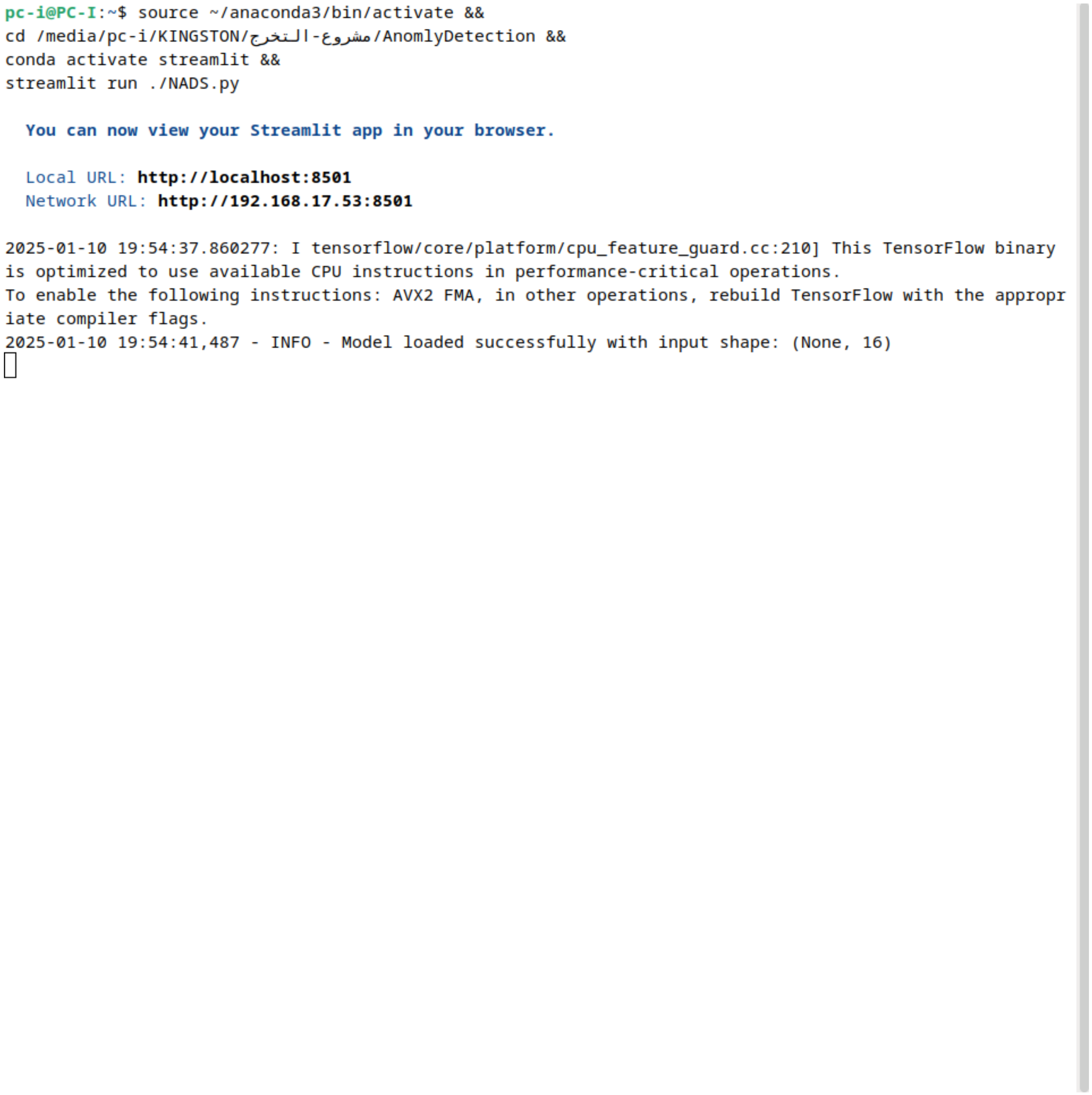
In this chapter, we will explore the significance and outcomes of requirements testing, unit testing, integration testing, and user acceptance testing, highlighting their importance in achieving a robust and reliable software solution.

5.2 Testing methodologies

5.2.1 Unit Testing Results

Unit testing is a type of software testing where individual units or components are tested. The purpose is to validate that each unit of the software code performs as expected. Unit testing is a Whitebox testing technique.

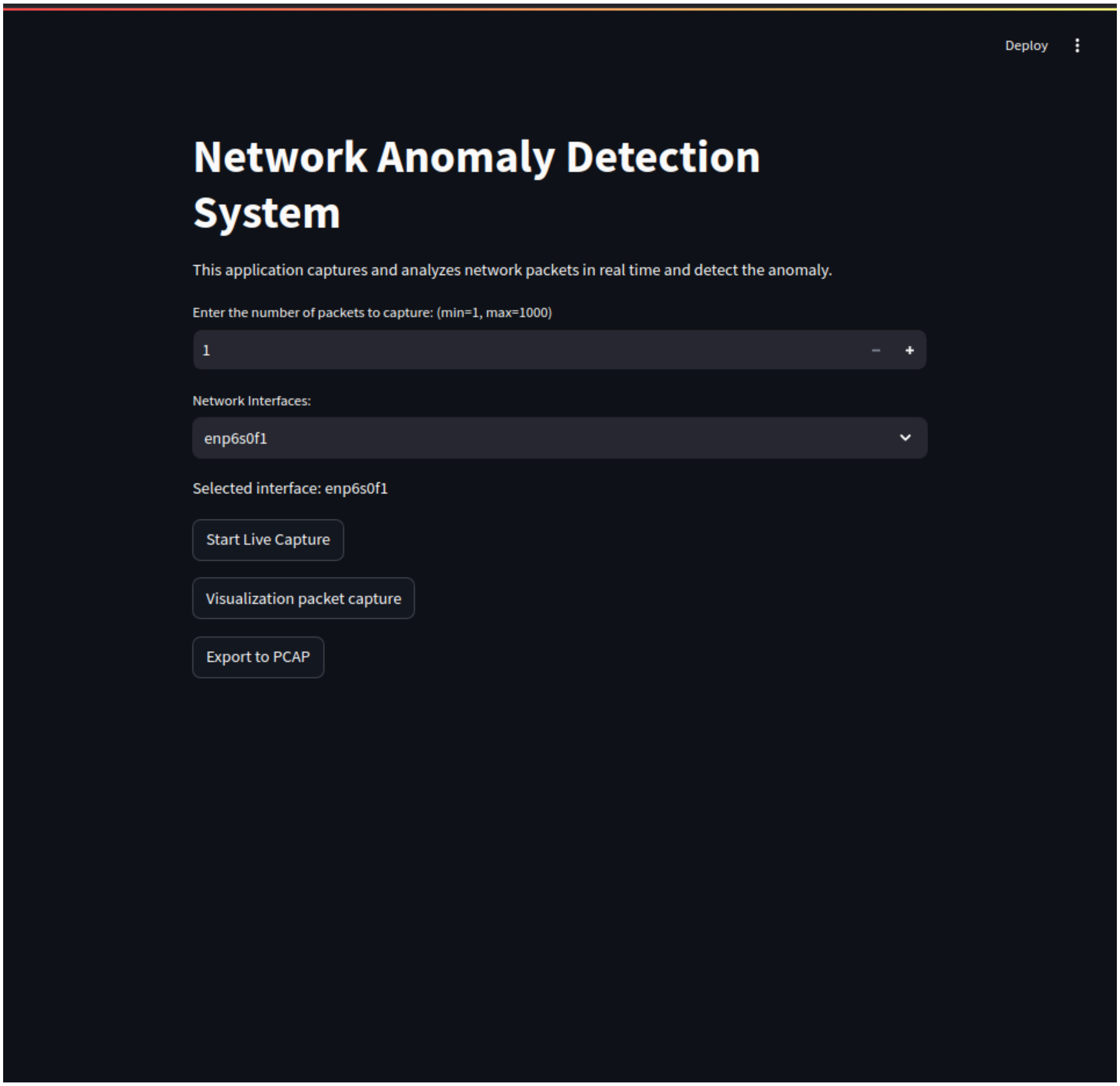
5.2.1.A: Using anaconda to open the GUI and begin using NADS.py.



```
pc-i@PC-I:~$ source ~/anaconda3/bin/activate &&
cd /media/pc-i/KINGSTON/مشروع-التخرج/AnomlyDetection &&
conda activate streamlit &&
streamlit run ./NADS.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.17.53:8501

2025-01-10 19:54:37.860277: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary
is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropr
iate compiler flags.
2025-01-10 19:54:41,487 - INFO - Model loaded successfully with input shape: (None, 16)
```

Figure 8: Unit Testing Results A

5.2.1.B: Main page.



Network Anomaly Detection System

This application captures and analyzes network packets in real time and detect the anomaly.

Enter the number of packets to capture: (min=1, max=1000)

1

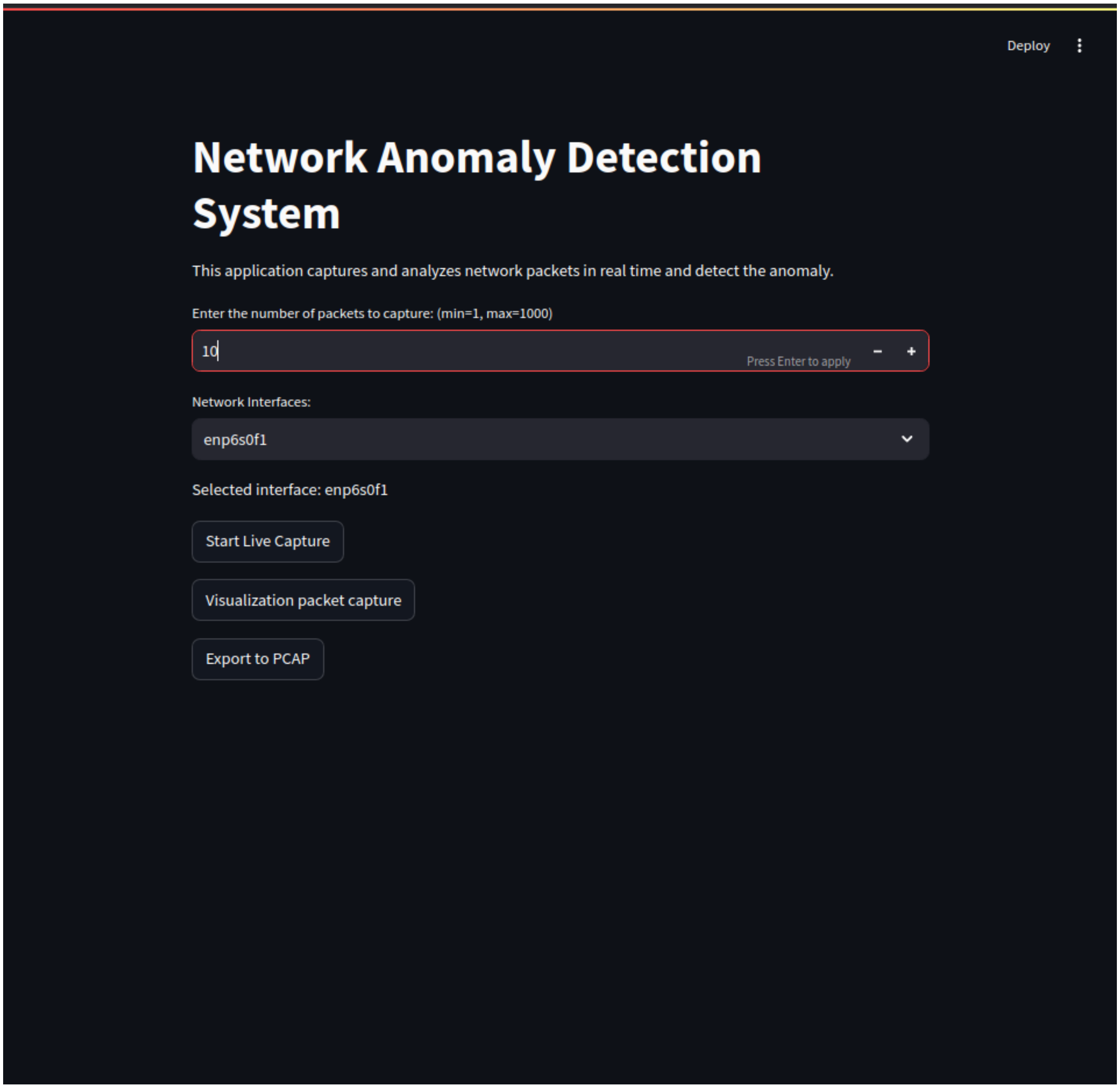
Network Interfaces:

enp6s0f1

Selected interface: enp6s0f1

Figure 9: Unit Testing Results B

5.2.1.C: Entering how many packets to capture and analyze.



Network Anomaly Detection System

This application captures and analyzes network packets in real time and detect the anomaly.

Enter the number of packets to capture: (min=1, max=1000)

10

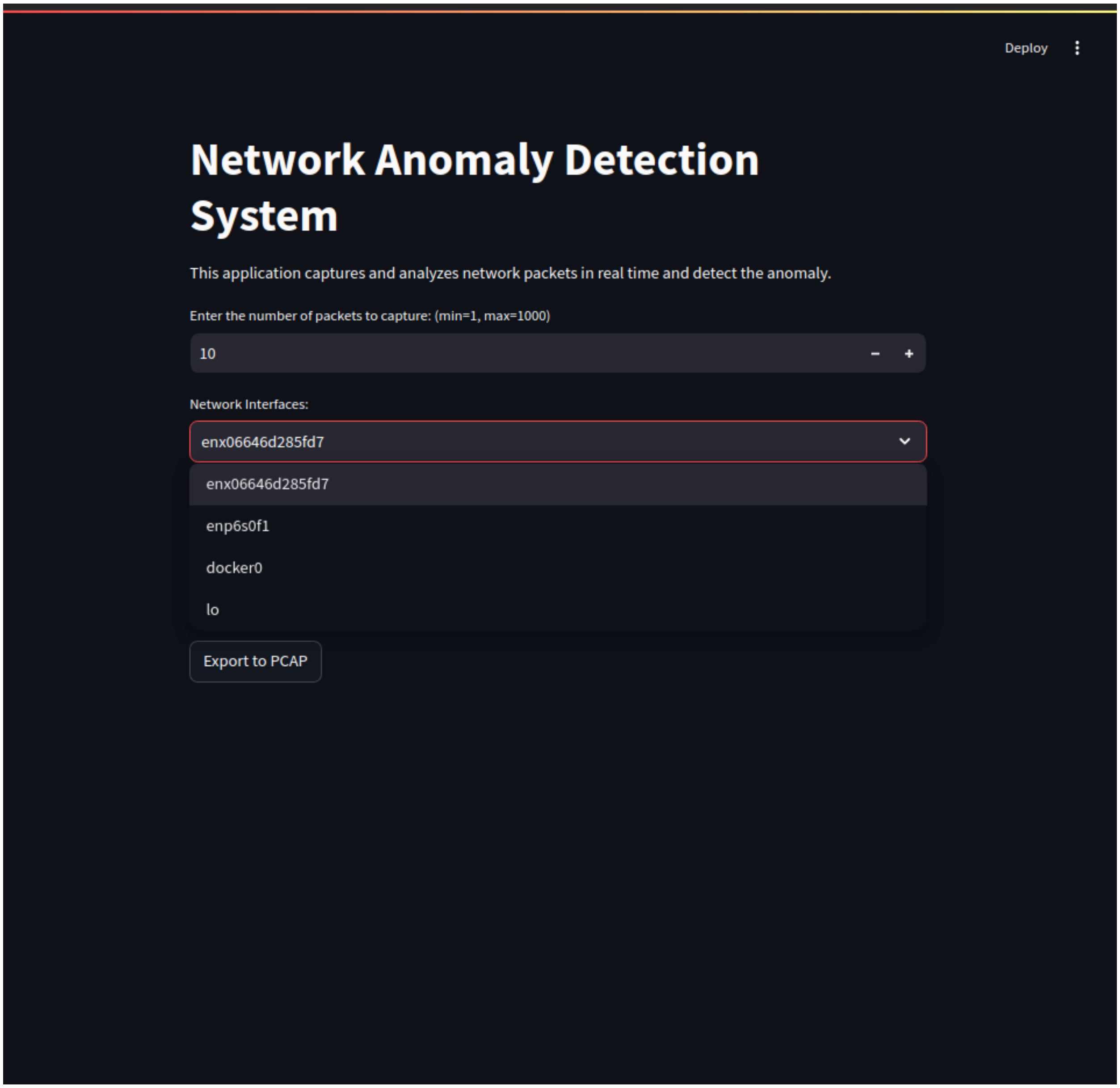
Network Interfaces:

enp6s0f1

Selected interface: enp6s0f1

Figure 10: Unit Testing Results C

5.2.1.D: Selecting an interface to monitor.



Network Anomaly Detection System

This application captures and analyzes network packets in real time and detect the anomaly.

Enter the number of packets to capture: (min=1, max=1000)

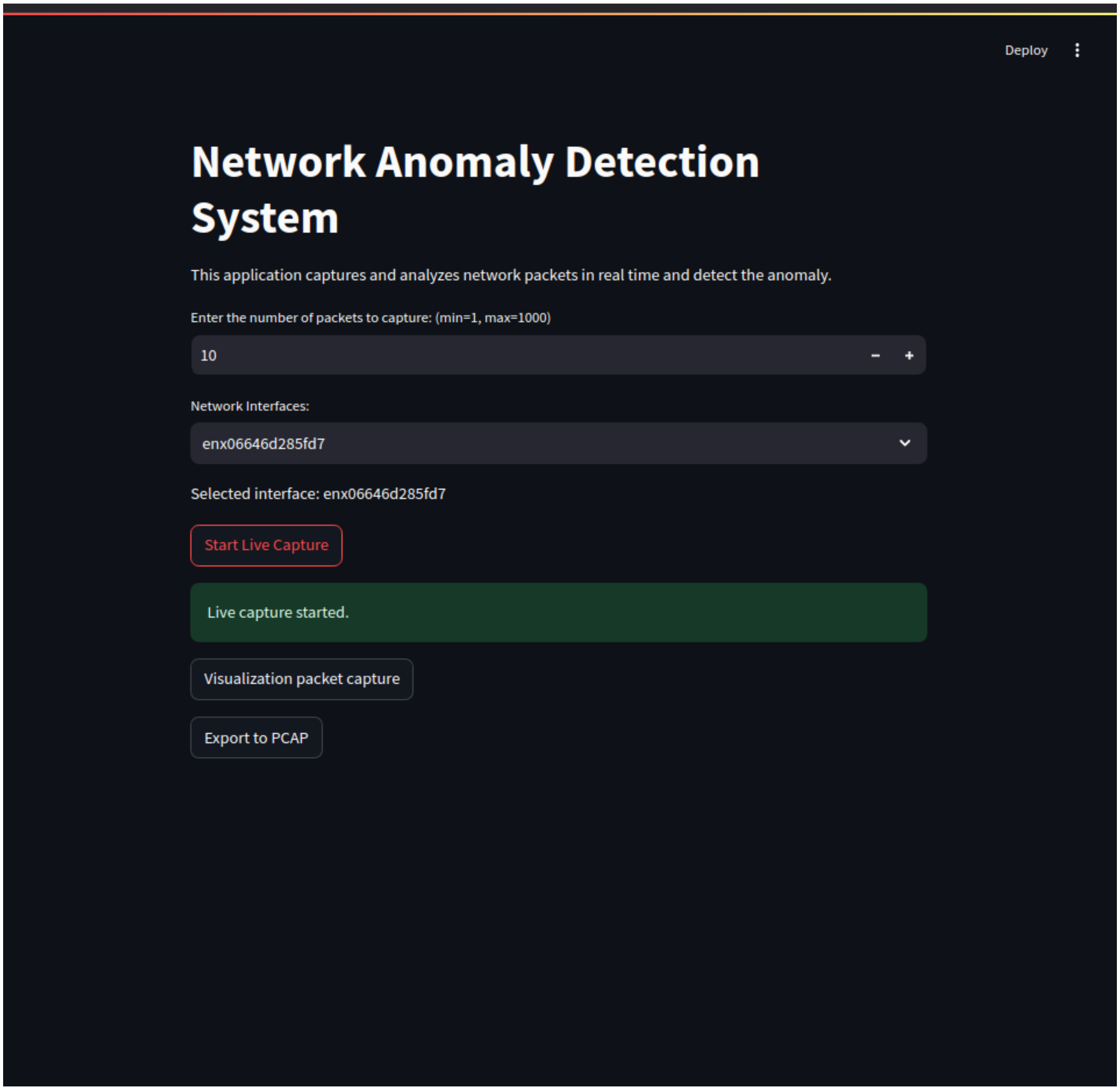
10

Network Interfaces:

enx06646d285fd7

Figure 11: Unit Testing Results D

5.2.1.E: Started the live capture.



Network Anomaly Detection System

This application captures and analyzes network packets in real time and detect the anomaly.

Enter the number of packets to capture: (min=1, max=1000)

10

Network Interfaces:

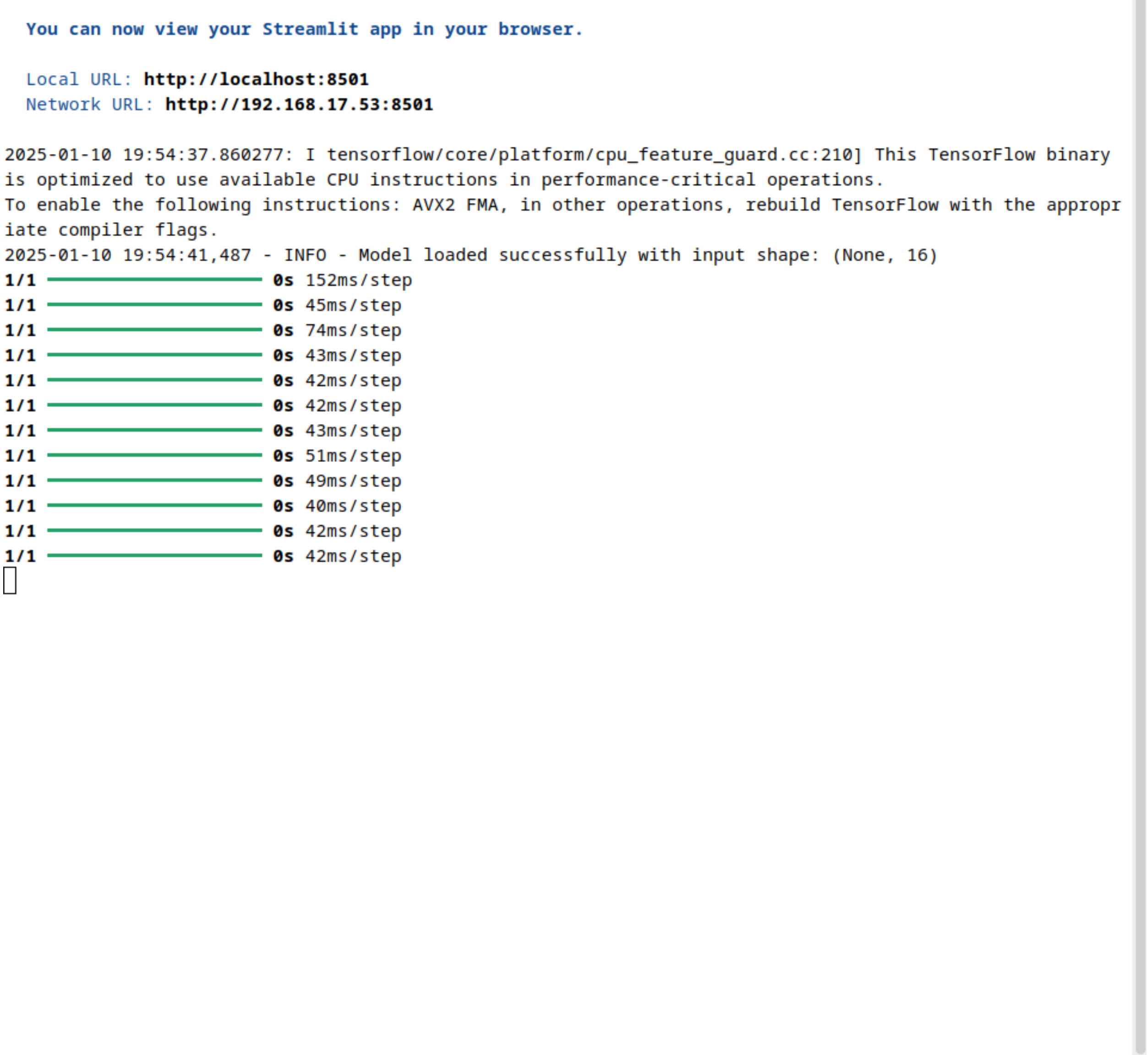
enx06646d285fd7

Selected interface: enx06646d285fd7

Live capture started.

Figure 12: Unit Testing Results E

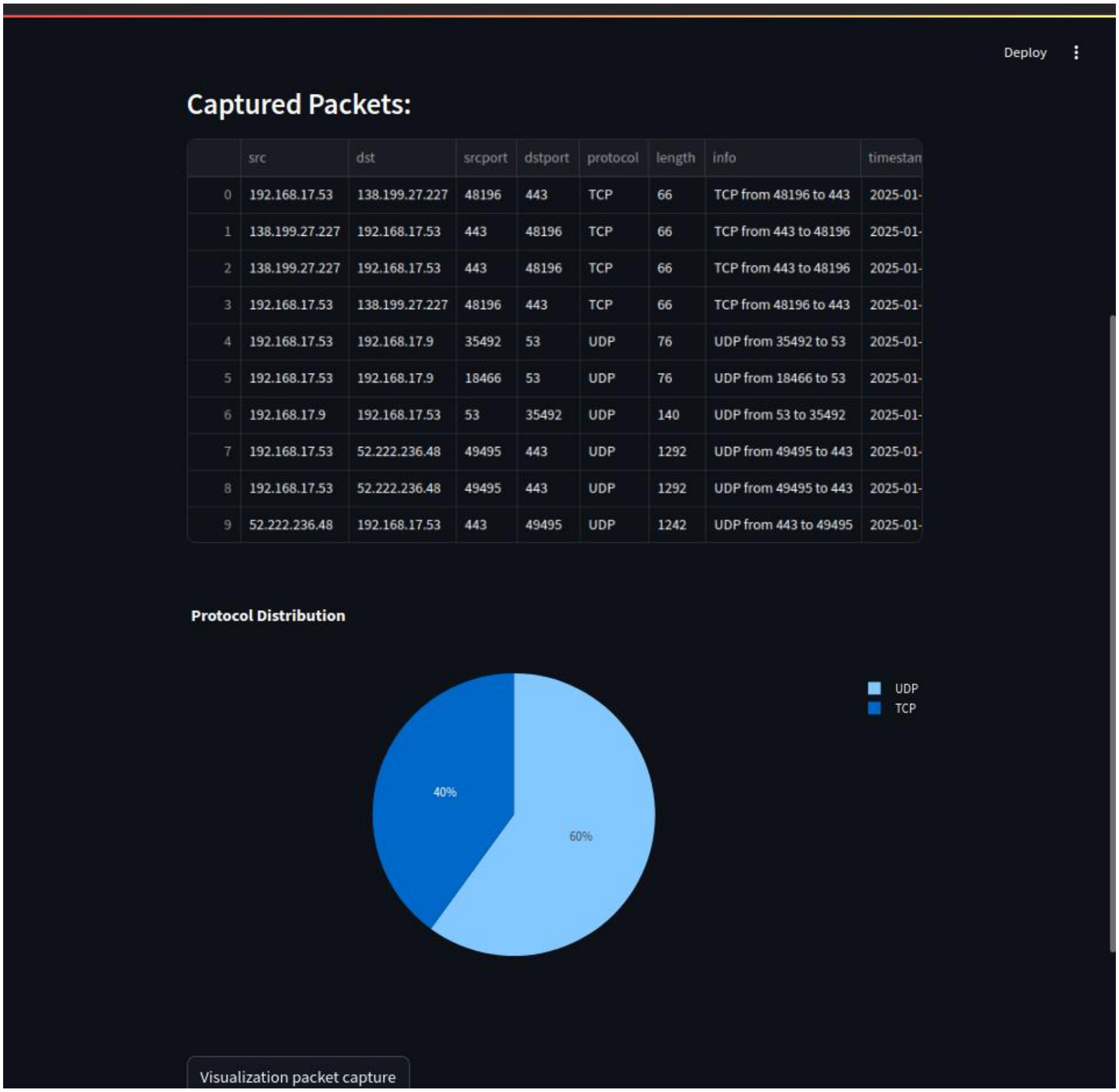
5.2.1.F: The Model is Analyzing the captured packets.



```
  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.17.53:8501

2025-01-10 19:54:37.860277: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary
is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropr
iate compiler flags.
2025-01-10 19:54:41,487 - INFO - Model loaded successfully with input shape: (None, 16)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 152ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
```

Figure 13: Unit Testing Results F

5.2.1.G: Packets have been analyzed by the Model and showing a detailed results about them in a form of a table & a simple graph.

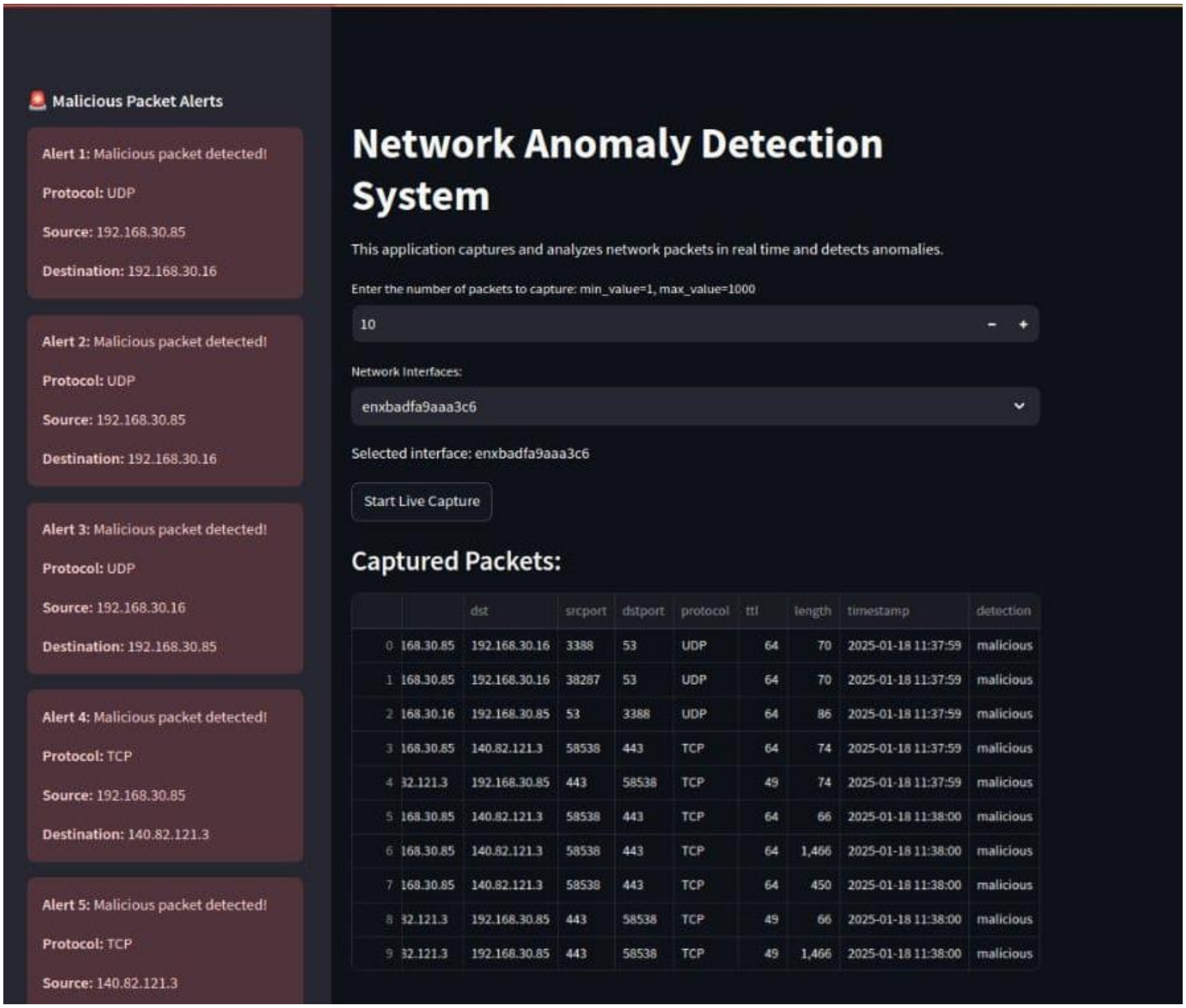


Captured Packets:

	src	dst	srcport	dstport	protocol	length	info	timestan
0	192.168.17.53	138.199.27.227	48196	443	TCP	66	TCP from 48196 to 443	2025-01-
1	138.199.27.227	192.168.17.53	443	48196	TCP	66	TCP from 443 to 48196	2025-01-
2	138.199.27.227	192.168.17.53	443	48196	TCP	66	TCP from 443 to 48196	2025-01-
3	192.168.17.53	138.199.27.227	48196	443	TCP	66	TCP from 48196 to 443	2025-01-
4	192.168.17.53	192.168.17.9	35492	53	UDP	76	UDP from 35492 to 53	2025-01-
5	192.168.17.53	192.168.17.9	18466	53	UDP	76	UDP from 18466 to 53	2025-01-
6	192.168.17.9	192.168.17.53	53	35492	UDP	140	UDP from 53 to 35492	2025-01-
7	192.168.17.53	52.222.236.48	49495	443	UDP	1292	UDP from 49495 to 443	2025-01-
8	192.168.17.53	52.222.236.48	49495	443	UDP	1292	UDP from 49495 to 443	2025-01-
9	52.222.236.48	192.168.17.53	443	49495	UDP	1242	UDP from 443 to 49495	2025-01-

Figure 14: Unit Testing Results G

5.2.1.H: NADS caught a malicious packet and has alerted the user.



🚨 Malicious Packet Alerts

Alert 1: Malicious packet detected!

Protocol: UDP

Source: 192.168.30.85

Destination: 192.168.30.16

Alert 2: Malicious packet detected!

Protocol: UDP

Source: 192.168.30.85

Destination: 192.168.30.16

Alert 3: Malicious packet detected!

Protocol: UDP

Source: 192.168.30.16

Destination: 192.168.30.85

Alert 4: Malicious packet detected!

Protocol: TCP

Source: 192.168.30.85

Destination: 140.82.121.3

Alert 5: Malicious packet detected!

Protocol: TCP

Source: 140.82.121.3

Network Anomaly Detection System

This application captures and analyzes network packets in real time and detects anomalies.

Enter the number of packets to capture: min_value=1, max_value=1000

10

Network Interfaces:

enxbadfa9aaa3c6

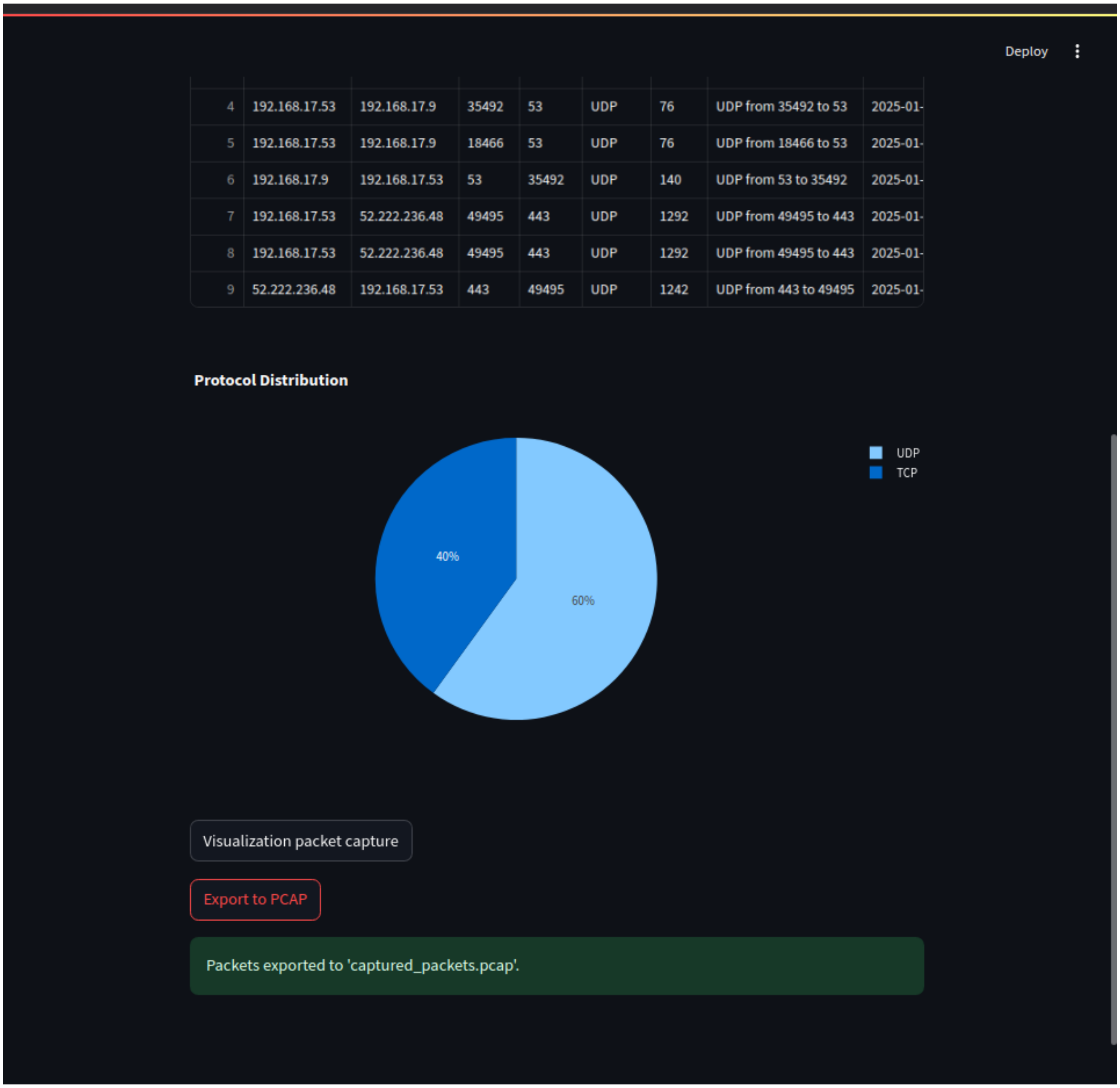
Selected interface: enxbadfa9aaa3c6

Captured Packets:

		dst	srcport	dstport	protocol	ttl	length	timestamp	detection
0	168.30.85	192.168.30.16	3388	53	UDP	64	70	2025-01-18 11:37:59	malicious
1	168.30.85	192.168.30.16	38287	53	UDP	64	70	2025-01-18 11:37:59	malicious
2	168.30.16	192.168.30.85	53	3388	UDP	64	86	2025-01-18 11:37:59	malicious
3	168.30.85	140.82.121.3	58538	443	TCP	64	74	2025-01-18 11:37:59	malicious
4	32.121.3	192.168.30.85	443	58538	TCP	49	74	2025-01-18 11:37:59	malicious
5	168.30.85	140.82.121.3	58538	443	TCP	64	66	2025-01-18 11:38:00	malicious
6	168.30.85	140.82.121.3	58538	443	TCP	64	1,466	2025-01-18 11:38:00	malicious
7	168.30.85	140.82.121.3	58538	443	TCP	64	450	2025-01-18 11:38:00	malicious
8	32.121.3	192.168.30.85	443	58538	TCP	49	66	2025-01-18 11:38:00	malicious
9	32.121.3	192.168.30.85	443	58538	TCP	49	1,466	2025-01-18 11:38:00	malicious

Figure 15: Unit Testing Results H

5.2.1.I: Exporting the analyzed packets as a .pcap file.

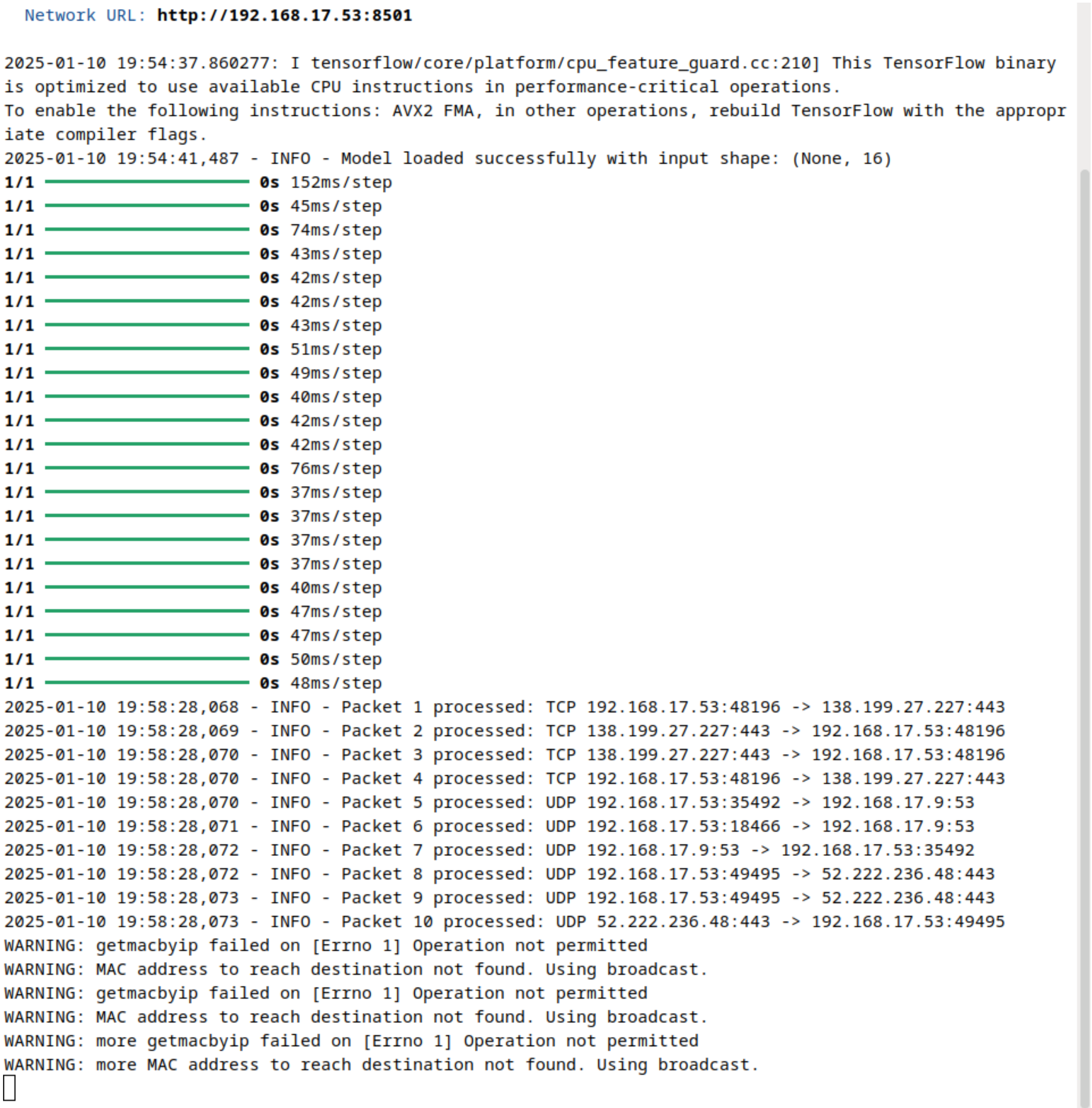


4	192.168.17.53	192.168.17.9	35492	53	UDP	76	UDP from 35492 to 53	2025-01-
5	192.168.17.53	192.168.17.9	18466	53	UDP	76	UDP from 18466 to 53	2025-01-
6	192.168.17.9	192.168.17.53	53	35492	UDP	140	UDP from 53 to 35492	2025-01-
7	192.168.17.53	52.222.236.48	49495	443	UDP	1292	UDP from 49495 to 443	2025-01-
8	192.168.17.53	52.222.236.48	49495	443	UDP	1292	UDP from 49495 to 443	2025-01-
9	52.222.236.48	192.168.17.53	443	49495	UDP	1242	UDP from 443 to 49495	2025-01-

Packets exported to 'captured_packets.pcap'.

Figure 16: Unit Testing Results I

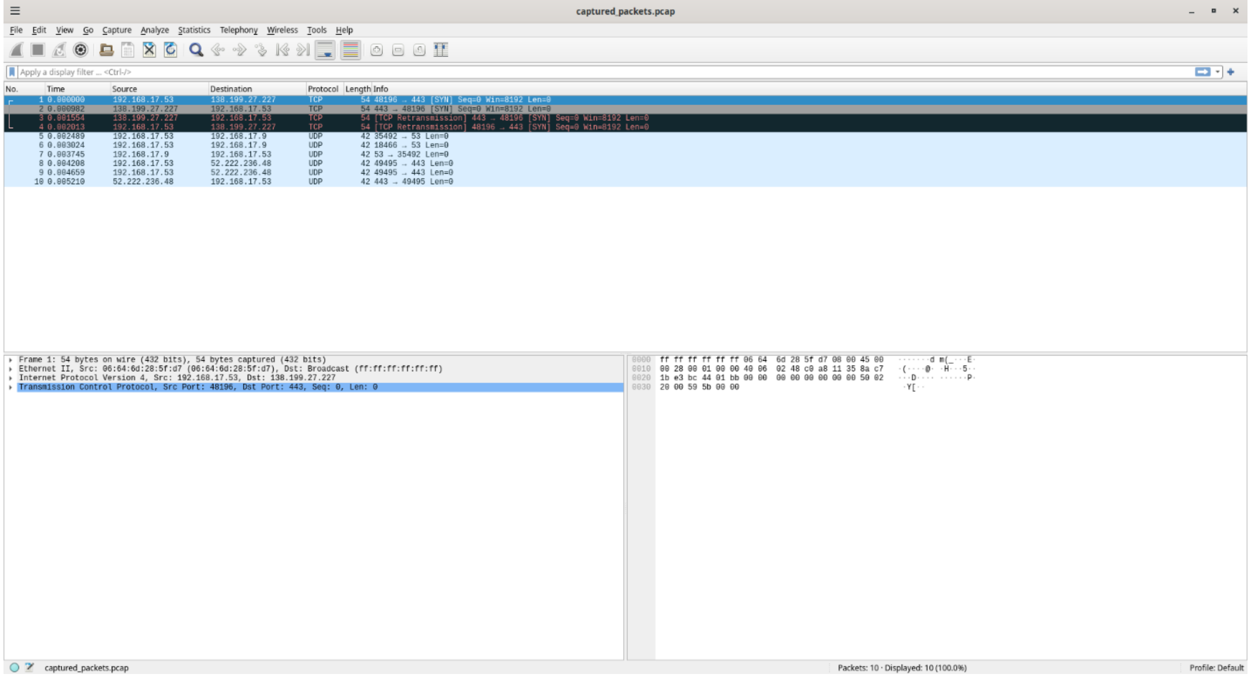
5.2.1.J: the captured packets have been processed and Exported in a .pcap file, ready to be read by Wireshark.



```
  Network URL: http://192.168.17.53:8501

2025-01-10 19:54:37.860277: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary
is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropr
iate compiler flags.
2025-01-10 19:54:41,487 - INFO - Model loaded successfully with input shape: (None, 16)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 152ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
2025-01-10 19:58:28,068 - INFO - Packet 1 processed: TCP 192.168.17.53:48196 -> 138.199.27.227:443
2025-01-10 19:58:28,069 - INFO - Packet 2 processed: TCP 138.199.27.227:443 -> 192.168.17.53:48196
2025-01-10 19:58:28,070 - INFO - Packet 3 processed: TCP 138.199.27.227:443 -> 192.168.17.53:48196
2025-01-10 19:58:28,070 - INFO - Packet 4 processed: TCP 192.168.17.53:48196 -> 138.199.27.227:443
2025-01-10 19:58:28,070 - INFO - Packet 5 processed: UDP 192.168.17.53:35492 -> 192.168.17.9:53
2025-01-10 19:58:28,071 - INFO - Packet 6 processed: UDP 192.168.17.53:18466 -> 192.168.17.9:53
2025-01-10 19:58:28,072 - INFO - Packet 7 processed: UDP 192.168.17.9:53 -> 192.168.17.53:35492
2025-01-10 19:58:28,072 - INFO - Packet 8 processed: UDP 192.168.17.53:49495 -> 52.222.236.48:443
2025-01-10 19:58:28,073 - INFO - Packet 9 processed: UDP 192.168.17.53:49495 -> 52.222.236.48:443
2025-01-10 19:58:28,073 - INFO - Packet 10 processed: UDP 52.222.236.48:443 -> 192.168.17.53:49495
WARNING: getmacbyip failed on [Errno 1] Operation not permitted
WARNING: MAC address to reach destination not found. Using broadcast.
WARNING: getmacbyip failed on [Errno 1] Operation not permitted
WARNING: MAC address to reach destination not found. Using broadcast.
WARNING: more getmacbyip failed on [Errno 1] Operation not permitted
WARNING: more MAC address to reach destination not found. Using broadcast.
```

Figure 17: Unit Testing Results J

5.2.1.K: The .pcap file opened with Wireshark.


captured_packets.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time      Source          Destination     Protocol Length Info
   1 0.000000  192.168.17.53   138.199.27.227  TCP      54 48196 → 443 [SYN] Seq=0 Win=8192 Len=0
   2 0.000982  138.199.27.227  192.168.17.53   TCP      54 443 → 48196 [SYN] Seq=0 Win=8192 Len=0
   3 0.001554  138.199.27.227  192.168.17.53   TCP      54 [TCP Retransmission] 443 → 48196 [SYN] Seq=0 Win=8192 Len=0
   4 0.002013  192.168.17.53   138.199.27.227  TCP      54 [TCP Retransmission] 48196 → 443 [SYN] Seq=0 Win=8192 Len=0
   5 0.002489  192.168.17.53   192.168.17.9    UDP      42 35492 → 53 Len=0
   6 0.003024  192.168.17.53   192.168.17.9    UDP      42 18466 → 53 Len=0
   7 0.003745  192.168.17.9    192.168.17.53   UDP      42 53 → 35492 Len=0
   8 0.004208  192.168.17.53   52.222.236.48   UDP      42 49495 → 443 Len=0
   9 0.004659  192.168.17.53   52.222.236.48   UDP      42 49495 → 443 Len=0
  10 0.005210  52.222.236.48   192.168.17.53   UDP      42 443 → 49495 Len=0
```

```
› Frame 1: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
› Ethernet II, Src: 06:64:6d:28:5f:d7 (06:64:6d:28:5f:d7), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
› Internet Protocol Version 4, Src: 192.168.17.53, Dst: 138.199.27.227
› Transmission Control Protocol, Src Port: 48196, Dst Port: 443, Seq: 0, Len: 0
```

```
0000  ff ff ff ff ff ff 06 64  6d 28 5f d7 08 00 45 00   ······d m(_···E·
0010  00 28 00 01 00 00 40 06  02 48 c0 a8 11 35 8a c7   ·(····@· ·H···5··
0020  1b e3 bc 44 01 bb 00 00  00 00 00 00 00 00 50 02   ···D···· ······P·
0030  20 00 59 5b 00 00                                   ·Y[··
```

Figure 18: Unit Testing Results K

5.2.2 Integration Testing Results

Sprints	**Sprint 1**	**Sprint 2**	**Sprint 3**	**Sprint 4**
Errors	Anomalies were not detected correctly due to missing data inputs.	System logs were incomplete or missing entries.	Failure in communication between modules handling data collection and anomaly detection.	Alerts were not generated for identified anomalies.
Actions	Adjusted the input validation process to ensure all required fields are provided.	Enhanced logging mechanisms to ensure all network events are captured.	Fixed communication protocols between modules and verified data integration.	Modified the alert generation module and resolved coding errors.

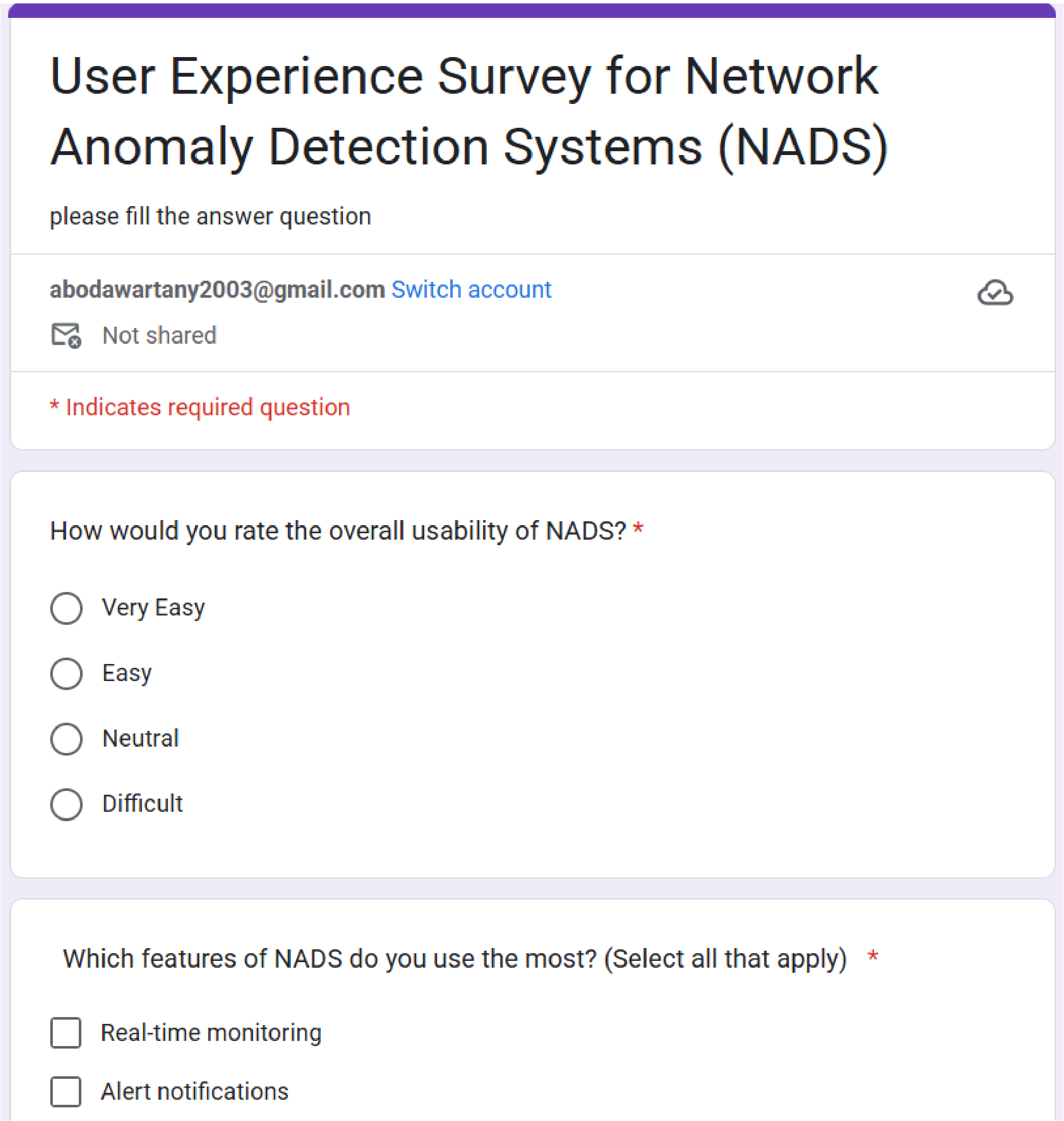
Table 4: Testing Results

5.2.3 System Testing Results

The entire system was tested to examine its functionality and quality. The test showed no problems, and the system functions were fully utilized and synchronized as presented in the demo of this project.

5.2.4 Acceptance System Results

A questionnaire was conducted online, targeting a group of network administrators, IT professionals, and students with varying levels of experience in network management. Participants were asked to interact with the NADS system and then answer the following questions:



User Experience Survey for Network Anomaly Detection Systems (NADS)

please fill the answer question

abodawartany2003@gmail.com Switch account

Not shared

* Indicates required question

How would you rate the overall usability of NADS? *

○ Very Easy

○ Easy

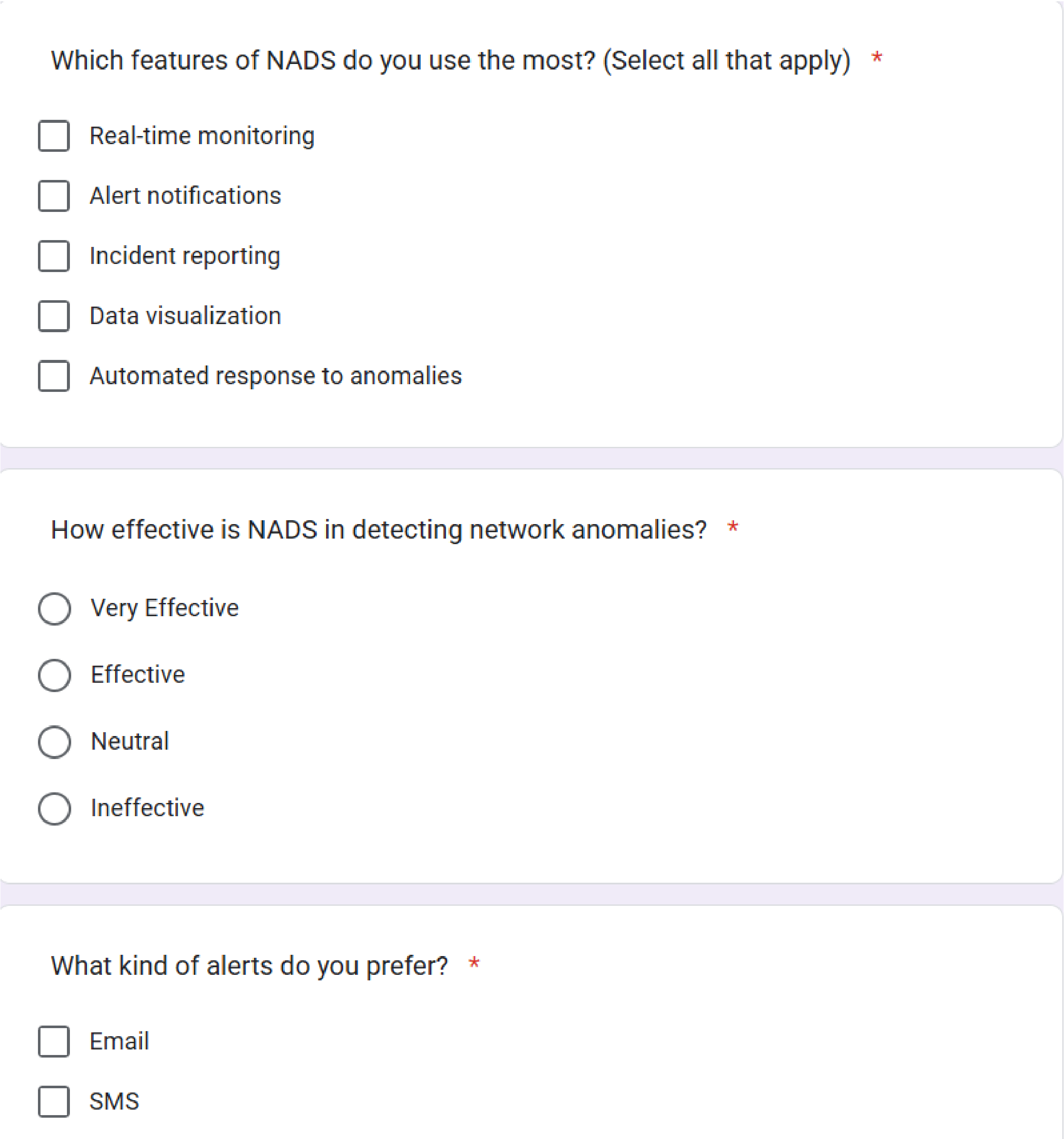
○ Neutral

○ Difficult

Which features of NADS do you use the most? (Select all that apply) *

□ Real-time monitoring

□ Alert notifications



Which features of NADS do you use the most? (Select all that apply) *

□ Real-time monitoring

□ Alert notifications

□ Incident reporting

□ Data visualization

□ Automated response to anomalies

How effective is NADS in detecting network anomalies? *

○ Very Effective

○ Effective

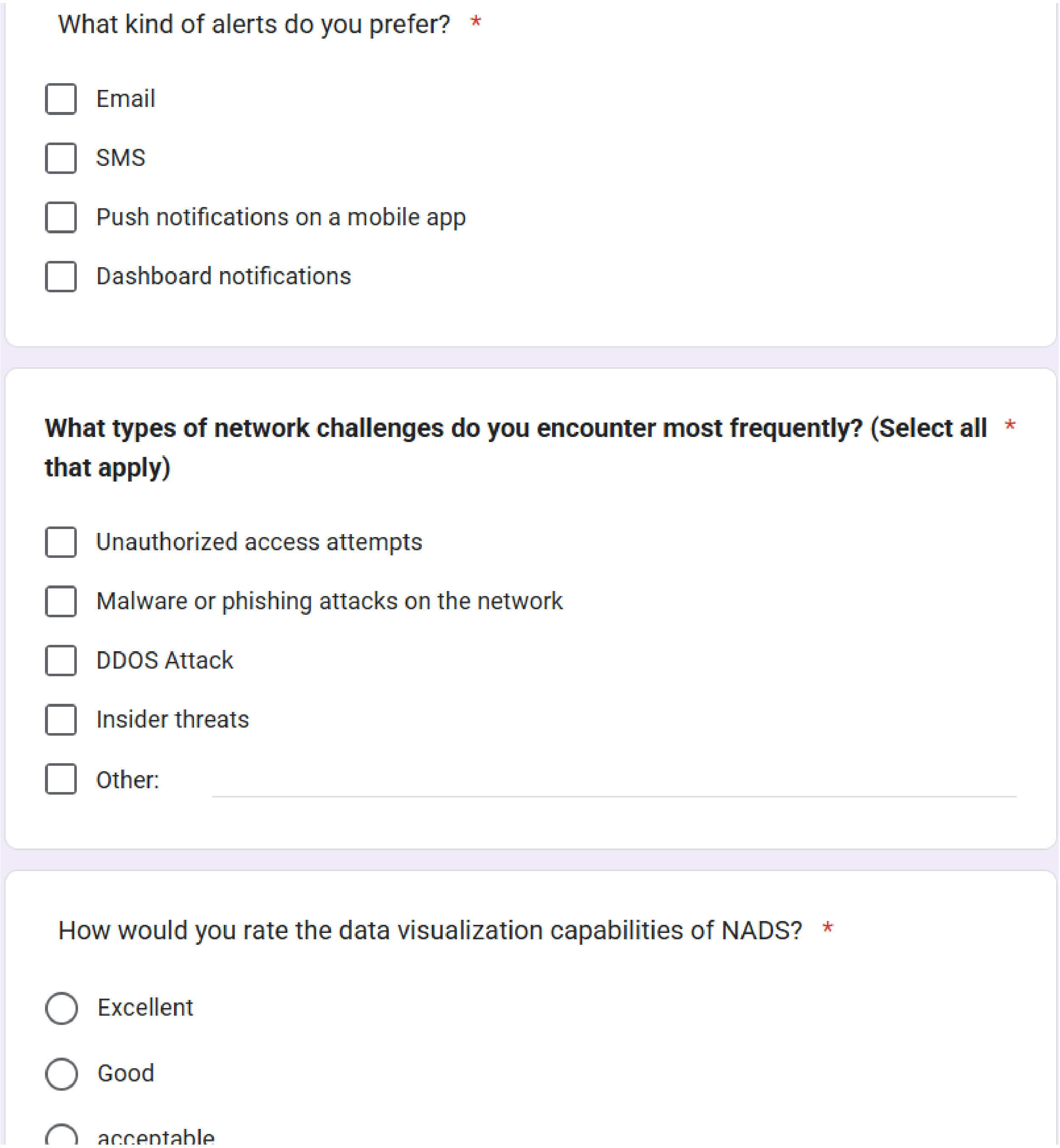
○ Neutral

○ Ineffective

What kind of alerts do you prefer? *

□ Email

□ SMS



What kind of alerts do you prefer? *

□ Email

□ SMS

□ Push notifications on a mobile app

□ Dashboard notifications

What types of network challenges do you encounter most frequently? (Select all that apply) *

□ Unauthorized access attempts

□ Malware or phishing attacks on the network

□ DDOS Attack

□ Insider threats

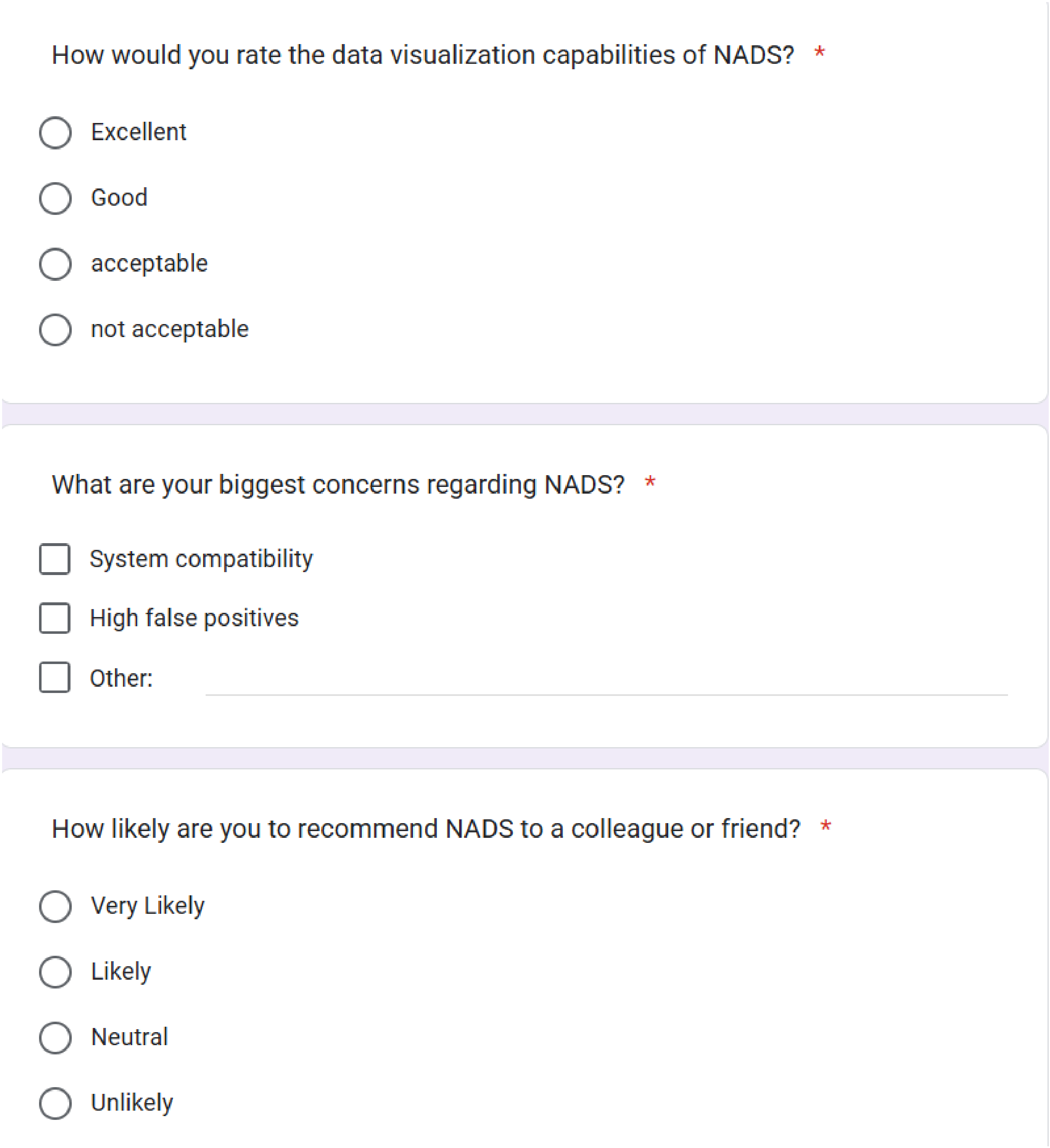
□ Other: ____________________

How would you rate the data visualization capabilities of NADS? *

○ Excellent

○ Good

○ acceptable



How would you rate the data visualization capabilities of NADS? *

○ Excellent

○ Good

○ acceptable

○ not acceptable

What are your biggest concerns regarding NADS? *

□ System compatibility

□ High false positives

□ Other: ____________________

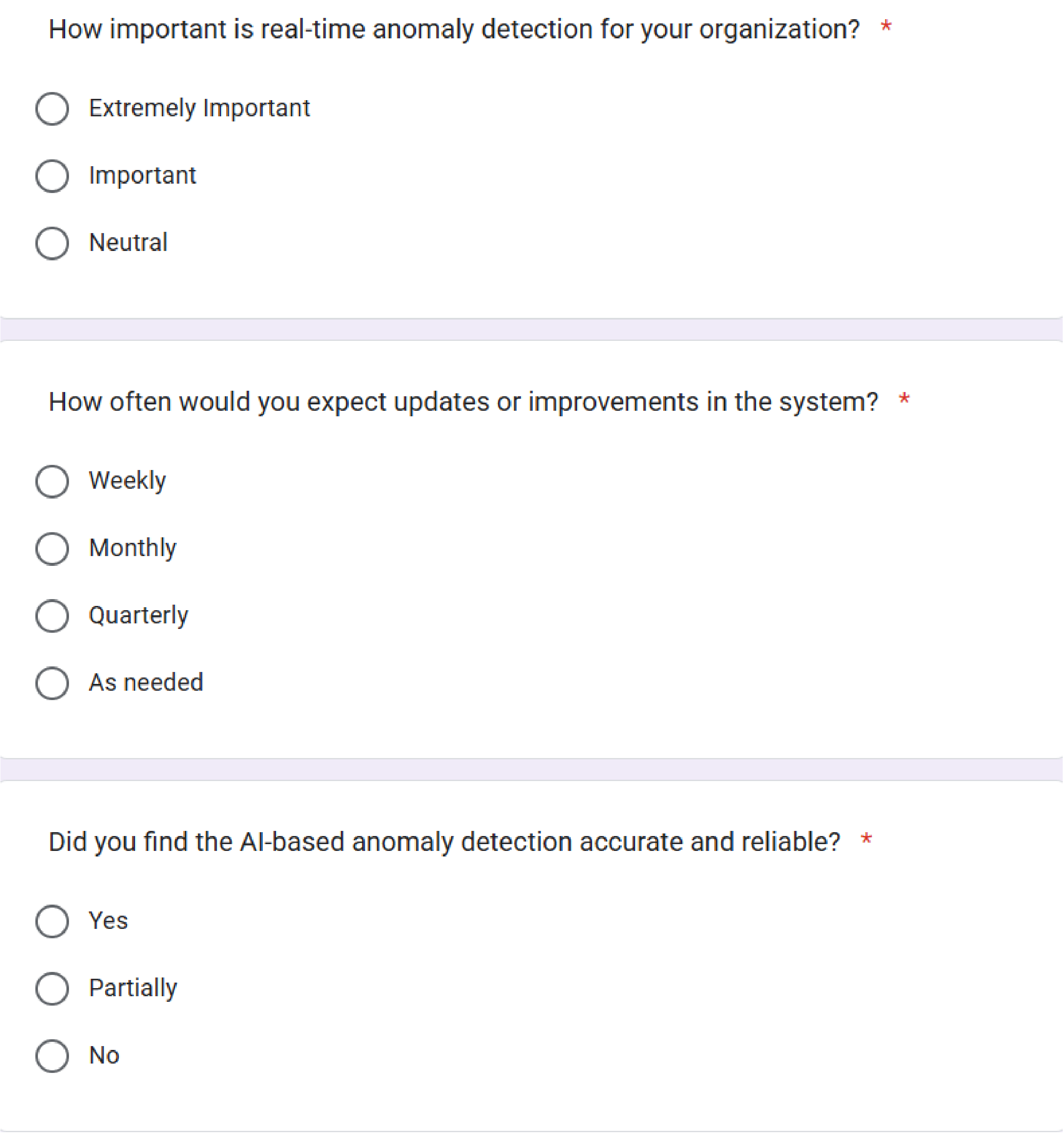
How likely are you to recommend NADS to a colleague or friend? *

○ Very Likely

○ Likely

○ Neutral

○ Unlikely



How important is real-time anomaly detection for your organization? *

○ Extremely Important

○ Important

○ Neutral

How often would you expect updates or improvements in the system? *

○ Weekly

○ Monthly

○ Quarterly

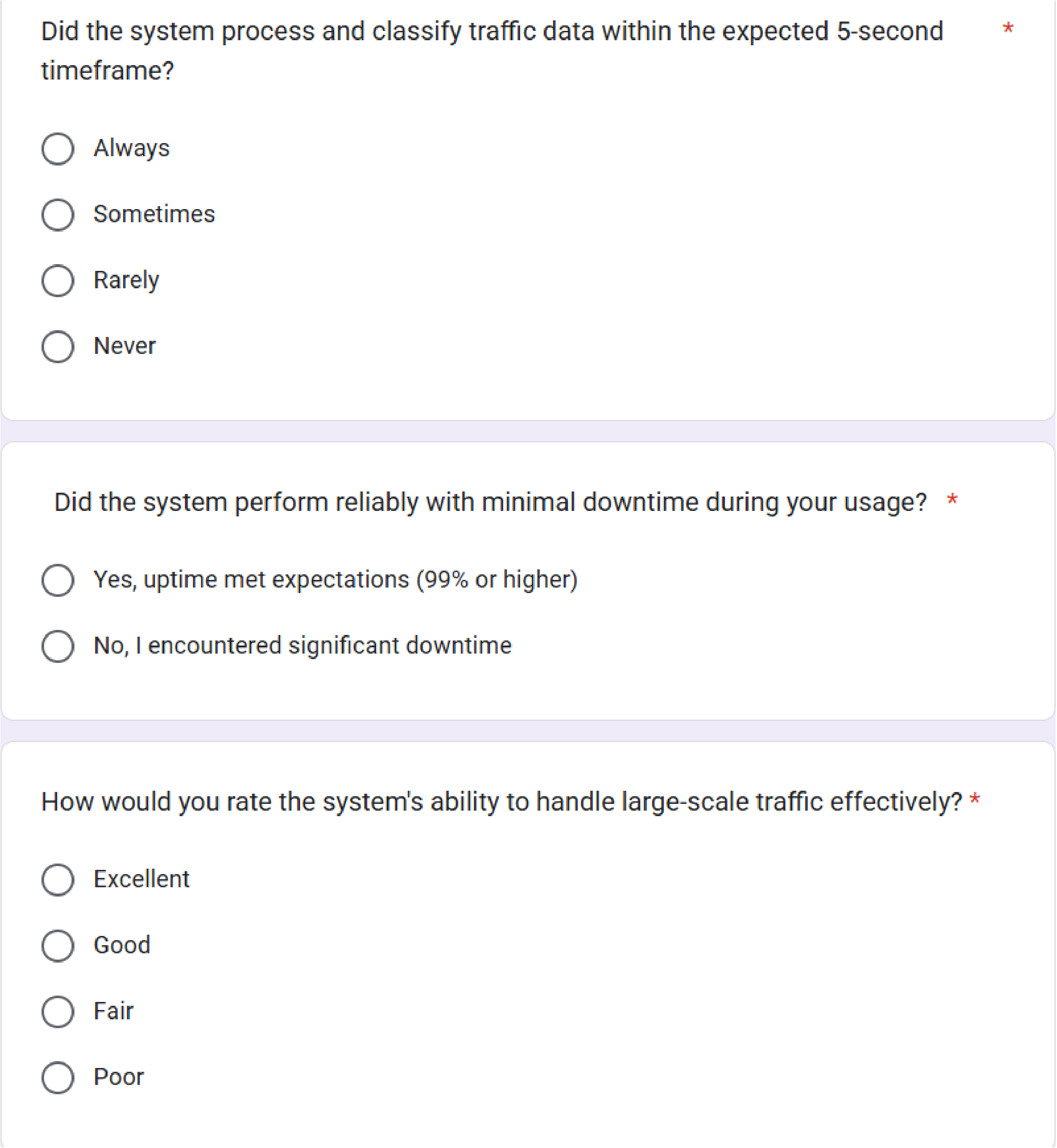
○ As needed

Did you find the AI-based anomaly detection accurate and reliable? *

○ Yes

○ Partially

○ No



Did the system process and classify traffic data within the expected 5-second timeframe? *

○ Always

○ Sometimes

○ Rarely

○ Never

Did the system perform reliably with minimal downtime during your usage? *

○ Yes, uptime met expectations (99% or higher)

○ No, I encountered significant downtime

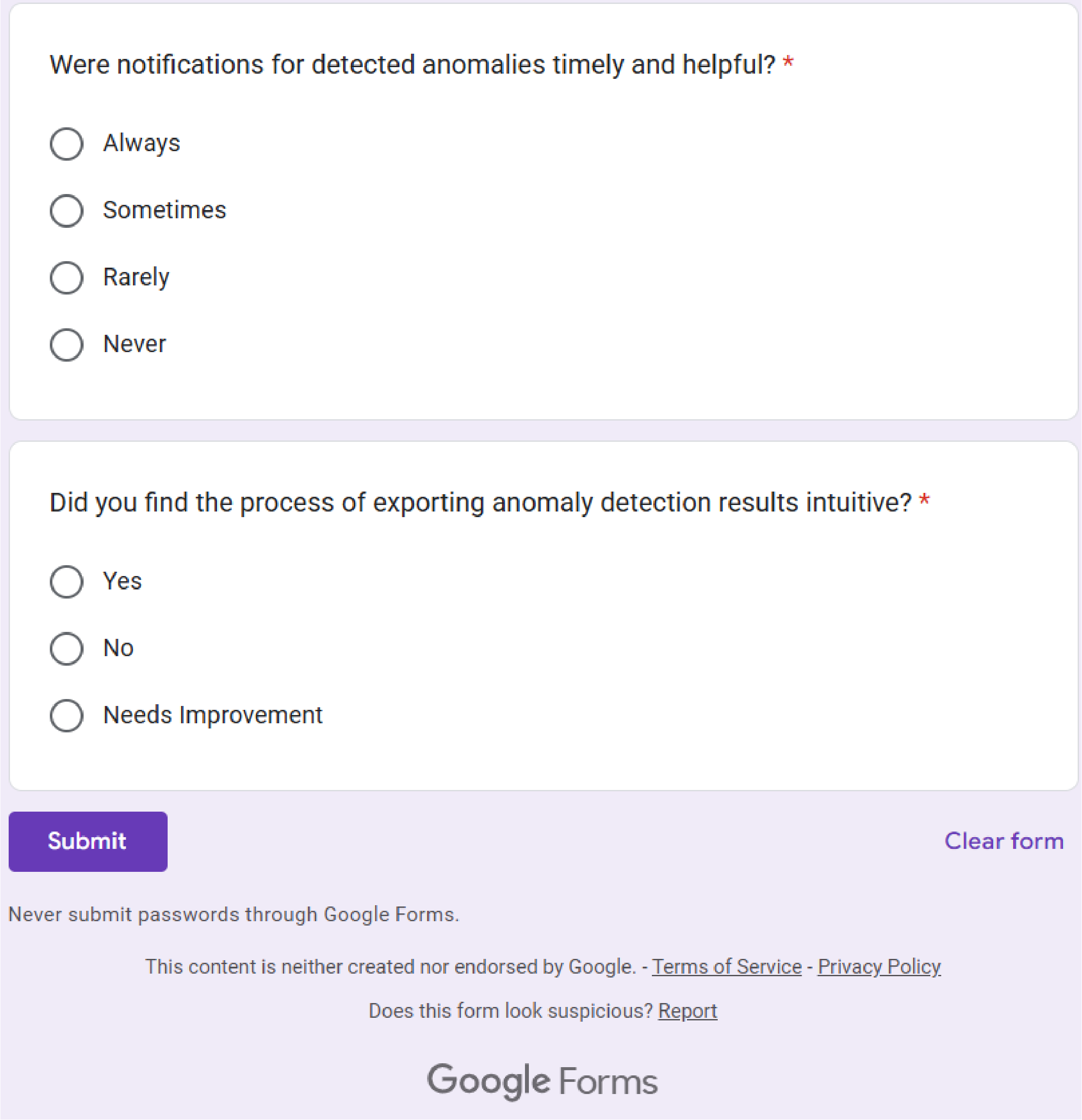
How would you rate the system's ability to handle large-scale traffic effectively? *

○ Excellent

○ Good

○ Fair

○ Poor



Were notifications for detected anomalies timely and helpful? *

○ Always

○ Sometimes

○ Rarely

○ Never

Did you find the process of exporting anomaly detection results intuitive? *

○ Yes

○ No

○ Needs Improvement

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. - Terms of Service - Privacy Policy

Does this form look suspicious? Report

5.3 Discussion and Evaluation

The testing results highlight that while the NADS performs well in detecting anomalies and integrating components, issues with scalability and false positives require optimization. Real-time alerts and reporting met expectations, ensuring usability and reliability for end-users. Further improvements in detection accuracy and performance under high traffic are essential to achieve system robustness.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Overview

This chapter concludes the project by summarizing the main findings, reflecting on the achieved objectives, highlighting the main contributions of the work, discussing its limitations, and proposing directions for future work and conclusion.

6.2 Summary about the Project

This project was dedicated to designing and implementing a real-time network anomaly detection system using machine learning techniques. By leveraging systematic data collection, preprocessing, and feature engineering, the system demonstrated its ability to identify abnormal network behaviors effectively. A deep learning-based model was integrated into the detection pipeline to analyze network traffic patterns, achieving reliable anomaly detection. This system presents a practical and scalable solution to improve the security of modern network environments, offering a proactive approach to threat detection.

6.3 Achieved Objectives

The project successfully accomplished its primary goals, including:

1. Data Preprocessing: A comprehensive pipeline was implemented to clean, normalize, and transform raw network traffic data into a format suitable for analysis.
2. Feature Selection and Engineering: Relevant features were identified and optimized to enhance model performance and reduce computational overhead.
3. Model Development and Training: Machine learning models, including a deep learning-based architecture, were developed, trained, and evaluated. The models achieved high accuracy in detecting anomalies, demonstrating their potential to identify threats in real-time network environments.
4. System Integration: The developed models were integrated into a user-friendly system that provides network administrators with alerts and visualization tools for effective monitoring and response.

6.4 Main Contributions of the Work

This project made several noteworthy contributions to the field of network security:

1. Real-Time Anomaly Detection: The proposed system offers a proactive, real-time approach to identifying suspicious network activity, reducing potential damage from cyberattacks.
2. Enhanced Detection Accuracy: By incorporating advanced deep learning techniques, the system outperformed traditional rule-based detection methods, achieving improved accuracy and robustness.
3. Scalability and Adaptability: The system can be adapted to various network environments and extended to handle diverse datasets, making it suitable for both small-scale and enterprise-level applications.
4. Tool for Administrators: The visualization and alerting modules provide actionable insights, enabling network administrators to respond quickly and efficiently to detected anomalies.

6.5 Limitations

Despite its success, the project faced several limitations:

1. Labeled Data Availability: The scarcity of high-quality, labeled datasets limited the model's training and evaluation. Generating labeled data remains a significant challenge in network security research.
2. High-Dimensional Data Handling: Managing the complexity of high-dimensional network traffic data required trade-offs between model performance and computational efficiency.
3. False Positives: While the system achieved high detection accuracy, minimizing false positives remains an area for improvement to reduce unnecessary alerts.
4. Scalability Challenges: Handling extremely large and dynamic network environments revealed limitations in system performance, highlighting the need for further optimization.

6.6 Future Work

Future research and development efforts could address the limitations and enhance the capabilities of the proposed system:

1. Improved Data Labeling: Leveraging advanced data annotation techniques, including semi-supervised and unsupervised learning, can help create larger and more diverse labeled datasets for training.
2. Feature Selection Optimization**:** Employing automated feature selection methods, such as genetic algorithms or reinforcement learning, can improve model accuracy and reduce computational requirements.
3. Model Optimization**:** Exploring more advanced machine learning architectures, such as transformer-based models, could improve anomaly detection while reducing false positives.
4. Scalability Improvements: Developing distributed computing techniques and incorporating edge computing could help the system handle large-scale network environments efficiently.
5. Adaptive Learning: Integrating online learning mechanisms would enable the system to adapt to new network behaviors and emerging threats dynamically.
6. Broader Application: Expanding the system to incorporate additional use cases, such as IoT security or cloud infrastructure monitoring, would increase its applicability across various domains.

6.7 Conclusion

In conclusion, this project successfully developed a machine learning-based network anomaly detection system that addresses the challenges of real-time threat identification. The system's robust performance highlights its potential for improving network security in diverse environments. However, the identified limitations provide clear opportunities for future research, which could refine and expand upon this work. Ultimately, this project underscores the critical role of machine learning in advancing network security and paves the way for more resilient and adaptive security systems.

REFERENCES

1-Enhanced anomaly traffic detection framework using BiGAN and contrastive learning
https://cybersecurity.springeropen.com/articles/10.1186/s42400-024-00297-7
(accessed and used from 16/10 to 20/12)

2- A comprehensive survey of anomaly detection techniques for high dimensional big data
https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00320-x
(accessed and used from 16/10 to 20/12)

3 LUFlow Network Intrusion Detection Data Set
https://www.kaggle.com/datasets/mryanm/luflow-network-intrusion-detection-data-set
(accessed and used from 16/10 to 5/1)

4- Introduction to Keras for engineers
https://keras.io/getting_started/intro_to_keras_for_engineers/
(accessed and used from 16/10 to 5/1)

5- Python wrapper for tshark, allowing python packet parsing using wireshark dissectors
https://github.com/KimiNewt/pyshark/
(accessed and used from 16/11 to 9/1)

6- User Experience Survey for Network Anomaly Detection Systems (NADS)
https://docs.google.com/forms/d/e/1FAIpQLScqeIQ_e066uJiLfP9E0BHf8xE1d6VCvipuXd6DWS9RdQM27w/viewform?usp=header
(accessed and used from 10/12 to 9/1)

7- Used for dataset and features references
https://github.com/StopDDoS/packet-captures
(accessed and used from 16/11 to 9/1)

8- Used for dataset and features references
https://github.com/mayanknauni/Wireshark-Capture-Anomaly-Detection/blob/main/analyzed_capture.csv
(accessed and used from 16/10 to 7/1)

9- General notes and some helpful codes to better the project's code
https://www.kaggle.com/code/dimaraed/deep-ff/notebook
(accessed and used from 16/10 to 9/1)

10-The attack model example to know specific features and errors in the model
https://github.com/bibs2091/Anomaly-detection-system/blob/master/model/attack_model.joblib
(accessed and used from 9/12 to 9/1)

11-heavily used as reference for the model training code such as correct buggy code
https://www.kaggle.com/code/arpitmathur24/luflow22-ml-notebook#Neural-Network
(accessed and used from 16/10 to 9/1)

Appendix A:

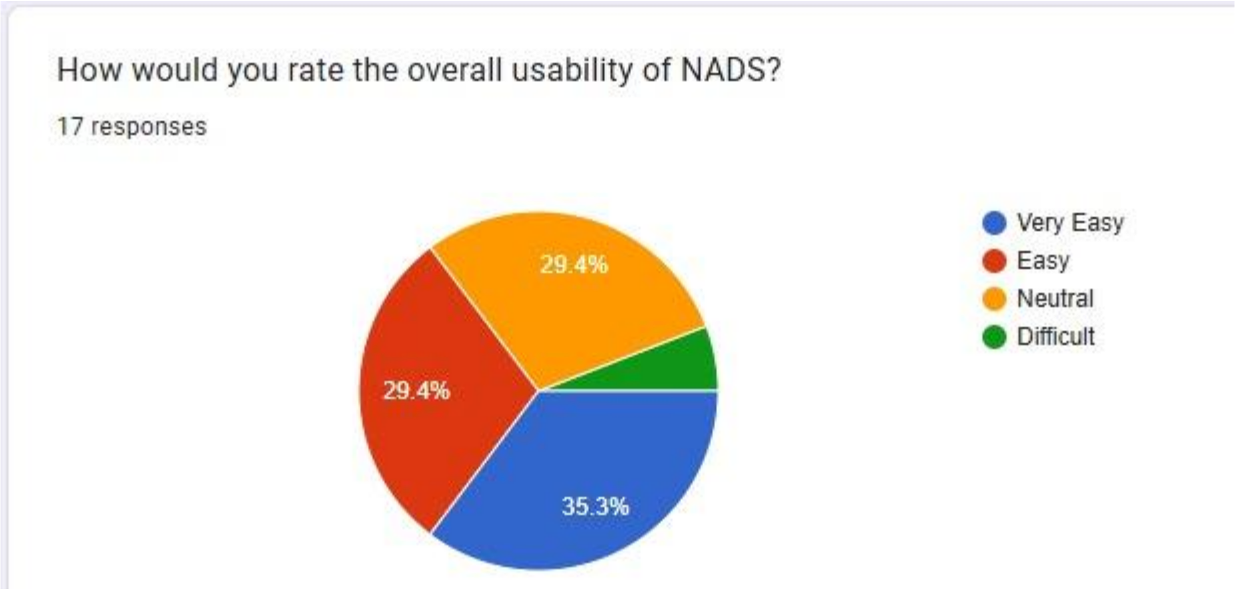
GUI code using Streamlit with Python:
https://drive.google.com/file/d/1_uUeyMeijaFkUxb6r5qSN_7144k3nopM/view?usp=drive_link

Appendix B:

AI Training code using TensorFlow to train the model on the dataset described in Chapter 3:
https://drive.google.com/file/d/112lMStGnC7LUr5CoawULdOfXgWBI0SAy/view?usp=drive_link

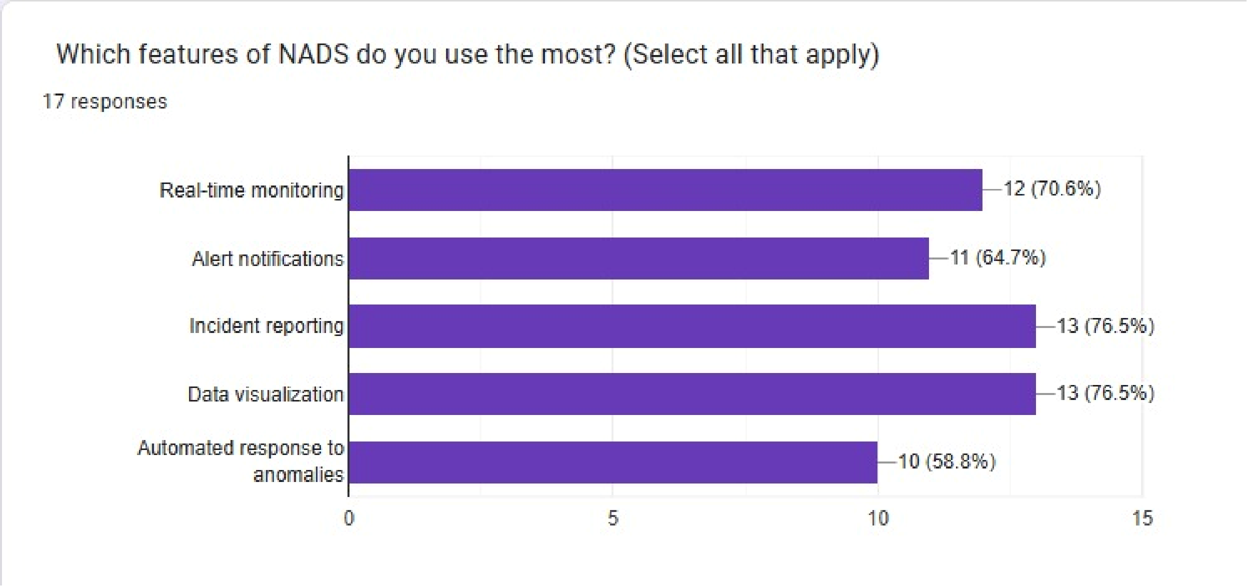
Appendix C:

The questionnaire results from different persons:



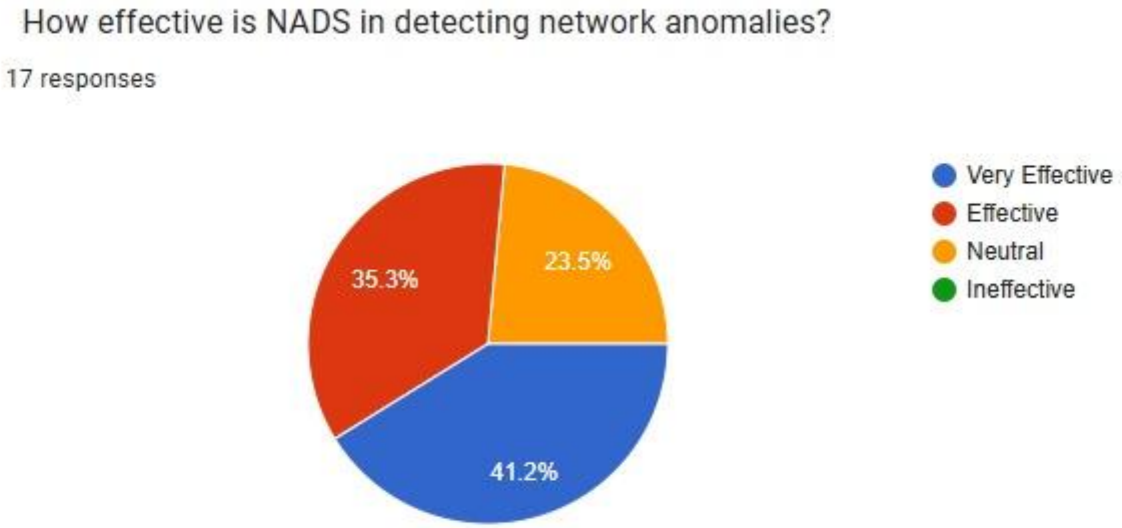
How would you rate the overall usability of NADS?

17 responses



Which features of NADS do you use the most? (Select all that apply)

17 responses

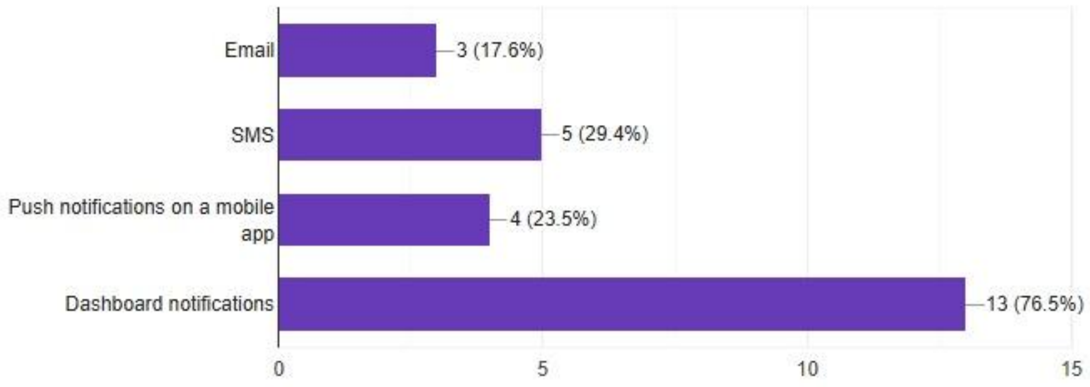


How effective is NADS in detecting network anomalies?

17 responses

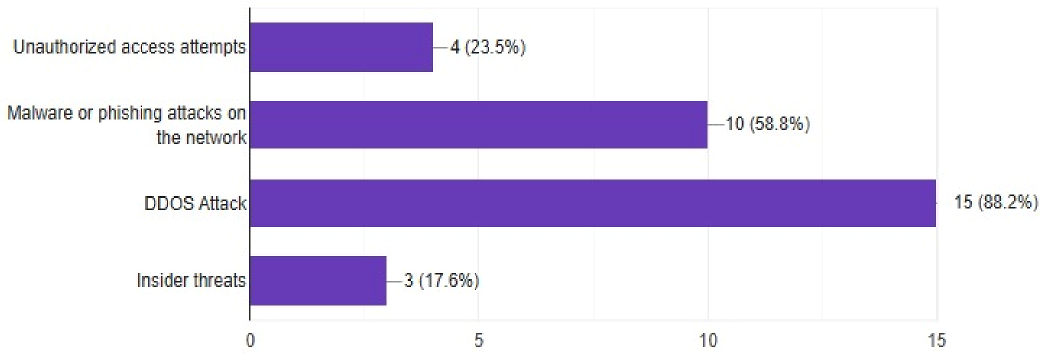
What kind of alerts do you prefer?

17 responses



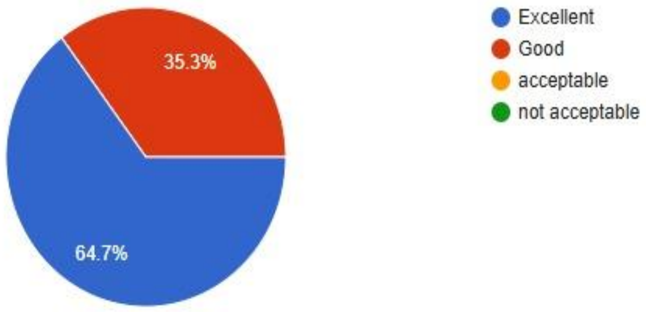
What types of network challenges do you encounter most frequently? (Select all that apply)

17 responses



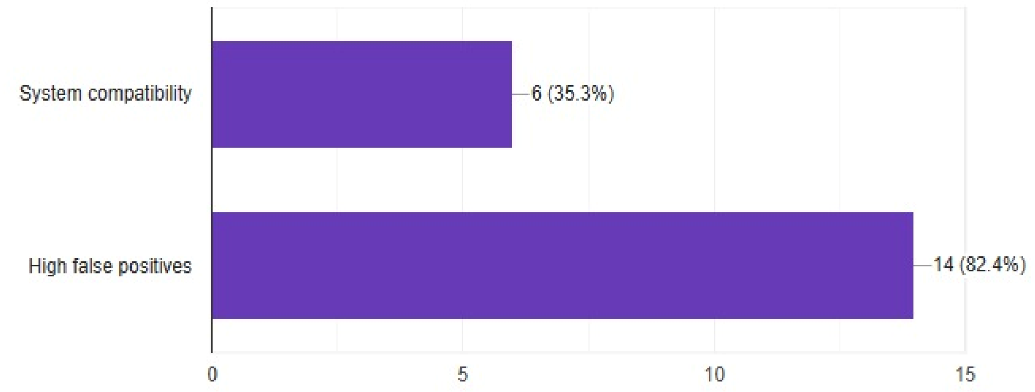
How would you rate the data visualization capabilities of NADS?

17 responses



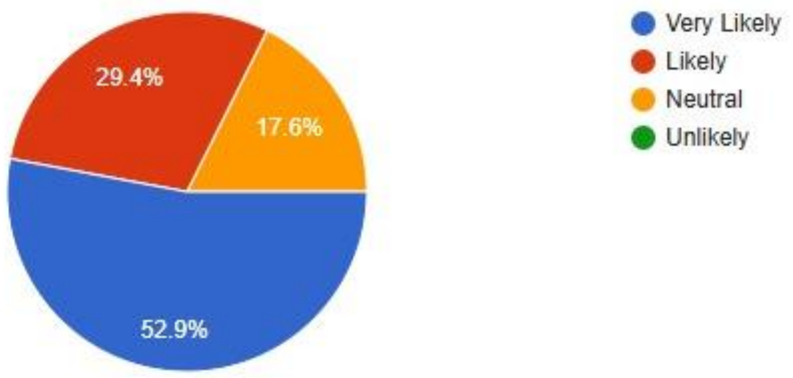
What are your biggest concerns regarding NADS?

17 responses



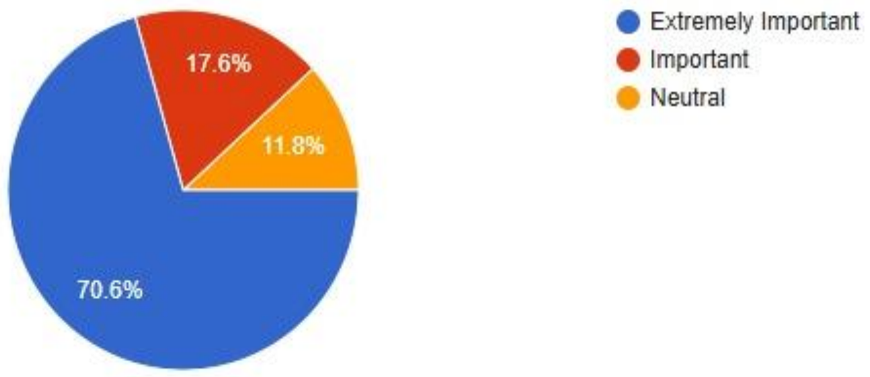
How likely are you to recommend NADS to a colleague or friend?

17 responses



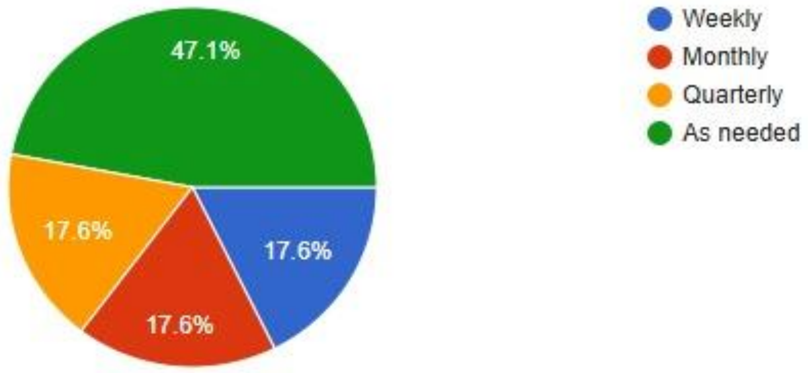
How important is real-time anomaly detection for your organization?

17 responses



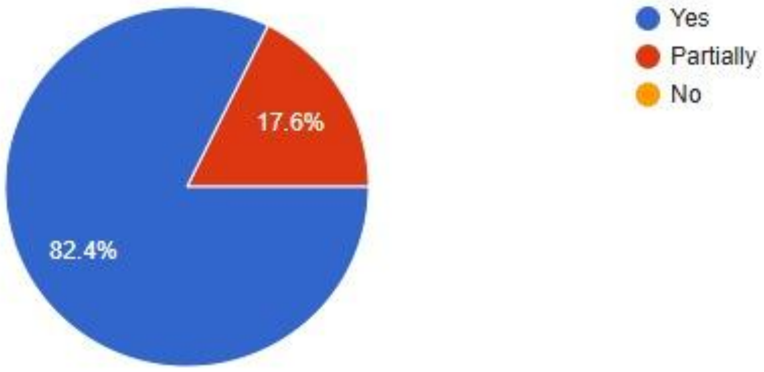
How often would you expect updates or improvements in the system?

17 responses



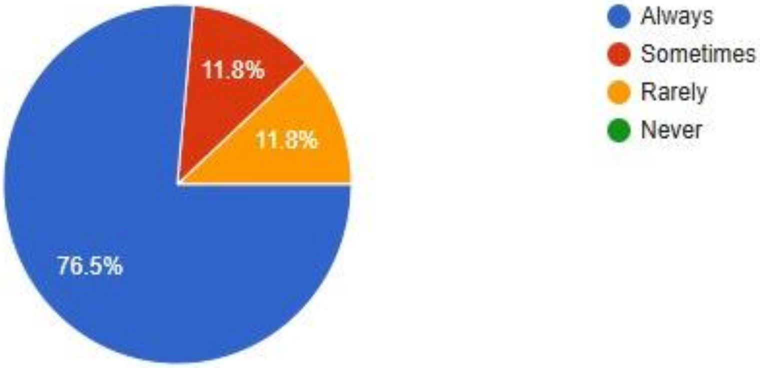
Did you find the AI-based anomaly detection accurate and reliable?

17 responses



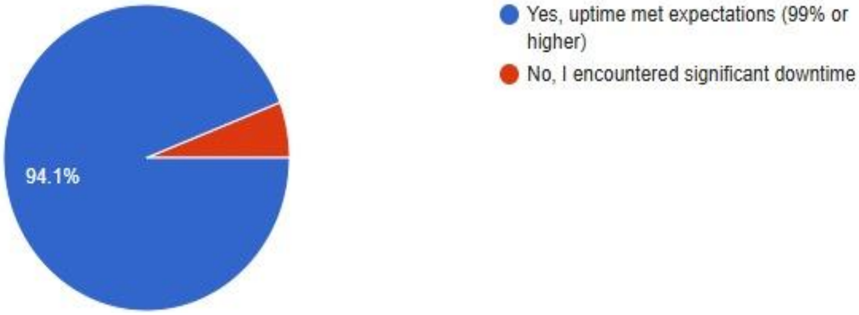
Did the system process and classify traffic data within the expected 5-second timeframe?

17 responses



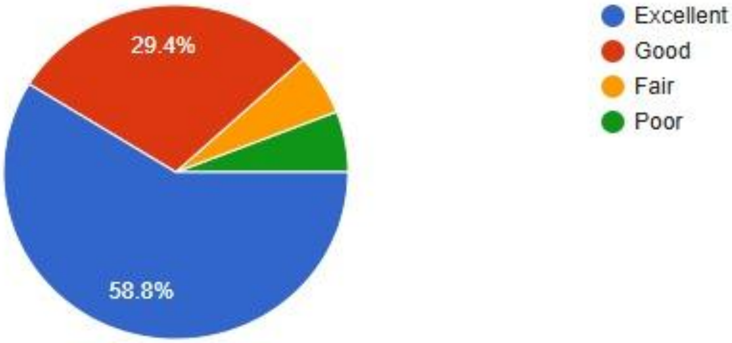
Did the system perform reliably with minimal downtime during your usage?

17 responses



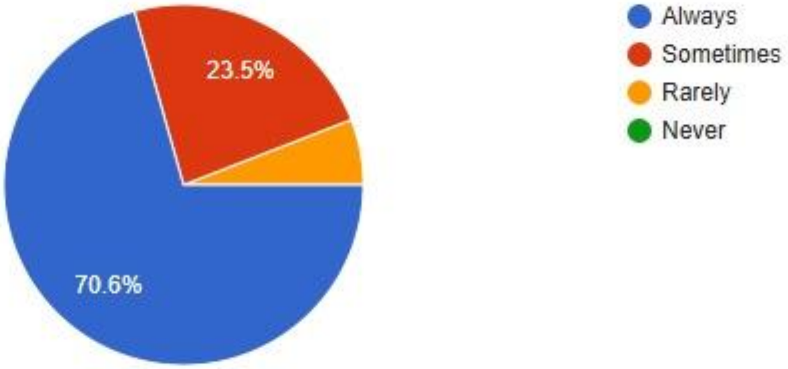
How would you rate the system's ability to handle large-scale traffic effectively?

17 responses



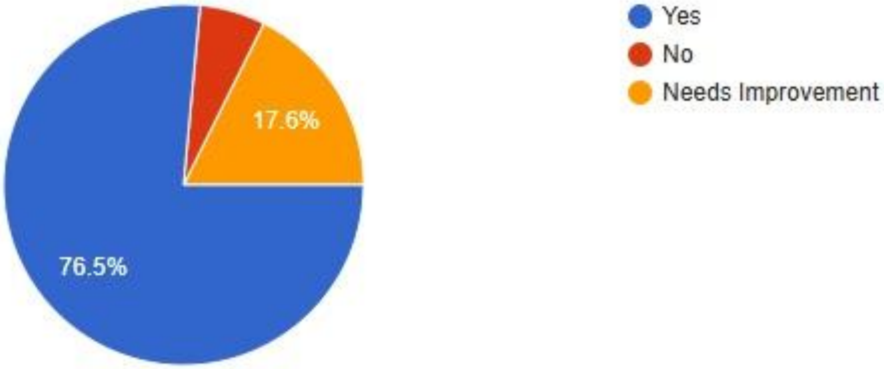
Were notifications for detected anomalies timely and helpful?

17 responses



Did you find the process of exporting anomaly detection results intuitive?

17 responses


● Yes
● No
● Needs Improvement