

iscte

UNIVERSITY
INSTITUTE
OF LISBON

Toward Hybrid Network Security: Machine Learning-Based Anomaly Detection With AI Chatbot-Assisted Mitigation Support

João Rodrigo Pinto Rabuge

Master in Telecommunications & Computer Engineering

Supervisor:

PhD Rui Miguel Neto Marinheiro, Associate Professor,
Iscte – Instituto Universitário de Lisboa

October, 2025

[This page is intentionally left blank.]



TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

**Toward Hybrid Network Security: Machine Learning-Based
Anomaly Detection With AI Chatbot-Assisted Mitigation
Support**

João Rodrigo Pinto Rabuge

Master in Telecommunications & Computer Engineering

Supervisor:

PhD Rui Miguel Neto Marinheiro, Associate Professor,
Iscte – Instituto Universitário de Lisboa

October, 2025

[This page is intentionally left blank.]

The best way to predict the future is to invent it

– *Alan Kay*

[This page is intentionally left blank.]

Acknowledgment

I would like to express my gratitude to my supervisor, Professor Rui Marinheiro, whose expertise, guidance, and support have been really important throughout this research.

I am deeply grateful to my entire family for their constant love and encouragement. A special thanks to my mother Manuela, my father Rodrigo and my brothers Pedro and Francisco. Their belief in my abilities and support have been a source of strength during every challenge.

I extend heartfelt thanks to my friends and colleagues, whose camaraderie and insightful discussions throughout both my bachelor's and master's studies have enriched my academic journey.

I would like to acknowledge Instituto de Telecomunicações and 2024.07624.IACDC/2024 funded by FCT – Fundação para a Ciência e a Tecnologia under PRR – Plano de Recuperação e Resiliência funding, investment RE-C05-i08 - Ciência Mais Digital, DOI: 10.54499/2024.07624.IACDC, for the support for this research

Finally, I wish to express my profound appreciation to all the professors from my Bachelor's and Master's programs. Their dedication to teaching, invaluable insights, and constant encouragement have significantly shaped my academic growth and passion for learning.

This thesis stands as a testament not only to my own efforts but also to the invaluable contributions of all those mentioned above.

[This page is intentionally left blank.]

Resumo

À medida que as infraestruturas de rede enfrentam um número crescente de ciberataques, a detecção inteligente de anomalias e a mitigação rápida tornam-se essenciais para garantir a segurança das redes. Esta dissertação propõe um sistema híbrido de detecção e resposta a intrusões que combina a detecção de anomalias baseada em Machine Learning (ML) com um *chatbot* de Artificial Intelligence (AI), para fornecer orientações interativas de mitigação. Diversos modelos de ML foram avaliados utilizando o conjunto de dados TON_Internet of Things (IoT), que disponibiliza tráfego de rede rotulado como benigno ou maligno [1], [2]. O modelo com melhor desempenho foi integrado numa plataforma de monitorização que captura tráfego (Zeek) e classifica os padrões como normais ou anómalos. Uma interface web (User Interface (UI)) apresenta os resultados da detecção e estatísticas da rede, enquanto o *chatbot* permite a interação em linguagem natural, explicando alertas e sugerindo ações de mitigação.

Durante a seleção do modelo, foram testados e comparados vários algoritmos, incluindo Random Forest (RF), Gradient Boosting (GB), Logistic Regression (LR) e k-Nearest Neighbors (KNN). O modelo RF alcançou o melhor equilíbrio entre precisão de detecção, interpretabilidade e eficiência computacional, sendo, por isso, adotado no sistema final. Ao integrar este modelo com um AI-driven *chatbot*, a solução proposta aborda desafios operacionais críticos nos Security Operations Centers (SOCs), como a sobrecarga de alertas, a interpretabilidade limitada e a lacuna entre detecção e resposta.

PALAVRAS CHAVE: *Mitigação, IDS, Machine Learning, IA, LLM, Chatbot*

[This page is intentionally left blank.]

Abstract

As network infrastructures face an increasing number of cyber attacks, intelligent anomaly detection and rapid mitigation are essential to ensure network security. This dissertation proposes a hybrid system for network intrusion detection and response that combines ML-based anomaly detection with an AI-powered chatbot for interactive mitigation guidance. Several ML models were evaluated using the TON_IoT dataset, which provides network traffic labeled as benign or malicious [1], [2]. The best-performing model was integrated into a real-time monitoring platform that captures traffic (Zeek) and classifies patterns as normal or anomalous. A web UI displays detection results and network statistics, while the chatbot enables natural language interaction, explaining alerts and suggesting mitigation actions.

During model selection, several algorithms, including RF, GB, LR, and KNN, were tested and compared. The RF model achieved the best balance between detection accuracy, interpretability, and computational efficiency, and was therefore adopted in the final system. By integrating this model with an AI-driven chatbot, the proposed solution addresses key operational challenges in SOCs, such as alert overload, limited interpretability, and the gap between detection and response.

KEYWORDS: *Mitigation, IDS, Machine Learning, IA, LLM, Chatbot*

[This page is intentionally left blank.]

Contents

Acknowledgment	iii
Resumo	v
Abstract	vii
List of Figures	xi
List of Tables	xiii
List of Acronyms	xv
Chapter 1. Introduction	1
Chapter 2. State of the Art	5
2.1. Introduction	5
2.2. Research Methodology	5
2.3. Network Anomaly Detection	5
2.4. Datasets for Intrusion Detection	7
2.5. Machine Learning Approaches for Intrusion Detection	8
2.6. AI Chatbots in Network Security	13
Chapter 3. System Architecture	17
3.1. High-Level Architecture	17
3.2. Architectural Planes	18
Chapter 4. Implementation	23
4.1. Introduction	23
4.2. Data Processing and Feature Engineering	23
4.3. Real-Time Pipeline and Module Integration	25
4.4. Chatbot Integration	27
4.5. Web Dashboard and Persistence	28
4.6. Integration, Testing, and Implementation Challenges	29
4.7. Security, Privacy, and Scalability	30
4.8. Artificial Intelligence in Code Generation, Writing, and Bibliography	30
4.9. Summary	31
Chapter 5. Results and Discussion	33
5.1. Effectiveness of a ML IDS (Model Comparison)	34

5.2. Role of the AI Chatbot in Security Operations (Offline Scenarios)	36
5.3. Real-Time Monitoring and System Responsiveness	41
Chapter 6. Conclusion and Future Work	45
Conclusion	45
Future Work	46
References	49
Appendix A. Additional Figures and Article	55
Example of CSV Context	55
Dashboard Components	55
First Page Of Article To Be Submitted	55

List of Figures

Figure 3.1	Three-plane architecture of the proposed system.	18
Figure 3.2	Component diagram of the system. Each plane groups its internal submodules and shows the data flow between them.	19
Figure 3.3	High-level system architecture and user-interface navigation.	20
Figure 4.1	Data preprocessing and feature-engineering pipeline from [23].	24
Figure 4.2	Implementation workflow.	26
Figure 5.1	Dashboard – Realtime monitoring control.	41
Figure 5.2	Dashboard – Monitor Dashboard.	42
Figure 5.3	Dashboard – Real-time statistics.	42
Figure A.1	Excerpt of the CSV context produced by the inference engine.	55
Figure A.2	Dashboard – Chatbot pane.	56
Figure A.3	Dashboard – Model Evaluation.	56
Figure A.4	First page of the article, including title, authors, abstract and keywords.	57

[This page is intentionally left blank.]

List of Tables

Table 2.1	Comparison of benchmark datasets for Intrusion Detection research.	8
Table 2.2	Summary of supervised ML algorithms applied to intrusion detection.	11
Table 2.3	Comparison of major ML implementation frameworks by scope, hardware efficiency, and typical applications. Adapted from [27], [28].	12
Table 5.1	Binary classification on TON_IoT (benign vs. malicious) with training time.	34
Table 5.2	Confusion matrix — Random Forest (binary).	34
Table 5.3	Confusion matrix — Gradient Boosting (binary).	34
Table 5.4	Confusion matrix — Logistic Regression (binary).	35
Table 5.5	Confusion matrix — k -Nearest Neighbours (binary).	35
Table 5.6	Multiclass classification on TON_IoT (attack type) with training time.	35
Table 5.7	Confusion matrix — Random Forest (multiclass).	35
Table 5.8	Confusion matrix — Gradient Boosting (multiclass).	36
Table 5.9	Confusion matrix — Logistic Regression (multiclass).	36
Table 5.10	Confusion matrix — k -Nearest Neighbours (multiclass).	36
Table 5.11	Traffic distribution by protocol in the offline DDoS PCAP.	37
Table 5.12	Top targeted IP addresses during the DDoS attack (combined TCP/UDP packet counts).	37
Table 5.13	Classification results from IDS inference during offline DDoS detection.	37
Table 5.14	Traffic distribution by protocol in the offline backdoor PCAP.	38
Table 5.15	Top destination IP addresses during the backdoor attack.	38
Table 5.16	Classification results from IDS inference during offline backdoor detection.	39
Table 5.17	Verification of chatbot factual correctness for the DDoS scenario.	40
Table 5.18	Verification of chatbot factual correctness for the backdoor scenario.	40

[This page is intentionally left blank.]

List of Acronyms

AI: Artificial Intelligence

API: Application Programming Interface

CNN: Convolutional Neural Networks

CPU: Central Processing Unit

CSV: Coma Separated Values

DDoS: Distributed Denial of Service

DL: Deep Learning

DoS: Denial of Service

GB: Gradient Boosting

GPU: Graphics Processing Unit

GPT: Generative Pre-trained Transformer

HTML: Hypertext Markup Language

IDS: Intrusion Detection System

IIoT: Industrial Internet of Things

IoT: Internet of Things

IP: Internet Protocol

KNN: k-Nearest Neighbors

LLM: Large Language Model

LR: Logistic Regression

LSTM: Long Short-Term Memory

ML: Machine Learning

MITM: Man In The Middle

NLP: Natural Language Processing

RAG: Retrieval Augmented Generation

RF: Random Forest

RL: Reinforcement Learning

REST: Representational State Transfer

RNN: Recurrent Neural Networks

SOC: Security Operations Center

TCP: Transmission Control Protocol

TPU: Tensor Processing Unit

UI: User Interface

XAI: eXplainable AI

CHAPTER 1

Introduction

Modern computer networks are increasingly exposed to cyber threats. While proactive strategies, such as those reviewed by Polónio et al. [3], aim to reduce exposure through early vulnerability mitigation, reactive mechanisms remain essential to handle unknown or fast-evolving attacks. Intrusion Detection System (IDS) is an important defense tool that can help monitor network traffic and identify network attacks. Some older **IDS** are an important reactive defense tool that can help monitor network traffic and identify network attacks. Some older IDS approaches were based on signatures [4], [5], [6] or static rules, which struggle to detect evolving and more sophisticated attacks, creating a need for more intelligent and adaptive anomaly detection techniques.

Machine Learning (ML) offers a better approach by learning patterns of normal and malicious behavior from the network, which leads to the detection of anomalies that do not match known signatures. Some ML algorithms have been applied to network security, from classical Machine Learning methods to Deep Learning (DL). The latter show improved detection rates compared to classical ML algorithms. In contrast, the ML and DL models with better detection capability generally behave like black boxes and their alerts can be of no use to analysts due to shortage of context or explanation [7], [8].

The latest studies have pointed out various potential directions for the future of ML-based detection systems, including enhanced scalability, adversarial resilience, interpretability and adaptation to emerging technologies [9], [10]. Combination of weak learners through ensemble methods has led to Random Forest (RF) and Gradient Boosting (GB) delivering consistent performance, while deep learning frameworks such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks have demonstrated strong capability in capturing both spatial and temporal patterns of *malicious network traffic*, enabling more accurate identification of intrusion events. Moreover, hybrid systems combining supervised and unsupervised learning are gradually being recognized as good methods for bringing out attacks that have not been seen before [6], [11]. The mentioned advancements in the field of intrusion detection through ML still come with obstacles like the need of model interpretability, high computational cost and the inability to provide user with actionable insights in real time.

During the same time frame, there has been also disruptive advances in Artificial Intelligence (AI) and Natural Language Processing (NLP), most notably with the rise of Large Language Model (LLM) like OpenAI's Chat Generative Pre-trained Transformer (GPT), Google's Gemini, and Anthropic's Claude. This has made a significant impact on interaction systems by allowing machines to grasp the situation, reason with facts, and produce

text that is in line with the conversation. Thus, the cybersecurity risks that may arise from these models are gradually being looked into. For example, through the use of the models, analysts can be assisted in providing explanations for alerts and summarising logs, and incident response workflows can be partially automated. These developments indicate that conversational AI is capable of eliminating the difference between very complicated technical systems and human operators, thus making it easier for non-expert users to understand and be aware of the situation [12].

The driving force of this research is the increasing demand for user-friendly anomaly detection systems. An AI chatbot incorporation with an IDS, incident detection system, is a possibility that leads to better user incident response. The chatbot could easily translate difficult to understand attack alerts and statistics into understandable insights to the user and suggest mitigation steps in real time. This addresses a key problem in modern Security Operations Center (SOC) environments, bridging the gap between complex detection data and executable mitigation steps [12], [13].

Building upon the advancements and limitations identified in recent studies, this work follows a research line that explores the convergence between intelligent intrusion detection and human-AI interaction. Existing research has shown that while ML techniques can significantly improve anomaly detection accuracy, their lack of interpretability and actionable feedback still limits practical adoption in operational environments [7], [9]. At the same time, advances in conversational AI, particularly in the field of LLMs, have demonstrated the ability to translate complex technical data into human-readable explanations and decision support.

In this context, the present research aims to unite these two worlds, data-driven anomaly detection and AI-assisted interpretation, to create a system that not only detects malicious network behaviour but also explains and contextualises it for the user. This integration addresses a critical need in modern security operations, where analysts must act quickly and understand the rationale behind alerts rather than relying solely on opaque model outputs.

Therefore, this thesis investigates the design, architecture, implementation, and results of a network security system that combines ML-based anomaly detection with an interactive AI chatbot for mitigation guidance. The primary objective of this research is to develop an intelligent and modular intrusion detection platform capable of not only detecting anomalies but also explaining them and guiding users toward effective mitigation actions in real time. By integrating detection, explanation, and response within a single environment, this work aims to bridge the gap between technical intrusion detection mechanisms and human-centred decision support, ultimately enhancing the usability and effectiveness of modern network defence systems.

This work is guided by the following research questions:

RQ1: How effective is a ML IDS?

To what extent can supervised ML algorithms improve detection performance, accuracy,

precision, recall, F1-score, when applied to modern traffic datasets? Which algorithms have the best performance?

RQ2: What is the role of an AI-driven chatbot in improving security operations?

Can an AI chatbot enhance the interpretability and usability of IDS results by providing contextual explanations and mitigation suggestions in natural language?

RQ3: How can anomaly detection be integrated into real-time network monitoring?

What architectural and implementation strategies enable the deployment of a real-time anomaly detection system capable of capturing, preprocessing, and classifying network traffic continuously?

The answers to these research questions are explored throughout the thesis and discussed in detail in Chapter 5. Each question is addressed through the system’s design, implementation, and evaluation process, allowing the results to directly validate the research questions.

These questions are addressed by building a prototype system that learns to detect intrusions using data from a labeled dataset and provides an AI assistant to explain and react to those detections. The research for this dissertation was divided into four main tasks:

- (1) Study and comparison of multiple datasets for network anomaly detection;
- (2) Study and comparison of multiple ML algorithms for network anomaly detection;
- (3) Development of a real-time network monitoring and analysis platform with a web interface;
- (4) Integration of an AI chatbot into the platform to interact with users about network events.

The main contributions of this thesis include the development of an intelligent and modular intrusion detection platform that combines ML-based anomaly detection with AI-assisted mitigation. The principal outcomes are:

- A comparative evaluation of multiple supervised ML models on a modern intrusion detection dataset, where the Random Forest classifier demonstrated the most balanced performance between accuracy, interpretability, and computational efficiency;
- A real-time IDS prototype based on classical supervised ML techniques, capable of performing continuous monitoring and real-time inference on moderate hardware. While the system supports live traffic monitoring, it was not possible to validate attack detection in production environments, since the test networks were attack-free;

- The integration of a conversational AI chatbot to support attack explanation and response, providing human-readable mitigation suggestions based on the most recent analysis context;
- A public open-source implementation to enable reproducibility and further research.¹

While the system supports real-time monitoring, it was not possible to validate attack detection in live environments since the test networks were attack-free.

Finally, the structure of this document is as follows:

- **Chapter 1 – Introduction:** Introduces the motivation, goals, and research context;
- **Chapter 2 – State of the Art:** Presents the background literature and technologies related to ML-based IDS and AI systems;
- **Chapter 3 – System Architecture:** Describes the structure and components of the proposed solution;
- **Chapter 4 – Implementation:** Details the development, model training, and integration processes;
- **Chapter 5 – Results and Discussion:** Presents evaluation metrics and interprets the findings, providing the answers to the defined research questions;
- **Chapter 6 – Conclusion and Future Work:** Summarizes key outcomes and proposes further research directions.
- **Appendix A — Additional Figures.**

¹<https://github.com/joarabuge/Intelligent-Network-Security>

CHAPTER 2

State of the Art

2.1. Introduction

IDS are a very important tool in protecting networks from attacks. IDS have evolved from simple signature-based approaches [4] to systems that are now powered by ML [6] and, more recently, DL [11]. Signature-based IDS depend on predefined static rules or attack signatures, which ensures high accuracy for known attacks but offers limited capacity to detect new or modified threats. On the other hand, the ML-based Intrusion Detection System is capable of recognizing and learning the regular network behavior and marking the deviations as potential attacks. As a result, the method provides better detection accuracy for old attacks or their modified versions, still, it may create a scenario with a higher false-positive rate [5], [6].

2.2. Research Methodology

The process of the bibliographic review started with the manual search in scientific repositories such as **IEEE Xplore**, **ACM Digital Library**, and **SpringerLink**, where targeted queries concerning intrusion detection systems, machine learning in cybersecurity, and explainable AI were used. These sources not only provided the necessary literature for the state-of-the-art analysis but also the groundwork on which the whole thesis was built.

The manual searches were complemented by **ChatGPT (OpenAI GPT-5) Deep Search mode** which was employed to improve the writing style and guarantee the uniformity of the terms.

2.3. Network Anomaly Detection

An anomaly in a computer network is simply the situation when there is any change in the traffic pattern that is not expected and might signal a threat, misuse, or failure. Traditionally, the two major paradigms for detection in an IDS are signature-based and anomaly-based detection. Signature-based systems operate by employing either predetermined rules or the keeping of fingerprints of already known malicious acts [4]. The aforementioned techniques are extremely accurate in identifying attacks that have been previously observed, in that each packet or flow is matched with a database containing the known patterns. Yet their main downside is that they cannot identify new or modified attacks that are devised in such a way as not to conform to the existing signatures, which makes them powerless against zero-day threats [6].

Anomaly-based systems were then developed as a response to these limitations. Rather than relying on specific signatures, these methods build a model of the normal network traffic and then act by treating deviations as potential attacks [5]. This tactic makes it possible to catch unknown or developing assault types that have not been documented previously. On the downside, however, the nature of standard network behavior means that anomaly detection methods may suffer from a very high rate of false positives where normal traffic changes are mistaken for attacks. To overcome this issue, hybrid systems are created that take the best features from both paradigms, using signatures for the confirmation of known threats and anomaly-based techniques to detect new ones, thereby reaching a compromise between precision and robustness [6].

Recently, scientists working on machine learning have improved the detection systems which are based on anomalies. That systems could automatically learn the patterns of both legitimate and malicious traffic through the dataset provided. These techniques are capable of operating differently, based on the learning strategy, according to the availability of labelled data. In supervised learning, the models are trained using the datasets wherein the instances of the traffic are marked as either benign or malicious, thereby allowing the new samples available to be classified directly [5]. This particular method normally attains a high level of precision provided that the labelled data is not only sufficient but also balanced. On the other hand, unsupervised learning methods like clustering, density-based detection, and autoencoders do not depend on past labels at all [6]. They spot the anomalies by identifying the differences between the current data and its distribution, thus, making it easier to find new attack behaviors. Nevertheless, these methods are usually more susceptible to false alarms because there are no clear reference labels to fall back on.

One technique or method that essentially has the properties of both the aforementioned methods is semi-supervised learning which is considered to be a hybrid strategy that works with a small amount of labelled data along with a large pool of unlabelled samples [6]. This method can be seen as particularly applicable for the network intrusion detection, as getting high-quality labelled datasets is still very expensive and time-consuming. Semi-supervised methods can keep the good detection performance while reducing their dependence on manual annotation, however, they need to deal with the potential bias that inaccurate pseudo-labelling can result in.

Overall, the evolution from rigid signature-based systems to data-driven anomaly detection has significantly improved the adaptability of IDS. ML, through supervised, unsupervised, and semi-supervised approaches, provides the foundation for building systems that can learn from data and adapt to continuously changing network environments [10], [14].

In addition to supervised, unsupervised, and semi-supervised paradigms, **Reinforcement Learning (RL)** has been explored as an alternative learning strategy for intrusion detection systems. In RL-based IDS, the detection process is modelled as an agent interacting

with a network environment, where actions such as classifying traffic or triggering mitigation responses are evaluated through reward signals. This paradigm allows the system to adapt its detection policy over time by maximising cumulative rewards rather than relying explicitly on large labelled datasets. Recent work has demonstrated the feasibility of this approach using Deep Reinforcement Learning techniques, such as Deep Q-Learning, to learn adaptive intrusion detection strategies that balance detection accuracy and false-alarm rates in dynamic network environments [15]. Despite its potential, RL-based IDS often face challenges related to training stability, environment modelling, and higher computational complexity, which currently limit their adoption in lightweight, real-time deployment scenarios.

2.4. Datasets for Intrusion Detection

Benchmark datasets are essential for developing, training, and evaluating IDS, as they provide a controlled environment in which algorithms can be compared under reproducible conditions. However, the quality, diversity, and realism of a dataset largely determine whether a trained model can generalize effectively to real network environments [6].

Over the past two decades, numerous public datasets have been proposed, each addressing specific limitations of its predecessors. Among the most widely used are KDD Cup 99, NSL-KDD, UNSW-NB15, CICIDS2017, Bot-IoT, and, more recently, TON_IoT. Table 2.1 provides a comparative overview of these datasets, summarizing their characteristics, advantages, and limitations.

The earliest benchmark, **KDD Cup 99 (1999)**¹, was derived from the DARPA’98 evaluation project. It includes 41 features extracted from Transmission Control Protocol (TCP) connections across four attack categories (Denial of Service (DoS), Probe, R2L, and U2R). Despite its popularity and historical importance, it contains redundant records that can bias model training and inflate accuracy scores [16]. Its successor, **NSL-KDD (2009)**², mitigated redundancy by removing duplicate records and balancing the distribution of samples. While this improved evaluation consistency, it still reflects outdated traffic and attack types that no longer represent modern network realities [6], [16].

The **UNSW-NB15 (2015)** dataset³ introduced more realistic attack scenarios and modern network protocols, including nine types of attacks such as Fuzzers, Backdoors, DoS, and Worms. It uses 49 features extracted with Argus and Bro (now Zeek), achieving a better balance between realism and diversity, although class imbalance persists [17]. Building on this, **CICIDS2017 (2018)**⁴ provides day-wise labelled traffic captures of normal and multiple attack scenarios (e.g., Distributed Denial of Service (DDoS), brute force, infiltration), which makes it a widely used dataset for enterprise networks. However, it lacks Internet of Things (IoT)/Industrial Internet of Things (IIoT) representation [18].

¹<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

²<https://www.unb.ca/cic/datasets/nsl.html>

³<https://research.unsw.edu.au/projects/unsw-nb15-dataset>

⁴<https://www.unb.ca/cic/datasets/ids-2017.html>

To support IoT-focused intrusion detection, the **Bot-IoT (2020)**⁵ dataset was created at UNSW Canberra. It contains realistic IoT traffic, covering DoS, DDoS, reconnaissance, and data exfiltration activities. Its main strengths are high-quality labelling and realistic botnet simulations, although its narrow focus on volumetric botnet attacks can bias ML models toward specific types of anomalies [19]. The most comprehensive modern dataset is **TON_IoT (2020)**⁶, which integrates telemetry from IoT/IIoT devices, operating system logs, and network traffic data. It provides both binary and multi-class attack labels across a wide range of scenarios, captured in real IoT network environments. Despite its realism and multi-source coverage, it is highly imbalanced, with over 96% of records corresponding to attack traffic [1], [2], [20].

TABLE 2.1. Comparison of benchmark datasets for Intrusion Detection research.

Dataset	Year	Traffic Type	Advantages	Limitations
KDD Cup 99	1999	Synthetic	Simple structure, easy baseline	Outdated attacks, redundancy
NSL-KDD	2009	Synthetic	Balanced and reproducible splits	Still outdated
UNSW-NB15	2015	Hybrid	Realistic protocols, diverse attacks	Moderate class imbalance
CICIDS2017	2018	Hybrid	Detailed labelling, modern traffic	Focused on enterprise networks
Bot-IoT	2020	IoT-focused	High-quality labels, realistic IoT setup	Narrow scope (botnet bias)
TON_IoT	2020	IoT/IIoT + Cloud	Multi-source realism, rich attack variety	Severe class imbalance

From Table 2.1, based on [20] it is evident that dataset development has progressed from synthetic and limited benchmarks to realistic, heterogeneous sources. Older datasets such as KDD Cup 99 and NSL-KDD remain useful for baseline evaluation, but modern research increasingly favours datasets like UNSW-NB15, CICIDS2017, and especially TON_IoT, due to their realism, multi-source structure, and applicability to IoT-based environments.

The **TON_IoT dataset** was therefore chosen for this research as it provides one of the most comprehensive and up-to-date representations of real network behaviour. Its selection is justified by several factors: heterogeneous data sources (IoT, IIoT, and edge devices) [1], [20], coverage of multiple attack types such as DDoS, Backdoor, Scanning, and Man In The Middle (MITM), realistic traffic captured from operational testbeds [2], and supervised labels enabling both binary and multi-class classification.

However, strong class imbalance is a well-documented limitation of modern intrusion detection datasets, including TON_IoT: certain attack types appear orders of magnitude more frequently than others, and benign traffic can dominate entire captures [1], [2], [9]. This issue is *routinely* handled in the literature using techniques such as resampling of minority classes, feature-selection and weighting strategies to favour under-represented classes, and the use of weighted evaluation metrics rather than raw accuracy [9], [21].

2.5. Machine Learning Approaches for Intrusion Detection

ML algorithms remain competitive in intrusion detection due to their proven efficacy on tabular network-traffic features, relatively modest computational demands, and superior

⁵<https://research.unsw.edu.au/projects/bot-iot-dataset>

⁶<https://research.unsw.edu.au/projects/toniot-datasets>

interpretability compared with many DL architectures [5], [6], [22]. While DL models such as CNN or Autoencoders can achieve high accuracy on large-scale data, they typically require greater computational resources and are harder to interpret. In contrast, classical ML models provide faster training and lower latency.

The selection of algorithms in this work was guided by: (1) empirical evidence of performance and reliability in IDS tasks, including prior studies using TON_IoT (2) complementary profiles across accuracy, interpretability, and computational cost; (3) suitability for real-time or near-real-time operation in resource-constrained environments.

Accordingly, four algorithms were selected: RF, GB, Logistic Regression (LR), and k-Nearest Neighbors (KNN). These methods have been widely documented in the IDS context: RF and GB evaluated on TON_IoT in [21], LR as a transparent baseline in [23], and KNN in [23].

Random Forest: RF is an ensemble of decision trees trained on bootstrapped subsets of the data. At each split, a random subset of features is considered, which reduces correlation between trees. Node impurity is commonly measured by the Gini index:

$$G = \sum_{i=1}^C p_i(1 - p_i),$$

where p_i is the proportion of class i in the node. Predictions are then aggregated through majority voting:

$$H(x) = \text{mode}\{h_1(x), h_2(x), \dots, h_T(x)\}.$$

RF is often considered robust to noise because averaging multiple independent trees can reduce the impact of outliers and random fluctuations. It tends to achieve strong accuracy on tabular data, as different trees may capture different feature interactions, improving generalisation across heterogeneous network traffic. Moreover, RF can provide feature-importance metrics that highlight which attributes contribute most to classification decisions, an important aspect for IDS interpretability [21]. A key limitation is reduced transparency compared to a single decision tree: while each tree in the forest is individually interpretable, the final prediction results from aggregating many diverse decision paths, making it difficult to trace a single, human-readable explanation for a specific classification.

Gradient Boosting: GB is an ensemble technique that builds several simple decision trees sequentially. Each new tree focuses on correcting the mistakes made by the previous ones, giving more weight to samples that were misclassified. The final model is obtained by combining all trees:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x),$$

where γ_m is the learning rate. This iterative process gradually minimises the loss function and delivers strong predictive performance on structured data such as network traffic flows [24]. However, if too many trees are added, if they are too deep, or if the learning

rate is too high, the model can start fitting random noise in the training data instead of learning general patterns (i.e. *overfitting*). Regularisation techniques, such as smaller learning rates, limiting tree depth, random subsampling and early stopping once validation performance stabilises, are often used to mitigate this risk.

Logistic Regression: LR is a statistical model that estimates the probability of class membership by assuming a linear relationship between features and the log-odds:

$$\log \frac{P(y = 1 | x)}{1 - P(y = 1 | x)} = \beta_0 + \sum_{i=1}^n \beta_i x_i \quad \Rightarrow \quad P(y = 1 | x) = \frac{1}{1 + \exp[-(\beta_0 + \sum_{i=1}^n \beta_i x_i)]}.$$

Because its convex optimisation guarantees convergence to a global optimum, training is generally stable and computationally efficient. Each coefficient β_i can provide insights into both the direction and strength of the influence of its corresponding feature, which supports explainable analysis of IDS alerts in many studies [23]. L1 (Lasso) and L2 (Ridge) regularisation can constrain the model’s coefficients to improve generalisation and numerical stability, particularly in high-dimensional traffic data. Although its linear decision boundary limits LR’s ability to model complex non-linear attack patterns, it remains a transparent and reliable baseline and is commonly used as a component in IDS ensemble frameworks.

k-Nearest Neighbours: KNN classifies an instance based on the majority label among its k nearest neighbours, typically using Euclidean distance:

$$d(x, x_i) = \sqrt{\sum_{j=1}^n (x_j - x_{ij})^2}.$$

Normalisation is essential because distance-based algorithms are sensitive to feature scale: variables with larger numeric ranges can dominate the distance metric and distort neighbour relationships. Min-max or z-score normalisation ensures that all features contribute proportionally. In contrast, tree-based models such as RF and GB are largely scale-invariant, while LR benefits from scaling mainly for numerical stability rather than distance computation. KNN is non-parametric and flexible, adapting naturally to non-linear boundaries, but on large datasets inference may become slower and it is typically more sensitive to irrelevant or highly correlated features [25].

To provide a concise comparative summary of the main supervised algorithms evaluated in this work, Table 2.2 was constructed based on findings from recent IDS research [9], [21], [23], [24], [25], [26]. It summarises each algorithm’s main strengths, limitations, typical IDS applications, and computational considerations.

One of the most used frameworks for implementation of ML algorithms is **scikit-learn** framework, which provides a unified and efficient Application Programming Interface (API) for classical ML models in Python. Scikit-learn includes optimised Central Processing Unit (CPU)-based implementations for algorithms such as RF, GB, LR, and

TABLE 2.2. Summary of supervised ML algorithms applied to intrusion detection.

Algorithm	Main Advantages	Limitations	IDS Applications / Refs	Comp. Cost
RF	Robust, handles mixed data, interpretable via feature importance	Less transparent; large model size	DDoS, general IDS [21], [24]	Medium
GB	High accuracy; models complex relations	Sensitive to hyperparameters; risk of overfitting	IoT IDS, ensemble models [9], [24]	Med–High
LR	Simple, fast, interpretable coefficients	Linear boundaries only	Baseline, meta-classifier [23], [26]	Low
KNN	Non-parametric; flexible decision boundaries	Slow inference; scale sensitive	Attack classification [25]	High

KNN, together with integrated preprocessing, feature selection, and cross-validation utilities. Its simplicity, stability, and deterministic execution make it well-suited for research focused on reproducibility, interpretability, and deployment in resource-constrained environments rather than large-scale deployments using Graphics Processing Unit (GPU) acceleration.

Alternative frameworks such as **TensorFlow** and **PyTorch** are designed for more complex DL architectures, offering automatic differentiation, GPU parallelism, and scalable training pipelines. These frameworks are ideal for deep neural networks and large-scale datasets but come at the cost of increased implementation complexity, higher hardware requirements, and reduced transparency for classical ML workflows.

A concise comparison of major ML frameworks is presented in Table 2.3, summarising their focus, hardware efficiency, and common use cases in IDS and general ML research. The information is synthesised from comparative studies in [27], [28] and official framework documentation.

Overall, scikit-learn was selected as the main implementation framework in this work because it provides an ideal balance of computational efficiency, simplicity, and interpretability. Its modular structure ensures compatibility with future DL extensions using TensorFlow or PyTorch, maintaining consistency with the modular architecture of the proposed IDS system, as can be checked from its widespread use in the scientific community in tables 1 and 2 in [28].

TABLE 2.3. Comparison of major ML implementation frameworks by scope, hardware efficiency, and typical applications. Adapted from [27], [28].

Framework	Primary Focus	Hardware Efficiency	Typical Use Cases
Scikit-learn	Classical ML (trees, regression, clustering)	CPU-optimised; low memory use	Tabular data, academic research, IDS prototyping
TensorFlow	Deep Learning (neural networks, autoencoders)	GPU/TPU acceleration	Large-scale DL models, image or time-series IDS
PyTorch	Deep Learning, research flexibility	GPU/CPU hybrid	Custom DL research and experimentation

Note: Adapted from performance and dependency analyses of open-source ML frameworks in [27], [28] and official framework documentation.

In parallel with classical ML, recent research has increasingly applied DL architectures such as CNN, Recurrent Neural Networks (RNN), LSTM and Autoencoders to intrusion detection [10], [11]. CNNs capture spatial correlations among network features, while RNNs and LSTMs model temporal dependencies in traffic flows. On the other hand, autoencoders carry out unsupervised reconstruction and mark big reconstruction errors as anomalies. Deep learning (DL) ensembles have been reported to achieve detection accuracy above 97–99% on benchmark datasets like UNSW-NB15 and TON_IoT, very often surpassing traditional machine learning (ML) models [6].

Notwithstanding, these improvements come at a price: Deep Learning (DL) models are in need of large human-annotated data, powerful CPUs usually in the form of GPUs/TPUs, and meticulous hyperparameter tuning to keep their performance consistent. Moreover, the very nature of their decision-making processes remains unclear, thus making it hard for the users to know what exactly the "black-box" is doing, which in turn, hampers transparency and complicates alert interpretation [7], [13]. Their limited interpretability and high computational demand make them impractical for real-time deployment in resource-constrained systems. Therefore, this study will consider the classical ML algorithms which will provide higher transparency, lower costs, and easier integration into lightweight, real-time network monitoring environments with negligible detection accuracy compromise.

In the IDS implementation, practical aspects like feature selection, explanation and realtime integration really contribute towards the performance and the operational value of the system.

Feature selection refers to the process of feature dimensionality reduction, which ultimately decreases computational cost and improves model accuracy by selecting only the

most informative attributes for classification. Common approaches include correlation filtering, information gain, and wrapper-based selection techniques. By focusing on the most relevant features, models become faster and more generalisable, a critical advantage when working with large, heterogeneous datasets such as TON_IoT [29], [30].

eXplainable AI (XAI) tools, especially **LIME** (Local Interpretable Model-Agnostic Explanations) [31] and **SHAP** (SHapley Additive exPlanations) [7], [13], enable analysts to identify which input features have the most impact on the model’s predictions. In the context of intrusion detection, these explainability techniques help determine whether a certain classification was caused by atypical packet rates, unusual port access, or abnormal protocol behaviour. Integrating SHAP-based explanations into the system output provides valuable interpretative insights for security analysts, while at the same time allowing these insights to be translated into natural-language summaries that can later support dialogic interaction with the integrated chatbot.

From the operational point of view, effective intrusion-detection systems must incur very low latency to function in live networks. Network monitoring solutions such as **Zeek**⁷, **Suricata**⁸, and **Snort**⁹ collect flow logs and metadata in real time from the network; these can then be parsed and fed into machine learning models for inference in near-real time. Zeek is known for its extensibility and is thus often used in research due to its output of detailed connection logs and flow features suitable for ML pipelines. In contrast, Suricata and Snort remain viable options where packet inspection or signature-matching is required, complementing anomaly-based methods applied in this research [20]. This combination of efficient capture, interpretable predictions, and low-latency data collection underpins the deployment of an intrusion-detection system capable of both accurate detection and meaningful human–AI collaboration.

2.6. AI Chatbots in Network Security

Conversational AI, chatbots, has recently gained traction in cybersecurity through the adoption of LLMs. These systems can interpret security logs, summarise technical alerts, and present information in natural language to assist analysts and reduce cognitive load [12].

Several notable prototypes have emerged, each exploring a different stage of automation and user interaction:

- **SecBot (2020)** [32] – focuses on strategic planning and static threat-intelligence workflows, providing high-level guidance rather than processing live intrusion data.

⁷Zeek – The Network Security Monitor. <https://zeek.org/>.

⁸Suricata – High performance open-source network analysis and threat detection engine. <https://suricata.io/>.

⁹Snort – Open-source network intrusion detection prevention system. <https://www.snort.org/>.

- **HuntGPT (2023)** [33] – integrates a Random Forest-based IDS with SHAP / LIME explanations and an LLM-based conversational interface; however, it operates on pre-recorded alerts rather than streaming network flows.

Recent works have begun exploring how XAI and LLMs can be integrated to justify IDS alerts. In such systems, a user can ask the chatbot why a detection occurred, and the chatbot retrieves the corresponding SHAP or LIME explanations from the underlying ML model.

Recent research has also started to explicitly position LLMs as a mechanism to improve the explainability layer of IDSs. For example, Houssel et al. [34] discuss the use of LLMs to support explainable intrusion detection by transforming model outputs and network-security context into human-readable justifications. This direction has since been reinforced by broader surveys focusing on transformers and LLMs for IDS, highlighting opportunities and open challenges such as grounding, evaluation protocols, robustness, and operational constraints [35]. More recent contributions further extend this line of work by proposing dedicated frameworks that combine LLMs-based reasoning with explainable IDS pipelines [36], and by exploring explainable anomaly detection in network traffic using LLMs-supported approaches in network management contexts [37].

To strengthen factual grounding, the integration of Retrieval Augmented Generation (RAG), a framework that combines an LLM with an external or local knowledge base to retrieve relevant contextual information, has recently been proposed to improve factual accuracy and reduce hallucinations in LLM-based systems [31], [38]. When a question is asked, RAG retrieves the most relevant detection logs or SHAP summaries and injects them into the prompt, allowing the LLM to generate answers that are both context-aware and data-driven. This approach mitigates hallucinations and ensures the chatbot’s explanations are directly supported by real IDS outputs. Nevertheless, privacy and security concerns arise when transmitting such summaries to external APIs. On-premise deployment or anonymisation of traffic logs are common mitigation strategies [7], [12], [13].

Compared with SecBot and HuntGPT, both of which rely on predefined or static data sources, the proposed system performs real-time integration through Zeek-based traffic monitoring. Zeek logs feed lightweight ML classifiers whose predictions are interpreted by an AI-driven chatbot focused on mitigation guidance. Unlike prior systems that aimed to justify detections or visualize SHAP explanations, the chatbot in this work is designed to assist users by suggesting concrete mitigation actions and good network practices based on the detected attack type and the available hardware components. Feature-importance analysis is instead carried out offline in the model evaluation module, where the most influential attributes for binary and multi-class classification are identified.

Overall, this integration of anomaly detection, explainable reasoning, and interactive AI represents a step toward more human-centred security automation. It bridges the

gap between detection and actionable response, enabling users to react immediately while understanding the context of the detected event.

From the reviewed literature, several broader trends and gaps can be identified:

- Ensemble ML methods such as RF and GB consistently outperform single classifiers in detection accuracy and robustness [24].
- DL approaches achieve very high accuracy but require significant computational resources and often lack interpretability [10], [11], [22].
- Feature selection and XAI support analyst-level interpretability, but conversational interfaces dedicated to mitigation remain rare [7], [13].
- Real-time IDS integrated with conversational AI are still largely unexplored, revealing a gap in lightweight, human-centred security automation [6].

Building upon these insights, the following chapter explores the system architecture and modular design that operationalise these opportunities, namely, the real-time integration of traffic monitoring, explainable model inference, and AI-assisted mitigation. This architecture serves as the practical foundation of the proposed approach and is detailed next.

[This page is intentionally left blank.]

System Architecture

This chapter presents the architecture of the intelligent network security system developed in this work. The platform was designed to detect, classify, and explain network anomalies in real time through a modular structure that separates data capture, machine-learning inference, and user interaction. The architecture is organised into three logical layers, or **planes**: the **Data Plane**, the **Analytics Plane**, and the **Interaction Plane**. Each plane fulfils a distinct role within the system but integrates seamlessly into a unified workflow.

3.1. High-Level Architecture

The architecture of the system follows a layered approach, which offers advantages in terms of scalability, modularity, and separation of concerns. On a high level, it organises the main functionalities into three coordinated planes, the **Data Plane**, **Analytics Plane**, and **Interaction Plane**, that cooperate to capture, process, and present network information in real time. The interplay among these planes is illustrated in Figure 3.1.

The depiction in Figure 3.2 offers a much more detailed component-level perspective, with the primary modules functioning in each plane and the data movement between them illustrated.

To support this perspective, Figure 3.3 not only provides a visualization of the user-interface navigation but also illustrates how the system’s main functionalities are accessed by the user. Starting from the login or registration stage, the user is redirected to the **Home** page, which serves as the central access point to the dashboard, reports, and conversational assistant. From the dashboard, different analytical modules can be reached, including **real-time monitoring**, **PCAP trace analysis**, and **model evaluation**, each of which links to more detailed result and statistics pages. This hierarchical structure ensures that all operations—from model execution to chatbot interaction—are integrated within a unified workflow, consistent with the separation of planes previously shown in Figure 3.1. By combining functional and navigational views, the figure highlights how both automation and human supervision coexist within the same interface, supporting a seamless analytical experience.

Both the technical and humanistic points of view of the system architecture are offered by these figures. The backend carries out real-time data gathering and drawing of conclusions, and the frontend combines and displays the outcomes to help the operator get a clear picture of the network activity.

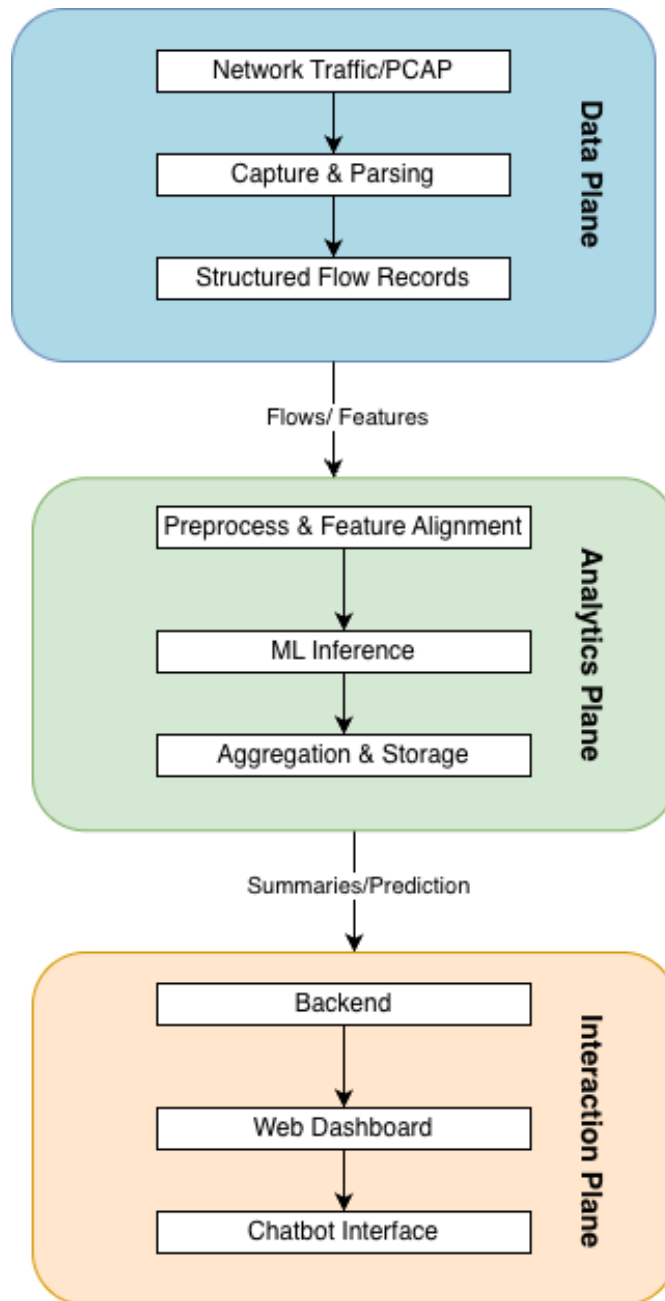


FIGURE 3.1. Three-plane architecture of the proposed system.

3.2. Architectural Planes

Each architectural plane represents a functional layer within the system, responsible for a specific stage of the data-to-decision pipeline. Together, these planes define a continuous workflow that begins with network traffic acquisition, progresses through machine-learning-based analysis, and culminates in user interaction and response. This layered organisation improves scalability, modularity, and clarity, allowing each component to evolve independently while maintaining seamless integration across the full intrusion detection process.

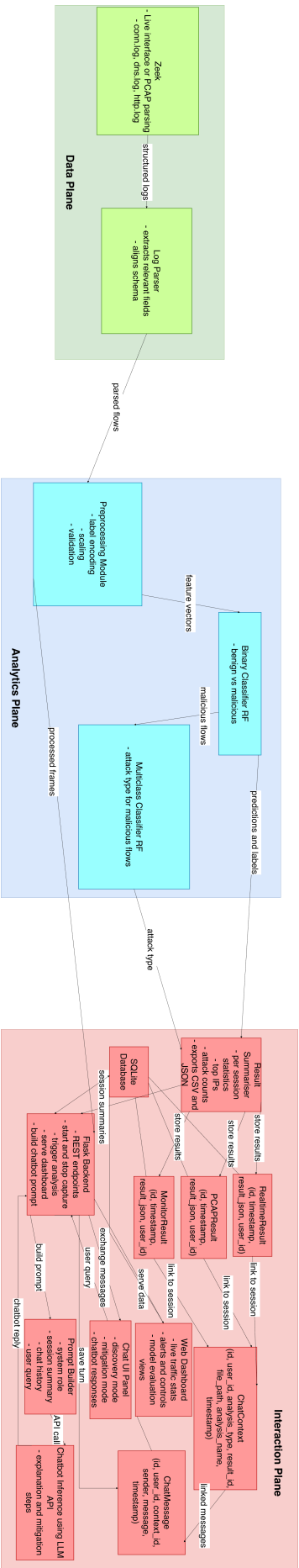


FIGURE 3.2. Component diagram of the system. Each plane groups its internal submodules and shows the data flow between them.

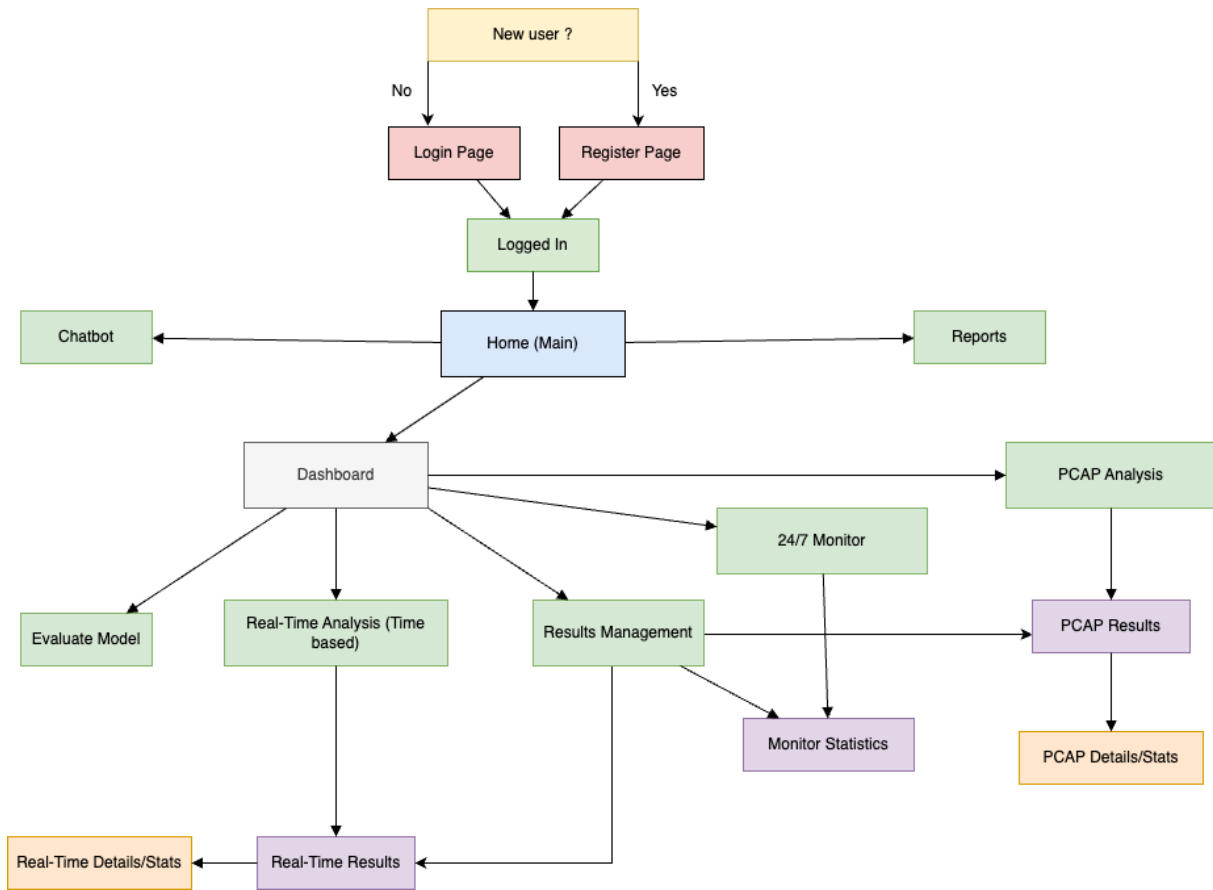


FIGURE 3.3. High-level system architecture and user-interface navigation.

The **Data Plane** is responsible for acquiring network traffic and converting it into structured flow records suitable for analysis. It supports multiple operational modes, including offline analysis of captured traces, real-time capture in short windows, and continuous 24/7 monitoring. Captured information is cleaned and formatted, producing standardised records that are fed into the analytical layer.

The **Analytics Plane** is able to convert the structured data from networks into actionable security insights. It detects anomalies and classifies potential attack types by applying preprocessing, feature alignment, and machine-learning inference. The results are then condensed into high-level summaries such as traffic statistics, rankings of sources and destinations, and counts of alerts, which are then stored locally for the purpose of subsequent visualization.

The inference process within this layer is a two-stage pipeline. The model, in the first stage, executes **binary classification** determining whether the traffic is benign or malicious. If a sample is deemed to be malicious, then the second stage is activated and **the specific attack type is classified** using a multi-class model. This design choice was made to minimize the usage of computational resources and to enhance the understanding of the model's outputs as the initial binary filter eliminates the need for performing multi-class inference on traffic flows that are benign. Furthermore, it is similar to the actual

SOC practice in which rapid anomaly detection is followed by detailed context-based analysis. The predictions are aggregated, and local storage is used to keep them, which is the analytical foundation for both the dashboard and chatbot modules.

The **Interaction Plane** is the main point of connection between the analytical core of the system and the end-user, and it has two main types of interfaces: a visual dashboard that shows results in real-time, and a conversational assistant that gives explanations and guidance in context. Both interfaces use the same information that has been processed, and thus there is no difference between the visual and the textual outputs. To ensure that no data from one user is accessible to another, session isolation is enforced and future deployments may even include layers of anonymisation to protect sensitive identifiers.

The conversational assistant functions in two different ways depending on whether device–IP mappings are available or not. In *discovery mode*, the chatbot summarises the global anomalies and asks the user to identify which IP addresses are connected to specific devices on the network when such mappings are still not defined. As soon as the user shares one or more mappings, the system moves to *mitigation mode* where the assistant is able to prepare a numbered list of remediation steps that are specific to the type of attack that has been identified and the device (or devices) that are affected. This dual-mode logic guarantees that the assistant is always in tune with the user’s situation and stage of response, facilitating both the exploratory understanding and the guided remediation through one interface.

To sum up, this chapter presented the configuration of the intelligent network security system, which was divided into three synchronized planes: **Data**, **Analytics**, and **Interaction**. It additionally illustrated the main component connections and user-interface composition, emphasizing how the three processes of data collection, analysis, and human interpretation are integrated into one system. The following chapter examines the real-world application of this architecture by giving an account of the technologies, datasets, and algorithms that were used to implement and verify the proposed design.

[This page is intentionally left blank.]

CHAPTER 4

Implementation

4.1. Introduction

This chapter details the practical implementation of the intrusion detection system and the integrated chatbot application. It focuses on the technical architecture, development environment, and module integration of the prototype. Each component of the system is described, including data processing, model training, real-time traffic monitoring and the chatbot interface, to explain how the system was built. Evaluation outcomes and performance metrics are not discussed here and will be presented in Chapter 5. All development, training, and testing were conducted on a 2024 MacBook Air M3 with an 8-core CPU and 8 GB of unified memory. The absence of a high-performance GPU motivated a focus on supervised ML algorithms for efficiency in training and inference.

The software stack used for implementation comprised:

- Python 3.11 for all data processing and backend code;
- scikit-learn (v1.6.1) for ML models such as Random Forest, Gradient Boosting, Logistic Regression, and KNN;
- pandas (v2.2.3), numpy (v2.2.4), and matplotlib (v3.10.1) for data handling and visualization;
- Flask (v3.1.0) for the web application backend and REST API, with Bootstrap for the web UI;
- SQLite (via SQLAlchemy v2.0.36) for lightweight data persistence;
- Zeek (v7.1.0) for network traffic monitoring and log generation;
- OpenAI API (v0.28.0) integration for the chatbot’s language model responses.

4.2. Data Processing and Feature Engineering

The preprocessing workflow of the **TON_IoT dataset** followed a structured multi-stage pipeline inspired by *Guo et al. (2023)* [23], which applied a standardized feature-engineering process for intrusion detection using the same dataset, as illustrated in Figure 4.1. The stages included **(1) data cleansing**, **(2) feature encoding**, **(3) normalization**, **(4) feature selection**, and **(5) data splitting**, ensuring consistency between offline model training and live inference.

The first step of the pipeline involved **data cleansing and handling missing values**. Columns that did not generalize across environments (such as `src_ip` and `dst_ip`) and features with constant values were removed. A Spearman correlation analysis was

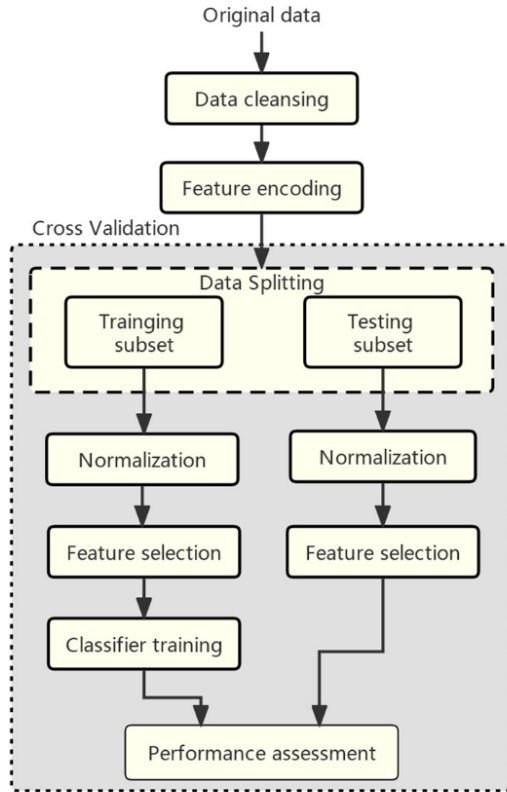


FIGURE 4.1. Data preprocessing and feature-engineering pipeline from [23].

applied to identify highly correlated numeric features (> 0.9), and redundant attributes were discarded to prevent multicollinearity.

To maintain a consistent schema between offline and live data, missing or inconsistent values were systematically treated. Non-numeric entries in numeric fields were converted to zero, while categorical missing values were replaced with a placeholder (“-”). This ensured that live traffic logs preserved the same structure as the training data, avoiding schema mismatches during inference.

The next stage, **feature encoding and normalisation**, converted categorical variables into numeric form using scikit-learn’s `LabelEncoder`, which assigns each category a unique integer code. This approach was chosen intentionally over one-hot encoding to avoid a substantial increase in feature dimensionality, given the large number of categorical variables in the **TON_IoT** dataset. Reducing dimensionality helps keep the feature vector compact, minimises memory usage, and accelerates both training and inference, aspects that are particularly relevant under limited computational resources. Because Random Forests and other tree-based models are insensitive to the numeric ordering of categorical encodings, the use of integer codes does not bias model training.

For numerical features, several scaling strategies were evaluated to ensure that all attributes contributed proportionally to the models: **Min–Max scaling**, which linearly rescales features to a fixed $[0, 1]$ interval while preserving their original distribution shape;

Z-score normalisation, which transforms features to have zero mean and unit variance; and **Robust scaling**, which uses medians and interquartile ranges, making it less sensitive to outliers. After empirical comparison, the **Min–Max scaling** technique was selected for the final pipeline because it offered the most stable and consistent results during real-time inference. This method confines all features to a common numerical range and thus improves numerical stability during continuous monitoring sessions.

The fitted scaler was serialised to disk using `joblib` and reloaded by the live pipeline so that each batch of traffic data would be transformed in an identical way to the training data, guaranteeing reproducibility and consistency between offline and real-time operation.

Finally, **data splitting and feature selection** were performed to prepare the data for model training. The preprocessing pipeline, including encoders, scaler, and feature selection, was serialized to disk using `joblib` to ensure reproducibility. The dataset was divided into 80% training and 20% testing subsets using a fixed random seed. Feature selection was performed implicitly through correlation filtering and variance analysis, improving efficiency without compromising classification accuracy. The resulting preprocessed files were saved for subsequent model training and evaluation.

The final stage in this pipeline, **model selection and training**, evaluated four supervised learning algorithms on the processed **TON_IoT** data: **Random Forest (RF)**, **Gradient Boosting (GB)**, **Logistic Regression (LR)**, and **k-Nearest Neighbors (KNN)**. As introduced in Chapter 3, the deployed system follows a two-stage inference pipeline: first a *binary* classifier distinguishing benign from malicious traffic, and then a *multiclass* classifier applied only to malicious flows to determine the specific attack type. This design reduces computational cost, isolates the minority malicious class, and simplifies the operational workflow. The deployed prototype therefore uses two Random Forest models, trained offline and serialized using `joblib`. Offline comparison results are discussed in Chapter 5.

- **Random Forest (RF)**: `n_estimators=100` with `random_state=42`. Used in both stages and the only model type deployed.
- **Gradient Boosting (GB)**: `random_state=42`, evaluated offline only.
- **Logistic Regression (LR)**: `max_iter=1000`, evaluated offline only.
- **k-Nearest Neighbors (KNN)**: $k = 5$, Euclidean distance, evaluated offline only.

These algorithms served as comparative baselines to validate the robustness, interpretability, and efficiency of the Random Forest classifier, which ultimately provided the best balance between accuracy and computational performance.

4.3. Real-Time Pipeline and Module Integration

After training the models, the system was implemented to perform real-time network monitoring and anomaly detection using the trained classifiers. The components communicate

through intermediate Coma Separated Values (CSV) files handled by the Flask backend. Figure 4.2 illustrates this workflow from packet capture to user-facing components such as the dashboard and chatbot.

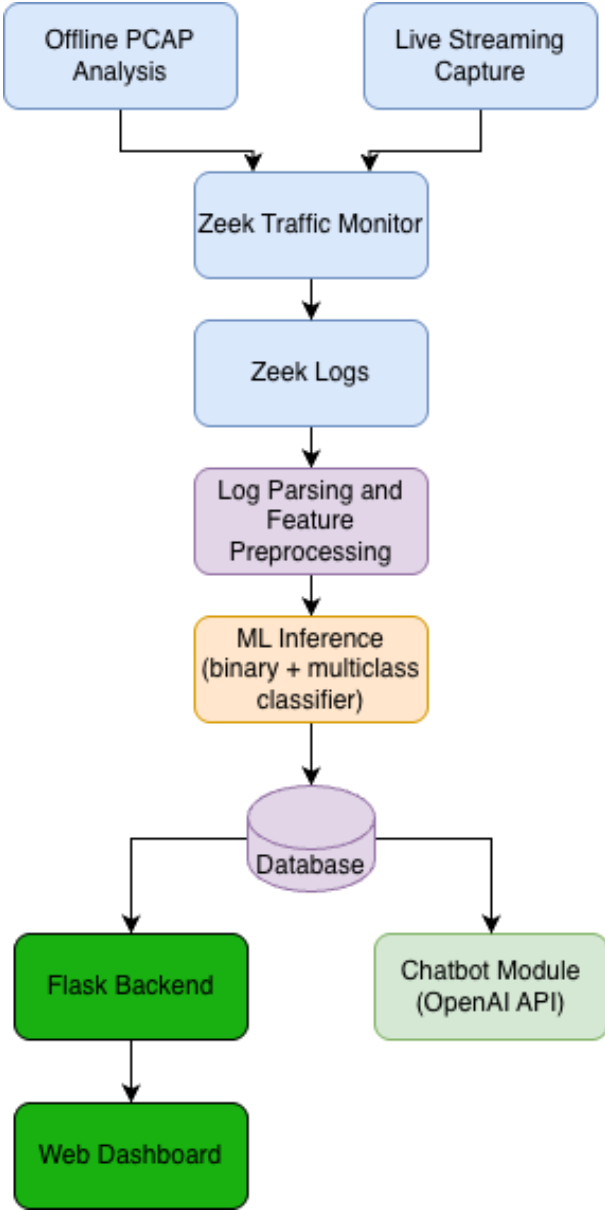


FIGURE 4.2. Implementation workflow.

Traffic Capture and Log Generation. Network traffic is monitored using Zeek, which can operate in two modes: *live mode*, capturing packets directly from a network interface, and *offline mode*, parsing PCAP files. In both cases, Zeek produces structured logs such as `conn.log`, `dns.log`, `ssl.log`, `http.log`, and `weird.log`. A small auxiliary Zeek script (`real-time/force_all_logs.zeek`) ensures that all protocol logs are consistently generated, even when some protocols are absent in the capture.

Log Parsing and Preprocessing. This stage focuses on transforming Zeek logs into model-ready input. New entries in the logs are continuously parsed and structured in near real-time. All relevant fields (e.g., timestamps, source/destination Internet Protocol

(IP)s and ports, protocol flags) are mapped to the schema expected by the machine-learning models. The serialized encoders and the `MinMaxScaler` from the offline phase are reloaded via `joblib` and applied identically to each batch, guaranteeing that live inference mirrors the training transformations. A lightweight validation step compares column names, order, and data types with a stored schema hash to prevent drift between offline and runtime pipelines.

4.4. Chatbot Integration

At runtime, the system employs two trained RF models in a two-stage inference pipeline. The first model performs binary classification to distinguish benign from malicious flows. For flows classified as malicious, a second model performs multiclass classification to identify the specific type of attack.

The results of this pipeline are exported into structured CSV files, which serve as the main input for the chatbot module. Each CSV includes a summary of the current session, covering metrics such as total flow count, number of anomalies detected, most frequent attack types, and top source/destination addresses.

These CSV summaries are injected into a predefined system prompt, which defines the assistant’s behaviour and instructs it to base its answers solely on the data provided. The chatbot does not have tool access or external retrieval capabilities, ensuring that all responses are grounded in the local context.

Operationally, the backend constructs the message list as follows:

- a single system prompt defining the assistant’s behaviour;
- the CSV-derived summary of the current session;
- an optional replay of the chat history for context continuity;
- and the new query.

This structured message is sent to the OpenAI API, which returns a context-aware response based on the available data. An illustrative example of a CSV summary and chatbot reply is provided in Appendix A.

The chatbot operates in two distinct modes. In *discovery mode*, when no device mappings are provided, it summarises global anomalies and asks the user to identify which IPs correspond to which devices. Once any mapping is provided, it switches to *mitigation mode*, producing a numbered list of remediation steps tailored to the detected attacks and corresponding devices.

On failure (e.g., missing `OPENAI_API_KEY`, timeout or rate-limit), the exception is caught and a visible banner is rendered in the chat pane with the cause and a suggested action (retry with backoff or check credentials).

Finally, while this section focuses on the chatbot’s inference and dialogue logic, the mechanisms responsible for storing its conversational history and session context are described in detail in the following section (*Web Dashboard and Persistence*), where the persistence layer is implemented and discussed.

4.5. Web Dashboard and Persistence

The user interface is implemented as a web dashboard served by the Flask backend, using Hypertext Markup Language (HTML) and JavaScript on the frontend with Bootstrap for styling. It provides a unified control and monitoring platform for the entire intrusion-detection pipeline. The main dashboard views and controls include:

- **Monitoring controls:** Options to start and stop continuous 24/7 capture, run a real-time analysis for a shorter window, or upload a PCAP file for offline evaluation.
- **Real-time results and history:** A live panel continuously updates the counts of benign versus malicious flows and lists recent alerts. Users can also query historical results stored in the local database.
- **Model-evaluation views:** Dedicated pages allow reviewing the trained model's performance through visualisations such as confusion matrices, classification reports, and feature-importance plots, assisting the interpretation of ML outputs.
- **Chatbot pane:** An interactive chat window where the user can query the system about the network status or request mitigation guidance. The chatbot's responses appear in this pane, providing explanations or actionable steps.

Representative screenshots of these dashboard components are presented in Appendix A: A.3, A.2, and in Chapter 5: 5.1 5.2

To guarantee session continuity and data traceability, all processed results and chatbot interactions are stored locally using SQLite. The database schema manages both analysis outputs and conversational contexts, hence supporting multi-session and multi-user operation.

LISTING 4.1. Simplified SQLite schema for result persistence and chat sessions.

```
CREATE TABLE RealtimeResult (  
    id INTEGER PRIMARY KEY,  
    timestamp TEXT,  
    result_json TEXT,  
    user_id INTEGER  
);
```

```
CREATE TABLE PCAPResult (  
    id INTEGER PRIMARY KEY,  
    timestamp TEXT,  
    result_json TEXT,  
    user_id INTEGER  
);
```

```

CREATE TABLE MonitorResult (
    id INTEGER PRIMARY KEY,
    timestamp TEXT,
    result_json TEXT,
    user_id INTEGER
);

```

```

CREATE TABLE ChatContext (
    id INTEGER PRIMARY KEY,
    user_id INTEGER,
    analysis_type TEXT,
    result_id INTEGER,
    file_path TEXT,
    analysis_name TEXT,
    timestamp TEXT
);

```

```

CREATE TABLE ChatMessage (
    id INTEGER PRIMARY KEY,
    user_id INTEGER,
    context_id INTEGER,
    sender TEXT,
    message TEXT,
    timestamp TEXT
);

```

Each of the three Result tables `RealtimeResult`, `PCAPResult`, and `MonitorResult` contains a timestamped CSV that shows the outcome of a specific analysis. These user-associated records enable persistent and auditable multi-session support.

The `ChatContext` table stores metadata for each interaction session, linking it to the corresponding analysis and any associated files. The `ChatMessage` table maintains the full conversation history between user and assistant, including timestamps and sender roles. This structure supports seamless context recovery across sessions and facilitates persistent logging for audit or follow-up.

4.6. Integration, Testing, and Implementation Challenges

Extensive integration testing was performed to validate the correct functioning of all modules. Each component was sequentially tested to confirm that:

- Zeek logs were correctly parsed into the expected feature format;
- live preprocessing produced identical outputs to the offline pipeline;

- the binary and multiclass models were executed in the correct sequence;
- the dashboard updated automatically with new detections;
- and the chatbot responses remained restricted to the current session context.

Several implementation challenges were encountered during development. To avoid reprocessing of previously seen logs, file offset tracking was implemented. Concurrency issues between Flask routes and background operations were mitigated using shared-variable locks to ensure thread safety. Since the chatbot depends on an external API, error handling was added to manage missing API keys, timeouts, and rate-limit exceptions.

4.7. Security, Privacy, and Scalability

The prototype implements basic but functional security that ensures user isolation and controlled access while maintaining a research-grade data-sharing design. Each registered user has an independent SQLite database for analysis outputs and chatbot interactions. Authentication is required and there is no cross-user access.

Unlike simplified IDS prototypes that discard payloads, this system stores limited payload content to preserve forensic context and support the chatbot’s interpretation. When interacting with the OpenAI API, the backend exports a CSV summary of recent flows (including IPs/ports and possibly small payload fragments) so the model can reason about current conditions. At this stage, these values are not anonymised prior to transmission; a production deployment would add a sanitisation layer (hashing/masking of IPs, hostnames, file paths, and payload snippets) and an allow-list of attributes.

Locally, all components (Zeek, Flask, ML inference, and SQLite) execute on the same host; API credentials are provided via environment variables (not hard coded). The implementation follows a method under resource constraints: CPU-only models, batched inference every few seconds, and lightweight CSV interchange.

From a scalability perspective, traffic capture, inference, database and chatbot can be containerised and deployed independently. Communication via CSV and Representational State Transfer (REST) endpoints enables horizontal scale or distributed/cloud deployments once data-protection layers (sanitisation, encryption in transit/at rest) are added.

4.8. Artificial Intelligence in Code Generation, Writing, and Bibliography

AI also played a practical and supportive role throughout the development of this project. Part of the software implementation was initially drafted using **GitHub Copilot**, an AI-powered code completion tool based on large language models trained on open-source repositories. Copilot was employed to accelerate the prototyping of certain modules, such as Flask routes, database schema definitions, and the interface logic between the monitoring, inference, and chatbot layers. However, the generated code was never adopted directly without validation. Each suggestion was reviewed line by line, tested, and manually adapted to ensure functional correctness, performance efficiency, and consistency

with the system's architecture. In several cases, AI suggestions served merely as structural templates, while the actual logic and optimisations were implemented entirely by hand.

A systematic validation process was followed for all AI-assisted code generation: (1) suggestions were produced in Visual Studio Code with Copilot enabled; (2) each output was analysed, executed, and tested independently; (3) the validated code was then polished, annotated, and commented in both English and Portuguese to clarify the rationale behind the implementation decisions. This mixed approach, combining AI-assisted drafting with manual engineering, not only accelerated development but also ensured that human control and accountability remained entirely intact.

4.9. Summary

The development of the intelligent network intrusion detection system was the main subject of this chapter, which dealt with the complete process from the capturing of real-time traffic through to model inference, visualization, and conversational explanation. The prototype showed that good modular integration was achieved, stable real-time performance was maintained on non-specialised high-performance hardware, and privacy and security principles were followed. The following chapter presents the experimental evaluation of the system, including quantitative results on detection accuracy, as well as an analysis of the chatbot's impact on operational network analysis.

[This page is intentionally left blank.]

Results and Discussion

This chapter presents the results obtained from the evaluation of the proposed intrusion detection system and its integrated AI assistant. The tests were set up in such a way that they would answer all the three research questions that were introduced in Chapter 1, dealing with (1) the efficiency of the ML models that were used for intrusion detection, (2) the role played by the AI chatbot in interpreting and providing the user with mitigation steps to reduce the impact of the anomalies that were detected, and (3) the ability of the system to function under real-time conditions.

The entire series of experiments was carried out on the **TON_IoT dataset**, which has been already presented and preprocessed with the description provided in Chapter 4. This dataset consists of different types of network and IoT telemetry coming from various sources, and it offers both binary (i.e., benign vs. malicious) and multi-class labels for different types of attacks. The preprocessing steps that were used, namely feature cleaning, encoding, and normalisation, were exactly the same as the ones described in Section 4.2, thereby guaranteeing coherence between the offline training data and the real-time inference scheme. The trained models underwent evaluation on the test subset of TON_IoT in order to have their performance assessed in a strictly unbiased manner.

Three complementary testing scenarios were implemented to validate different aspects of the system:

- **Scenario 1 – Model comparison (RQ1):** evaluates the detection performance of four classical ML algorithms, Random Forest, Gradient Boosting, Logistic Regression and k -Nearest Neighbours—on the TON_IoT dataset, comparing accuracy, precision, recall, and F1-score.
- **Scenario 2 – Offline chatbot-assisted analysis (RQ2):** tests the integration between the IDS and the AI assistant using two packet captures (PCAPs) containing known attack types, DDoS and Backdoor. The chatbot’s explanations, mitigation advice, and contextual accuracy were examined to assess its operational value.
- **Scenario 3 – Real-time monitoring (RQ3):** observes the system’s behaviour under natural, attack-free network conditions. Since the testing network contained no active attacks, the results focus on responsiveness, stability, and the absence of false positives, rather than confirmed detections.

Each of the following sections corresponds to one of these scenarios and directly addresses the related research question. Quantitative results are complemented by qualitative analyses of chatbot interactions, dashboard behaviour, and reporting capabilities.

5.1. Effectiveness of a ML IDS (Model Comparison)

This section answers **RQ1**: *How effective are classical machine learning algorithms for intrusion detection using the TON_IoT dataset? Which algorithms have the best performance?* All models share the same preprocessing pipeline described in Chapter 4. Because the dataset is highly imbalanced (where attacks dominate), the deployed RF applies calibrated decision thresholds to reduce spurious alarms: if $p(\text{benign}|x) > p(\text{malign}|x)$ the flow is *benign*; otherwise, it is *malicious* only when $p(\text{malign}|x) \geq 0.80$. Low-confidence malicious predictions ($[0.80, 0.85)$) are tagged as *Unknown Traffic*, and when $p(\text{malign}|x) \geq 0.85$ the flow is assigned to the most probable attack class. This threshold tuning directly addresses the dataset imbalance and helps avoid false alarms.

Table 5.1 summarises the binary results (benign vs. malicious) and includes training time. All models achieve very high scores, with RF slightly ahead in F1 and GB close behind; LR lags in all metrics; k -NN remains strong but less scalable. Notably, GB trains faster than RF in the binary setting, but in the multiclass setting RF is *much* faster to converge while maintaining top accuracy (see Table 5.6). These results are consistent with prior TON_IoT studies reporting near-perfect ensemble performance [24], [39], [40].

TABLE 5.1. Binary classification on TON_IoT (benign vs. malicious) with training time.

Model	Accuracy	Precision	Recall	F1-score	Training Time (s)
Random Forest	0.99989	0.99995	0.99994	0.99989	503.91
Gradient Boosting	0.99898	0.99990	0.99899	0.99898	324.84
Logistic Regression	0.98930	0.98922	0.98930	0.98900	2.03
k -NN	0.99906	0.99906	0.99906	0.99906	4.20

The RF binary confusion matrix (Table 5.2) shows only **51** false positives and **63** false negatives out of more than one million flows. For transparency and comparability, the binary confusion matrices of the other models are also reported (Tables 5.3–5.5).

TABLE 5.2. Confusion matrix — Random Forest (binary).

	Pred. Benign	Pred. Malicious
Actual Benign	57,988	51
Actual Malicious	63	1,009,703

TABLE 5.3. Confusion matrix — Gradient Boosting (binary).

	Pred. Benign	Pred. Malicious
Actual Benign	57,564	475
Actual Malicious	612	1,009,154

After evaluating the binary classification, the models were tested on the multiclass attack-type identification task. Table 5.6 summarises the results, showing that all four

TABLE 5.4. Confusion matrix — Logistic Regression (binary).

	Pred. Benign	Pred. Malicious
Actual Benign	49,048	8,991
Actual Malicious	2,436	1,007,330

TABLE 5.5. Confusion matrix — k -Nearest Neighbours (binary).

	Pred. Benign	Pred. Malicious
Actual Benign	57,432	607
Actual Malicious	397	1,009,369

algorithms achieved nearly perfect accuracy. The Random Forest again provides the best trade-off between accuracy and efficiency, training almost ten times faster than Gradient Boosting (95.9s vs. 926.6s), making it a practical choice for retraining and deployment.

The corresponding Random Forest confusion matrix is shown in Table 5.7, illustrating precise classification across all four attack categories: backdoor, DDoS, MITM and scanning. The model correctly identified all classes with minimal misclassifications (only three MITM flows mislabelled as DDoS). For completeness and transparency, the multiclass confusion matrices for the remaining algorithms are presented in Tables 5.8, 5.9, and 5.10, confirming similar behaviour across ensemble and non-ensemble methods.

In summary, all models are highly effective on TON_IoT, with ensemble methods reaching essentially perfect scores, in line with the literature. RF is the most balanced choice for deployment: although GB trained faster in the binary case, RF provided better or equal accuracy, vastly faster multiclass training, and simpler parallelisation and interpretability (feature importance), making it the model adopted in the prototype.

TABLE 5.6. Multiclass classification on TON_IoT (attack type) with training time.

Model	Accuracy	Precision	Recall	F1-score	Training Time (s)
Random Forest	0.999997	0.999997	0.999997	0.999997	95.92
Gradient Boosting	1.000000	1.000000	1.000000	1.000000	926.65
Logistic Regression	0.999220	0.999220	0.999220	0.999110	46.05
k -NN	0.999957	0.999957	0.999957	0.999957	7.96

TABLE 5.7. Confusion matrix — Random Forest (multiclass).

	Backdoor	DDoS	MITM	Scanning
Backdoor (True)	61,021	0	0	0
DDoS (True)	0	193,762	0	0
MITM (True)	0	3	218	0
Scanning (True)	0	0	0	754,762

TABLE 5.8. Confusion matrix — Gradient Boosting (multiclass).

	Backdoor	DDoS	MITM	Scanning
Backdoor	61,021	0	0	0
DDoS	0	193,762	0	0
MITM	0	0	221	0
Scanning	0	0	0	754,762

TABLE 5.9. Confusion matrix — Logistic Regression (multiclass).

	Backdoor	DDoS	MITM	Scanning
Backdoor	61,007	14	0	0
DDoS	540	193,214	0	8
MITM	4	217	0	0
Scanning	0	9	0	754,753

TABLE 5.10. Confusion matrix — k -Nearest Neighbours (multiclass).

	Backdoor	DDoS	MITM	Scanning
Backdoor	61,017	1	3	0
DDoS	0	193,757	0	5
MITM	7	6	206	2
Scanning	0	13	0	754,749

5.2. Role of the AI Chatbot in Security Operations (Offline Scenarios)

This section answers **RQ2**: *How effectively can the integrated AI chatbot assist in interpreting and mitigating detected network anomalies?* Two controlled offline experiments were performed using PCAP traces representing distinct attack categories, a DDoS flood and a backdoor intrusion. Both traces were processed through the implemented IDS pipeline to test whether the system, combined with the AI assistant, could correctly identify, describe, and propose relevant countermeasures for the detected anomalies.

Chatbot-Assisted Mitigation Support for DDoS Attack

The first test evaluated the system on an *offline* PCAP containing a realistic DDoS flood. Unlike a simple ICMP flood, this attack mixed multiple transport protocols, the majority of malicious traffic was TCP-based, with a smaller UDP component, alongside nominal benign traffic. In total, the PCAP contained approximately 100,000 packets. Table 5.11 breaks down the capture by protocol, while Table 5.12 lists the top targeted IP addresses, aggregated across TCP and UDP flows.

After inference, the IDS classified each flow into one of three categories: *benign*, *unknown*, or *malicious (DDoS)*. Table 5.13 summarises the resulting breakdown, showing that benign traffic was correctly preserved while the majority of malicious flows were attributed to the DDoS class.

TABLE 5.11. Traffic distribution by protocol in the offline DDoS PCAP.

Protocol	Packet Count	Percentage
TCP (attack flood)	80,000	80%
UDP (attack flood)	10,000	10%
Other (benign)	10,000	10%

TABLE 5.12. Top targeted IP addresses during the DDoS attack (combined TCP/UDP packet counts).

Destination IP	Packet Count
192.168.1.184	50,000
192.168.1.1	25,000
192.168.1.190	15,000
192.168.1.194	6,000
192.168.1.152	4,000

TABLE 5.13. Classification results from IDS inference during offline DDoS detection.

Traffic Type	Packet Count
Benign traffic	60,303
Unknown traffic (malign)	2,561
DDoS (malign)	99,830
Total Analysed	162,694

Because the PCAP did not include ground-truth labels for every packet, the IDS’s effectiveness was evaluated qualitatively by verifying the correctness of the flagged anomalies. Benign background traffic was not misclassified with high confidence, and the unknown category primarily corresponded to low-confidence predictions, confirming that the thresholding mechanism effectively avoided false alarms.

Immediately after detection, the AI chatbot was engaged to interpret the results. The user interacted in natural language, and the assistant provided diagnostic information and corresponding mitigation steps, as illustrated below.

User: “What anomalies are present?”

Chatbot: “I have detected a high-volume network anomaly that looks like a DDoS attack. The source IP 192.168.1.184 is sending an exceptionally large number of packets – about 80,000 TCP packets and 10,000 UDP packets. The main target of this flood is 192.168.1.1 (network gateway) with 25,000 packets, followed by 192.168.1.190 (15,000 pkts) and others. This pattern indicates a TCP/UDP flood aimed at overwhelming network resources.”

Chatbot (follow-up): “To give specific advice, could you tell me what types of devices these IP addresses correspond to? For example, is

192.168.1.184 a workstation or some other device?”

User: “192.168.1.184 is a MacBook, 192.168.1.1 is our Cisco router, and 192.168.1.190 is a network switch.”

Chatbot: “Thanks. For the MacBook (.184), you can apply a temporary iptables rule to drop all incoming TCP traffic from the attacker IP. On the Cisco router (.1), add an access-list entry to block or rate-limit packets from 192.168.1.184. On the switch (.190), you may disable the port if directly connected, or configure a storm-control rule to limit the flood. These measures will block the attack traffic at multiple points.”

This dialogue demonstrates that the chatbot can translate detection outputs into concrete mitigation guidance. However, future iterations could enhance contextual awareness by automatically retrieving device information from the network inventory, this is listed in Chapter 6 as future work.

Chatbot-Assisted Mitigation Support for Backdoor Attack

To further evaluate the system’s adaptability beyond denial-of-service scenarios, a second offline PCAP trace containing a backdoor intrusion was analysed. The backdoor attack manifested as persistent connections between internal hosts and external command-and-control servers, intermixed with benign background traffic. Table 5.14 summarises the protocol distribution, while Table 5.15 lists the top destination IPs. All data originate from the 664,085 packets captured.

TABLE 5.14. Traffic distribution by protocol in the offline backdoor PCAP.

Protocol	Packet Count	Percentage
TCP	566,123	85.3%
UDP	95,289	14.4%
Other (ICMP/unknown)	2,673	0.4%

TABLE 5.15. Top destination IP addresses during the backdoor attack.

IP address	Packet Count
192.168.1.33	428,901
192.168.1.37	80,092
192.168.1.1	20,249
192.168.1.195	19,024
192.168.1.193	16,959

The IDS detected and labelled flows into benign and anomalous categories, with results detailed in Table 5.16. Normal traffic was recognised with high confidence, while anomalous flows were further divided into *backdoor*, *unknown*, and *MITM* subclasses.

TABLE 5.16. Classification results from IDS inference during offline backdoor detection.

Traffic Type	Packet Count
Normal (benign)	155,991
Backdoor (malign)	508,040
Unknown traffic	50
MITM (malign)	4
Total Analysed	664,085

Immediately after detection, the chatbot was queried to interpret and assist in the remediation of the detected threat:

User: “What anomalies are present?”

Chatbot: “I have detected persistent backdoor connections in the traffic. In total, 664,085 packets were analysed, of which 155,991 were normal and 508,094 were anomalous. Within those anomalies, 508,040 flows were labelled backdoor, 50 flows were labelled unknown traffic and 4 flows were labelled MITM.”

Chatbot (follow-up): “The top destination IPs involved in these anomalous flows are 192.168.1.33 (428,901 packets), 192.168.1.37 (80,092), 192.168.1.1 (20,249), 192.168.1.195 (19,024) and 192.168.1.193 (16,959). Could you tell me what type of device each of these IP addresses corresponds to?”

User: “192.168.1.33 and 192.168.1.37 are MacBook hosts, 192.168.1.193 and 192.168.1.195 are Windows PCs, and 192.168.1.1 is a Juniper router.”

Chatbot: “Thanks. Switching to *mitigation mode*. Based on the mappings you provided, here is a plan to contain and remediate the backdoor activity: create and apply a firewall filter on the Juniper router to block traffic to compromised endpoints; block SSDP/UPnP (a common backdoor transport) at the router; quarantine the compromised hosts on a separate VLAN; patch and scan each Windows and macOS endpoint; harden the Juniper router; and enable detailed logging and monitoring for re-infection.”

This dialogue confirms that the assistant can articulate the scope of the attack and outline consistent mitigation steps aligned with standard network security practices.

Chatbot Response Verification

To verify the accuracy and reliability of the chatbot’s mitigation suggestions, each response was manually cross-checked against the official documentation of the respective

platforms: Linux `iptables`¹, Cisco Access Control Lists (ACLs)², and Juniper firewall filters.³ This validation step was essential given that expert evaluation for each platform was not feasible. By directly comparing the chatbot’s responses with these authoritative sources, it was possible to assess their technical correctness and practical applicability. Rather than assigning a subjective score, the analysis focused on the factual accuracy, contextual relevance, and actionability of each suggestion.

TABLE 5.17. Verification of chatbot factual correctness for the DDoS scenario.

Criterion	Correct (Yes/No)	Comment
Detection accuracy	Yes	Correct identification of DDoS behaviour (TCP/UDP flood).
Mitigation relevance	Yes	Suggested actions match standard Linux <code>iptables</code> usage and Cisco ACL configuration.
Context awareness	Partial	Device role (router, switch, workstation) was not inferred automatically and had to be provided by the user.
Clarity of response	Yes	Wording is understandable to a non-expert.

TABLE 5.18. Verification of chatbot factual correctness for the backdoor scenario.

Criterion	Correct (Yes/No)	Comment
Detection accuracy	Yes	Correctly summarised the proportion of anomalous traffic and identified the attack type as a persistent backdoor connection.
Mitigation relevance	Yes	Proposed actions (firewall filter on the Juniper router, VLAN isolation, endpoint patching and scanning) are consistent with standard hardening procedures for C2-style backdoors.
Context awareness	Partial	Remediation steps depended on manual identification of which IPs mapped to which physical devices.
Clarity of response	Yes	Terminology and instructions are clear for a semi-expert network operator.

The verification showed that every suggestion made by the chatbot was completely correct from a technical point of view and in accordance with the official device documentation. A limitation was identified in the area of contextual automation, as the chatbot

¹<https://linux.die.net/man/8/iptables>

²https://www.cisco.com/c/en/us/td/docs/ios/security/command/reference/sec_book.html

³https://www.juniper.net/documentation/en_US/junos/topics/reference/configuration/firewall-filters.html

was unable to automatically infer the target device and relied on explicit user input. This limitation is expected to be resolved in a future version of the system.

A visual representation of the chatbot interface used during these offline tests is provided in Appendix A (Figure A.2), illustrating how the assistant presents contextual explanations and mitigation advice to the operator.

5.3. Real-Time Monitoring and System Responsiveness

This section addresses **RQ3**: *How does the system behave under continuous, real-time monitoring conditions?* Unlike the previous offline evaluations (RQ1–RQ2), this experiment focused not on measuring detection accuracy, but on verifying whether the monitoring components could operate continuously, updating statistics in real time, and maintaining interface stability.

A 30-minute test was performed using the system’s *24/7 monitoring* mode, launched through the dashboard control panel. The Flask backend continuously ingested Zeek traffic, processed each flow through the trained RF classifier, and updated the dashboard every 20 seconds with new visual data. The interface remained responsive throughout the session, and no false positives were triggered under normal traffic.

Because the environment used for this test contained only benign traffic, no attacks were detected. Therefore, while the interface successfully visualised live statistics, it was not possible to empirically verify the accuracy of real-time anomaly detection. As such, these results demonstrate the *functional capability* of the platform to operate in real time, but not the full reliability of its detection in a live attack scenario. Latency, throughput, and detection precision may vary under heavier or malicious loads, and should be evaluated in future controlled experiments.

Figures 5.1 illustrate the 24/7 monitor control where start and stop monitoring can be performed by the user.

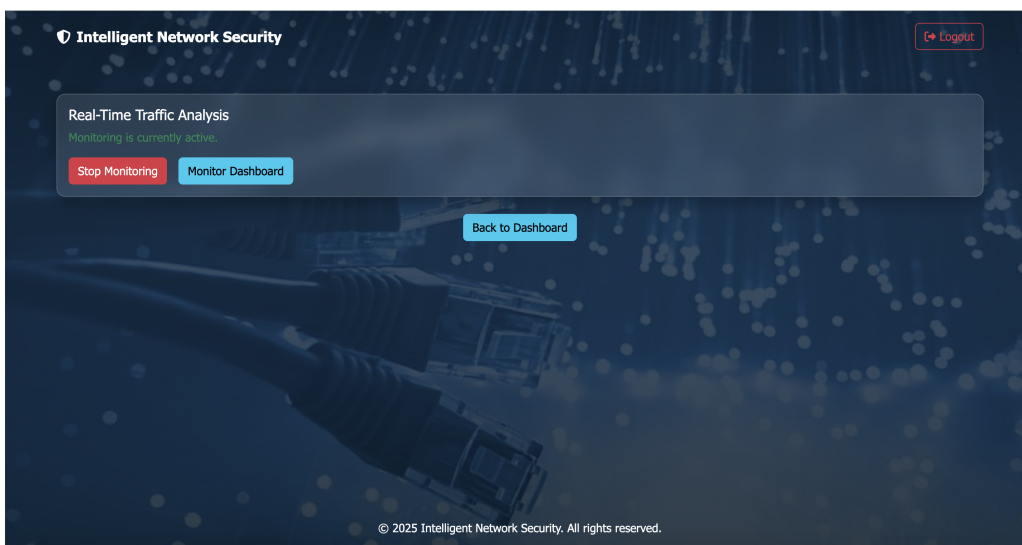


FIGURE 5.1. Dashboard – Realtime monitoring control.

Figure 5.2 presents the real-time dashboard view, where a time-series diagram updates every 20 seconds to show the total packets captured and processed.

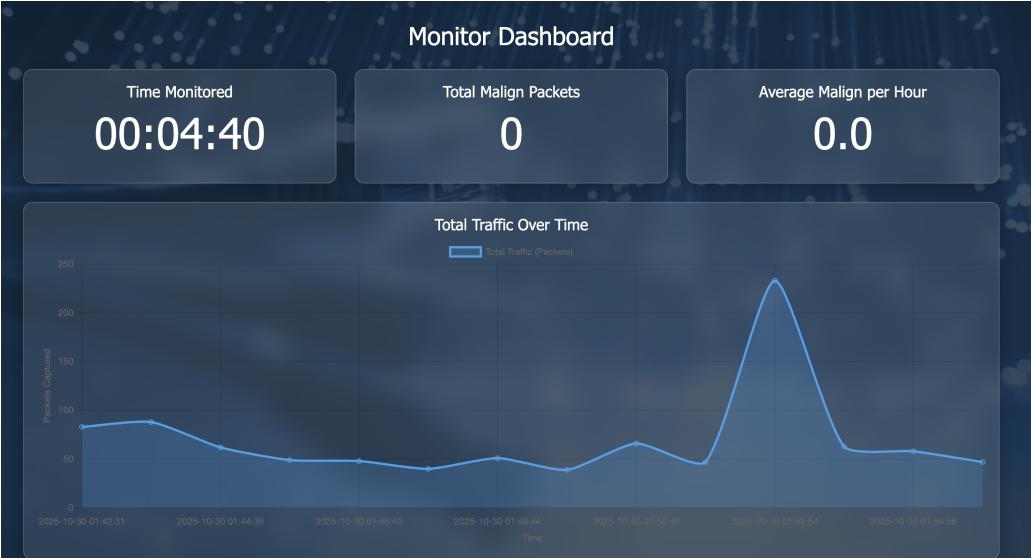


FIGURE 5.2. Dashboard – Monitor Dashboard.

Finally, Figure 5.3 displays the statistical view summarising key flow features such as source IP.

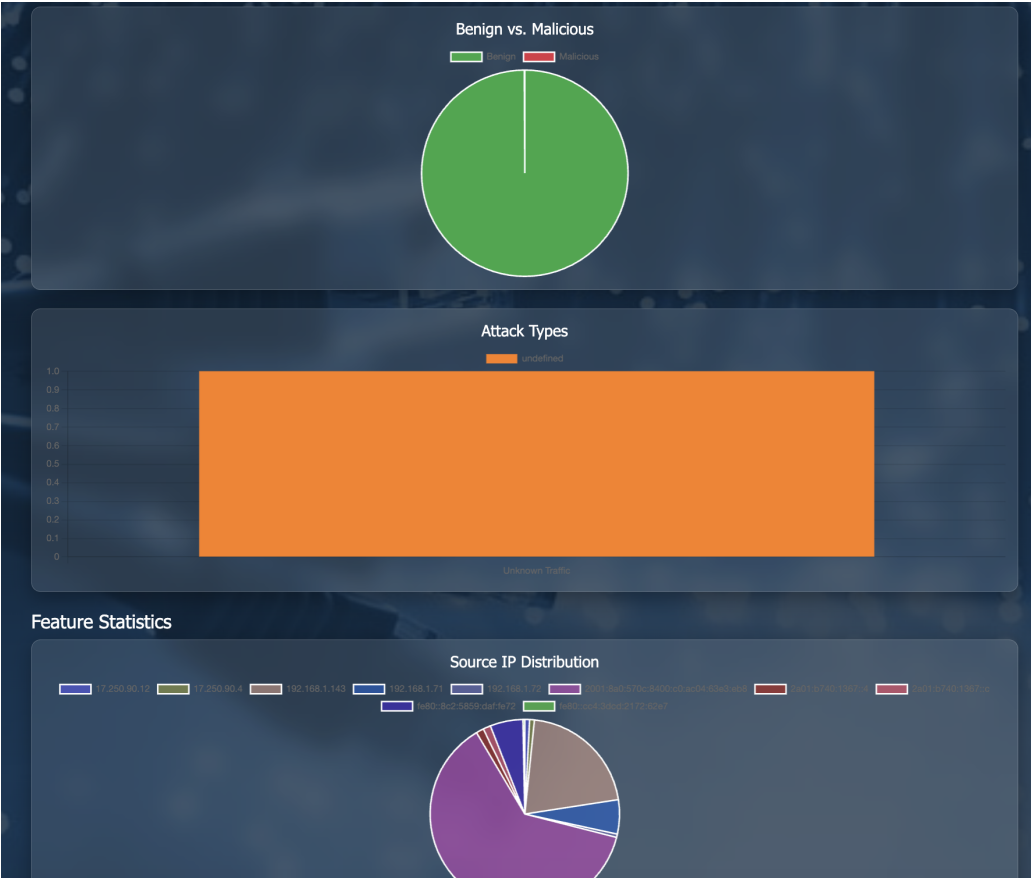


FIGURE 5.3. Dashboard – Real-time statistics.

Together, these components demonstrate that the prototype is able to maintain continuous monitoring, data aggregation, and real-time visualisation without interruption.

In general, the foregoing remarks indicate that the new system could continuously work and give real-time network statistics without the degradation of performance. The prototype accomplished stable real-time data input and visual feedback, which proved the technical aspect of the system to be ready for 24/7 operation. Nevertheless, since the network used for the tests did not have any ongoing attacks, the intrusion detection accuracy in real-time settings is still unproved. Therefore, RQ3 could be regarded as partially answered: the system is capable of real-time operation from the functionality aspect but still needs to be tested in the conditions of live attack for the full effectiveness of IDS to be confirmed.

[This page is intentionally left blank.]

Conclusion and Future Work

Conclusion

The thesis focused on an integrated intrusion detection and AI-chatbot system that was designed, implemented, and evaluated. The system consists of a classical machine learning (ML) model and a conversational AI assistant that interpret and helps the user to mitigate network anomalies together. The primary intent was to develop a user-friendly solution that would be able to guide the security analysts during the anomaly detection process, providing not only high detection performance but also human-readable instructions via natural language.

The implemented system was relying on overall traffic parsing, feature extraction, and supervised ML classifiers trained on the **TON_IoT dataset**. These classifiers carried out both binary (benign vs. malicious) and multiclass (attack-type) prediction. The classifiers supported an AI chatbot which was able to explain results, offer mitigation advice, and convert IDS outputs into context-aware recommendations in plain English. The chatbot was rule-based and has context-driven behavior rather than being autonomous. Its complexity was purposely limited in order to provide deterministic and verifiable responses.

The research questions defined in Chapter 1 can now be answered as follows:

RQ1 – Effectiveness of ML algorithms for intrusion detection. Four different algorithms, namely Random Forest, Gradient Boosting, Logistic Regression, and k -Nearest Neighbours, were put to the test and their performance was compared using the TON_IoT dataset. All models performed almost perfectly, which proves that ensemble methods are strong contenders even on this dataset. In the end, the Random Forest classifier was the overall winner in terms of accuracy and speed, with an F1-score of 0.99989 for binary classification and 0.999997 for multiclass identification. This excellent performance is the reason for its choice as the model to be used in the prototype.

RQ2 – Role of the AI chatbot in security operations. The chatbot not only correctly interpreted IDS outputs but also gave correct mitigation advice in the two offline scenarios (DDoS and backdoor) that were performed. A confirmation of the facts indicated that the suggestions made by the chatbot corresponded to the documents of Linux, Cisco, and Juniper considered official. Nevertheless, the chatbot needed to be informed by humans when it came to recognizing the roles of the devices (e.g., router, switch, workstation), which in turn affected its understanding of the context. Nonetheless, all the replies were very well grounded technically, easy to follow and appropriate for the professional working at the network.

RQ3 – Real-time monitoring capability. Monitoring the network continuously in a harmless network environment. The monitoring system proved to be stable, quick in processing requests, and its visualization of the live data was absolutely correct, with a refreshing of the dashboard every 20 seconds. The presence of no attacks made it impossible to completely verify the real-time detection of intrusions. Therefore, it can be concluded that the system is capable of working in real-time conditions, but its performance under live attack traffic has to be experimentally verified.

In general, the findings reveal that the created system not only fulfills its design objectives but also: it acquires the highest accuracy on the TON_IoT dataset, encompasses a transparent AI assistant for explainable mitigation, and runs efficiently on basic hardware. Empirical decision thresholds were established in this work to reduce the number of false positives in a highly unbalanced dataset, however, this technique may also result in unresponsiveness to attacks with low confidence. This dilemma is the outcome of a well-thought-out design that gives priority to reliability and interpretability over extensive sensitivity.

To conclude, the present work offers proof of the possibility to integrate ML-based intrusion detection with conversational AI for the sake of amplifying situational awareness. Moreover, it presents a lightweight, modular, and interpretable prototype that connects automated threat detection to human decision-making.

Future Work

There are various research paths that can not only extend but also reinforce this project:

- **Real-time validation:** Experiment with the platform in live-attack settings that are controlled to calculate the parameters of latency, throughput, and reliability in real-time detection.
- **Context automation:** Incorporate automatic identification of device-type (e.g., through SNMP or ARP mapping) such that the chatbot is capable of inferring contextual roles without the need for any user input.
- **Adaptive thresholds:** Set up the changing of classification thresholds through online learning or feedback from operator confirmation in order to cut down on the false negatives.
- **Explainability and transparency:** The chatbot’s power to produce visual explanations (e.g., feature importances, rule traces) for the alerts it communicates should be broadened.
- **Security governance:** Make audit logging and access control for prompt-injection testing and safe deployment of language models in security environments part of the system.
- **Data anonymization and privacy preservation:** While this work discusses the need for anonymisation when exporting traffic summaries to external APIs, future versions of the system should incorporate a dedicated sanitisation layer.

This may include hashing or tokenisation of IP addresses, masking of hostnames and file paths, and configurable attribute allow-lists, enabling deployment in privacy-sensitive environments without compromising analytical value.

- **Dataset imbalance mitigation:** The TON_IoT dataset used in this work exhibits a strong class imbalance, which may bias model training and evaluation. Future work should explore advanced imbalance-handling strategies, such as targeted resampling techniques, cost-sensitive learning, and class-weighted loss functions, as well as validation across multiple datasets to assess generalisation beyond a single imbalanced benchmark.

Future development should focus on testing the prototype in realistic, heterogeneous network environments and further integrating the chatbot into real-time monitoring loops. Such advancements would bring the system closer to deployment as an interpretable and trustworthy AI-driven network defence assistant.

Recent approaches also explore integrated architectures for automated vulnerability mitigation, beyond pure traffic-based intrusion detection. For instance, Polónio et al. [3] propose a framework that prioritizes and mitigates high-risk vulnerabilities in networked systems. Their design could be extended with the machine learning-driven approach presented in this work, especially in bridging detection and operator feedback loops. Furthermore, in their systematic review [41], the same authors discuss proactive mitigation techniques, often leveraging SDN, which provides a broader context for approaches aiming to reduce threat dwell time, a concern also addressed here via AI-assisted mitigation.

As this work relied on a synthetic dataset (TON_IoT), future validation in real-world IoT environments is essential to further assess the system's robustness and adaptability. Several sensor platforms and edge-based architectures previously proposed, including wireless crowd monitoring systems [42], [43], [44], and elastic edge resource provisioning for IoT services [45], may serve as practical testbeds for future deployments. Integrating the proposed detection and mitigation system into such environments would enable realistic traffic capture and evaluation under operational conditions, providing insights into usability, performance, and scalability.

[This page is intentionally left blank.]

References

- [1] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. d. Hartog, “Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets,” *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 485–496, 2022. DOI: 10.1109/JIOT.2021.3085194
- [2] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, “Ton_iottelemetrydataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems,” *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020. DOI: 10.1109/ACCESS.2020.3022862
- [3] J. Polónio, J. Moura, and R. N. Marinheiro, “Toward automatic detection and mitigation of high-risk cybersecurity vulnerabilities at networked systems,” *IEEE Access*, vol. 13, pp. 181 957–181 976, 2025. DOI: 10.1109/ACCESS.2025.3622497
- [4] D. Denning, “An intrusion-detection model,” *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987. DOI: 10.1109/TSE.1987.232894
- [5] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Network anomaly detection: Methods, systems and tools,” *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, 2014. DOI: 10.1109/SURV.2013.052213.00046
- [6] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: Techniques, datasets and challenges,” *Cybersecurity*, vol. 2, no. 1, pp. 1–26, 2019. DOI: 10.1186/s42400-019-0038-7
- [7] V. Z. Mohale and I. C. Obagbuwa, “Evaluating machine learning-based intrusion detection systems with explainable AI: Enhancing transparency and interpretability,” *Frontiers in Computer Science*, vol. 7, p. 1 520 741, 2025. DOI: 10.3389/fcomp.2025.1520741
- [8] R. Kalakoti, R. Vaarandi, H. Bahşi, and S. Nõmm, “Evaluating explainable ai for deep learning-based network intrusion detection system alert classification,” in *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP, INSTICC*, SciTePress, 2025, pp. 47–58, ISBN: 978-989-758-735-1. DOI: 10.5220/0013180700003899
- [9] A. Thakkar and R. Lohiya, “A review on challenges and future research directions for machine learning-based intrusion detection system,” *Archives of Computational Methods in Engineering*, vol. 30, pp. 4245–4269, 2023. DOI: 10.1007/s11831-023-09943-8

- [10] Z. Xu, Y. Wu, S. Wang, J. Gao, T. Qiu, Z. Wang, H. Wan, and X. Zhao, *Deep learning-based intrusion detection systems: A survey*, 2025. arXiv: 2504.07839 [cs.CR]. [Online]. Available: <https://arxiv.org/abs/2504.07839>
- [11] J. Lansky, S. Ali, M. Mohammadi, M. K. Majeed, S. H. T. Karim, S. Rashidi, M. Hosseinzadeh, and A. M. Rahmani, "Deep learning-based intrusion detection systems: A systematic review," *IEEE Access*, vol. 9, pp. 101 574–101 599, 2021. DOI: 10.1109/ACCESS.2021.3097247
- [12] A. Kovari, "Explainable ai chatbots towards xai chatgpt: A review," *Heliyon*, vol. 11, no. 2, e42077, 2025, ISSN: 2405-8440. DOI: <https://doi.org/10.1016/j.heliyon.2025.e42077> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2405844025004578>
- [13] V. Z. Mohale and I. C. Obagbuwa, "A systematic review on the integration of explainable artificial intelligence in intrusion detection systems to enhancing transparency and interpretability in cybersecurity," *Frontiers in Artificial Intelligence*, vol. Volume 8 - 2025, 2025, ISSN: 2624-8212. DOI: 10.3389/frai.2025.1526221 [Online]. Available: <https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2025.1526221>
- [14] M. A. Talukder, M. M. Islam, M. A. Uddin, K. F. Hasan, S. Sharmin, S. A. Alyami, and M. A. Moni, "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," *Journal of Big Data*, vol. 11, no. 1, p. 33, 2024. DOI: 10.1186/s40537-024-00886-w [Online]. Available: <https://doi.org/10.1186/s40537-024-00886-w>
- [15] M. A. Hossain, "Deep q-learning intrusion detection system (dq-ids): A novel reinforcement learning approach for adaptive and self-learning cybersecurity," *ICT Express*, vol. 11, no. 5, pp. 875–880, 2025, ISSN: 2405-9595. DOI: <https://doi.org/10.1016/j.icte.2025.05.007> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2405959525000694>
- [16] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6. DOI: 10.1109/CISDA.2009.5356528
- [17] N. Moustafa and J. Slay, "Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6. DOI: 10.1109/MilCIS.2015.7348942
- [18] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy -*

- Volume 1: ICISSP*, INSTICC, SciTePress, 2018, pp. 108–116, ISBN: 978-989-758-282-0. DOI: 10.5220/0006639801080116
- [19] J. Ashraf, M. Keshk, N. Moustafa, M. Abdel-Basset, H. Khurshid, A. D. Bakhshi, and R. R. Mostafa, “Iotbot-ids: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities,” *Sustainable Cities and Society*, vol. 72, p. 103041, 2021, ISSN: 2210-6707. DOI: <https://doi.org/10.1016/j.scs.2021.103041> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2210670721003255>
- [20] N. Moustafa, S. Garg, E. Stinkova, T. Jones, and C. Sioutis, “A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_{iot} datasets,” *Sustainable Cities and Society*, vol. 72, p. 102994, May 2021. DOI: 10.1016/j.scs.2021.102994
- [21] R. A. Disha and S. Waheed, “Performance analysis of machine learning models for intrusion detection system using gini impurity-based weighted random forest (giwrf) feature selection technique,” *Cybersecurity*, vol. 5, no. 1, p. 1, 2022. DOI: 10.1186/s42400-021-00103-8 [Online]. Available: <https://doi.org/10.1186/s42400-021-00103-8>
- [22] M. Ali, K. Thakur, S. Schmeelk, J. Debello, and D. Dragos, “Deep learning vs. machine learning for intrusion detection in computer networks: A comparative study,” *Applied Sciences*, vol. 15, p. 1903, Feb. 2025. DOI: 10.3390/app15041903
- [23] G. Guo, X. Pan, H. Liu, F. Li, L. Pei, and K. Hu, “An iot intrusion detection system based on ton iot network dataset,” in *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, 2023, pp. 0333–0338. DOI: 10.1109/CCWC57344.2023.10099144
- [24] C. K. K. Reddy, P. A. Reddy, P. S. Reddy, M. Shuaib, S. Alam, S. Ahmad, and A. Rajaram, “Twined ensemble framework for network security: Integrating random forest, adaboost, and gradient boosting for enhanced intrusion detection,” *Discover Internet of Things*, vol. 5, no. 1, p. 107, 2025.
- [25] M. Nikhitha and D. Jabbar, “K nearest neighbor based model for intrusion detection system,” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, pp. 2258–2262, Jul. 2019. DOI: 10.35940/ijrte.B2458.078219
- [26] S. Chalichalamala, N. Govindan, and R. Kasarapu, “Logistic regression ensemble classifier for intrusion detection system in internet of things,” *Sensors*, vol. 23, p. 9583, Dec. 2023. DOI: 10.3390/s23239583
- [27] A. J. Liu, A. Mukherjee, L. Hu, J. Chen, and V. N. Nair, *Performance and interpretability comparisons of supervised machine learning algorithms: An empirical study*, 2022. arXiv: 2204.12868 [stat.ML]. [Online]. Available: <https://arxiv.org/abs/2204.12868>
- [28] M. R. E. aoun, L. N. Tidjon, B. Rombaut, F. Khomh, and A. E. Hassan, *An empirical study of library usage and dependency in deep learning frameworks*, 2022. arXiv:

- 2211.15733 [cs.SE]. [Online]. Available: <https://arxiv.org/abs/2211.15733>
- [29] A. Ozkan-Okay, U. Kose, and E. Sert, “A novel feature selection approach to classify intrusion detection system data using machine learning algorithms,” *Applied Sciences*, vol. 13, no. 19, p. 11067, 2023. DOI: 10.3390/app131911067 [Online]. Available: <https://www.mdpi.com/2076-3417/13/19/11067>
- [30] J. García, J. Entrena, and Á. Alesanco, “Empirical evaluation of feature selection methods for machine learning based intrusion detection in iot scenarios,” *Internet of Things*, vol. 28, p. 101367, 2024, ISSN: 2542-6605. DOI: <https://doi.org/10.1016/j.iot.2024.101367> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2542660524003081>
- [31] D. Guttikonda, D. Indran, L. Narayanan, T. Pasarad, and S. B. J., “Explainable ai: A retrieval-augmented generation based framework for model interpretability,” in *Proceedings of the 17th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART, INSTICC, SciTePress*, 2025, pp. 948–955, ISBN: 978-989-758-737-5. DOI: 10.5220/0013241300003890
- [32] M. Franco, B. Rodrigues, E. John Scheid, A. Jacobs, C. Killer, L. Granville, and B. Stiller, “Secbot: A business-driven conversational agent for cybersecurity planning and management,” Nov. 2020, pp. 1–7. DOI: 10.23919/CNSM50824.2020.9269037
- [33] T. Ali and P. Kostakos, *Huntgpt: Integrating machine learning-based anomaly detection and explainable ai with large language models (llms)*, 2023. arXiv: 2309.16021 [cs.CR]. [Online]. Available: <https://arxiv.org/abs/2309.16021>
- [34] P. R. B. Houssel, P. Singh, S. Layeghy, and M. Portmann, “Towards explainable network intrusion detection using large language models,” in *2024 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BD-CAT)*, 2024, pp. 67–72. DOI: 10.1109/BDCAT63179.2024.00021
- [35] H. Kheddar, “Transformers and large language models for efficient intrusion detection systems: A comprehensive survey,” *Information Fusion*, vol. 124, p. 103347, 2025, ISSN: 1566-2535. DOI: <https://doi.org/10.1016/j.inffus.2025.103347> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1566253525004208>
- [36] P. R. Houssel, S. Layeghy, P. Singh, and M. Portmann, “Ex-nids: A framework for explainable network intrusion detection leveraging large language models,” *Computers and Electrical Engineering*, vol. 129, p. 110826, 2026, ISSN: 0045-7906. DOI: <https://doi.org/10.1016/j.compeleceng.2025.110826> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045790625007694>
- [37] K. Jerabek, J. Koumar, J. Setinský, and J. Pesek, “Explainable anomaly detection in network traffic using llm,” in *NOMS 2025-2025 IEEE Network Operations and*

- Management Symposium*, 2025, pp. 1–6. DOI: 10.1109/NOMS57970.2025.11073574
- [38] M. Klesel and H. F. Wittmann, “Retrieval-augmented generation (rag),” *Business & Information Systems Engineering*, vol. 67, no. 4, pp. 551–561, 2025. DOI: 10.1007/s12599-025-00945-3 [Online]. Available: <https://doi.org/10.1007/s12599-025-00945-3>
- [39] S. Abiramasundari and V. Ramaswamy, “Distributed denial-of-service (ddos) attack detection using supervised machine learning algorithms,” *Scientific Reports*, vol. 15, no. 1, p. 13098, 2025. DOI: 10.1038/s41598-024-84879-y [Online]. Available: <https://doi.org/10.1038/s41598-024-84879-y>
- [40] M. S. Sawah, H. Elmannai, A. A. El-Bary, K. Lotfy, and O. E. Sheta, “Distributed denial of service (ddos) classification based on random forest model with backward elimination algorithm and grid search algorithm,” *Scientific Reports*, vol. 15, no. 1, p. 19063, 2025. DOI: 10.1038/s41598-025-03868-x [Online]. Available: <https://doi.org/10.1038/s41598-025-03868-x>
- [41] J. Polónio, J. Moura, and R. Neto Marinho, “On the road to proactive vulnerability analysis and mitigation leveraged by software defined networks: A systematic review,” *IEEE Access*, vol. 12, pp. 98546–98566, 2024. DOI: 10.1109/ACCESS.2024.3429269
- [42] T. M. d. Santos, R. N. Marinho, and F. B. e. Abreu, “Wireless crowd detection for smart overtourism mitigation,” in *Smart Life and Smart Life Engineering: Current State and Future Vision*, E. Kornysheva, R. Deneckère, and S. Brinkkemper, Eds. Cham: Springer Nature Switzerland, 2025, pp. 237–258, ISBN: 978-3-031-75887-4. DOI: 10.1007/978-3-031-75887-4_11
- [43] T. Vieira, S. Paiva, R. N. Marinho, and F. Brito e Abreu, “Building optimized wireless pedestrian counting sensors combining passive and active technologies,” in *2025 Sixteenth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2025, pp. 574–581. DOI: 10.1109/ICUFN65838.2025.11170080
- [44] R. D. d. Silva, R. N. Marinho, and F. B. e. Abreu, “Crowding detection combining trace elements from heterogeneous wireless technologies,” in *2019 22nd International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2019, pp. 1–6. DOI: 10.1109/WPMC48795.2019.9096131
- [45] P. Cardoso, J. Moura, and R. N. Marinho, “Elastic provisioning of network and computing resources at the edge for iot services,” *Sensors*, vol. 23, no. 5, p. 2762, 2023. DOI: 10.3390/s23052762

[This page is intentionally left blank.]

APPENDIX A

Additional Figures and Article

This appendix provides an example of the structured CSV context generated by the real-time inference pipeline. And additional UI figures.

The excerpt below uses rows taken from the public **TON_IoT** dataset; values are shown verbatim to support reproducibility.

Example of CSV Context

The following excerpt shows a portion of the CSV file produced by the inference engine. Each row corresponds to a network flow analysed by the models.

```
ts,src_ip,src_port,dst_ip,dst_port,proto,service,duration,orig_bytes,  
resp_bytes,conn_state,src_pkts,dst_pkts,label,type  
  
1556203759,192.168.1.30,42756,192.168.1.184,443,tcp,-,0,0,0,S0,1,0,1.0,ddos  
1556203759,192.168.1.180,57388,199.15.84.131,43,tcp,-,0,15,2859,SF,8,7,0.0,  
normal  
1556203760,192.168.1.30,42104,192.168.1.184,443,tcp,-,0,0,0,S0,1,0,1.0,ddos  
1556203756,192.168.1.30,42572,192.168.1.1,53,udp,dns,0,84,424,SF,2,2,1.0,  
ddos
```

FIGURE A.1. Excerpt of the CSV context produced by the inference engine.

The backend continuously reads this file and aggregates the most recent flows into summary statistics — such as total counts, benign vs. malicious ratios, and per-attack breakdowns — which are then used to build the chatbot’s contextual input.

Dashboard Components

The following figure illustrate the chatbot pane.

- **Chatbot pane:** An interactive chat window where the user can query the system about the network status or request mitigation guidance. The chatbot’s responses appear in this pane, providing explanations or actionable steps.
- **Model Evaluation:** Displays the model accuracy, confusion matrices and top features for either binary and multiclass.

First Page Of Article To Be Submitted

This appendix includes the first page of the scientific article ready to be submitted for publication, up to and including the abstract and keywords, as shown in Figure A.4.

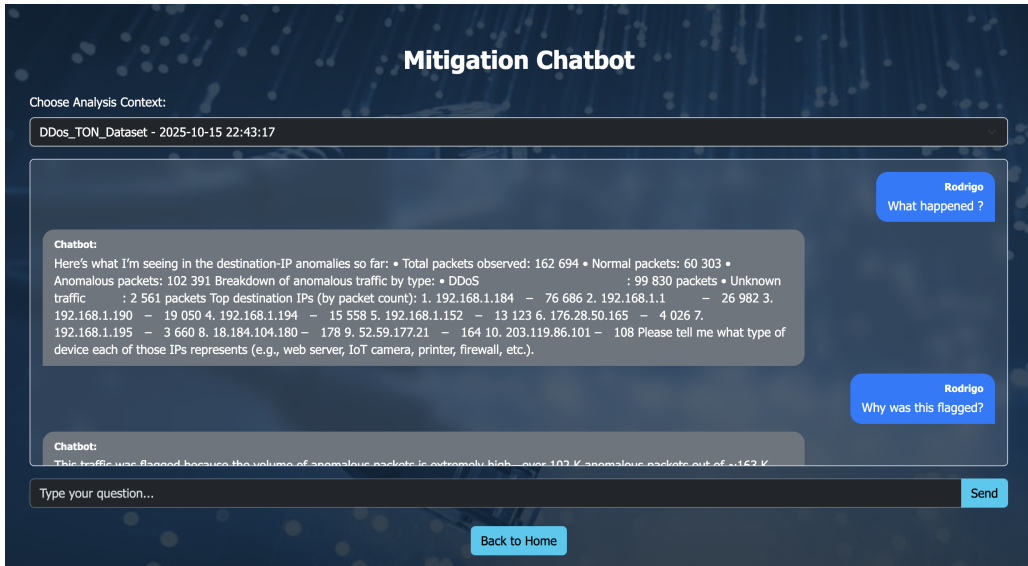


FIGURE A.2. Dashboard – Chatbot pane.

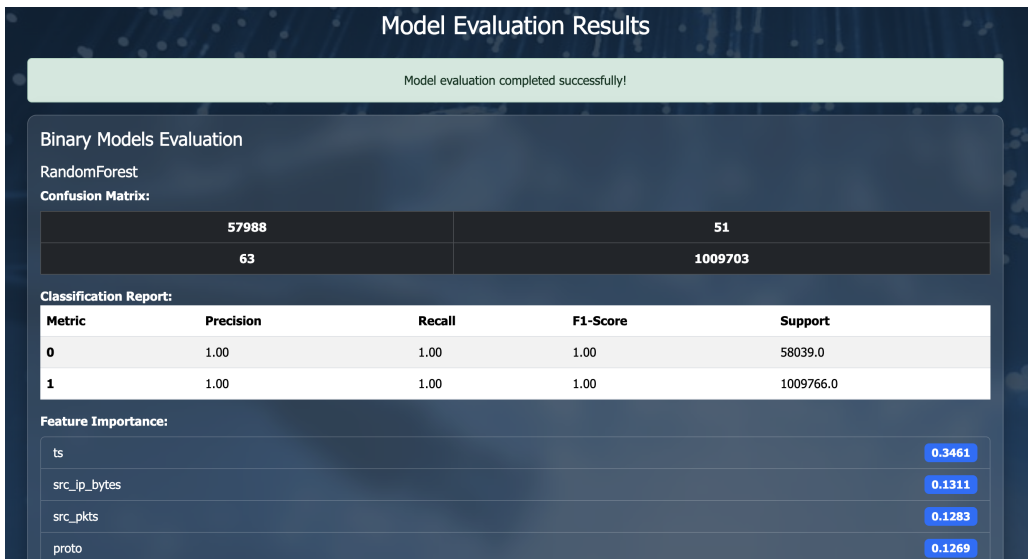


FIGURE A.3. Dashboard – Model Evaluation.

Toward Hybrid Network Security: Machine Learning-based Anomaly Detection with AI Chatbot-assisted Mitigation Support

João Rodrigo Pinto Rabuge^a, Rui Miguel Neto Marinheiro^b

^aDepartment of Information Science and Technology, ISCTE – Instituto Universitário de Lisboa, Portugal

ARTICLE HISTORY

Compiled December 23, 2025

ABSTRACT

Modern network infrastructures face a rising volume and diversity of cyber attacks, which makes intelligent anomaly detection and rapid mitigation support operationally critical. This work proposes an intrusion detection and response framework that combines supervised Machine Learning (ML) for anomaly detection with an AI-driven conversational assistant that supports mitigation guidance in natural language. Several classical ML models, Random Forest, Gradient Boosting, Logistic Regression and k-Nearest Neighbours, were trained and evaluated on the TON_IoT dataset, a recent multi-source dataset containing labelled benign and malicious traffic, including IoT/IIoT scenarios. The best-performing model was integrated into a real-time monitoring platform that uses Zeek to capture live traffic and classify flows as benign or malicious, and if malicious, into specific attack classes such as DDoS, Backdoor or MITM. A web dashboard presents live statistics, alerts and flow summaries, while a chatbot interface explains detections, contextualises network behaviour and proposes actionable mitigation steps aligned with standard hardening practices for routers, firewalls and endpoints.

Results show that the selected Random Forest model achieves near perfect precision, recall and F1-score on TON_IoT, F1 approximately 0.9999 in binary classification, while remaining computationally lightweight enough to run inference continuously on commodity hardware (Apple M3 class laptop, no discrete GPU). The integrated assistant produces mitigation playbooks that were manually verified against vendor documentation (iptables, Cisco ACLs, Juniper firewall filters) for technical correctness and practicality. The prototype demonstrates that coupling ML-based anomaly detection with an AI chatbot can bridge the gap between “alert raised” and “action taken”, improving interpretability and shortening operator response time in settings similar to a SOC.

This work contributes: (i) a layered architecture that links traffic capture, anomaly detection and human-facing guidance in a single loop; (ii) an evaluation of supervised ML models on TON_IoT for both binary and multiclass intrusion detection; and (iii) an AI-driven mitigation assistant that supports incident response using context from live network analysis instead of static threat intelligence feeds.

KEYWORDS

Intrusion Detection System; Anomaly Detection; Machine Learning; SOC Automation; Chatbot; Explainability; TON_IoT; Zeek; Random Forest

CONTACT João Rodrigo Pinto Rabuge. Email: jrpre1@iste-iul.pt, Rui Miguel Neto Marinheiro. Email: rui.marinheiro@iscte-iul.pt

FIGURE A.4. First page of the article, including title, authors, abstract and keywords.