A Hybrid Machine Learning Network Intrusion Detection System for Small and Medium Enterprises (SMEs)

ABSOLOM ORIANGA

S23B23/075

MARK TRAVIS LUFENE

S23B23/032

ALLAN KAGIMU SSEBATTA

S23B23/057

A PROJECT REPORT SUBMITTED TO THE FACULTY OF ENGINEERING, DESIGN AND TECHNOLOGY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF A DEGREE OF BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY OF UGANDA CHRISTIAN UNIVERSITY

May, 2026

UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

Declaration

We, Absolom Orianga, Mark Travis Lufene, and Allan Kagimu Ssebatta, declare to the best of our knowledge that the work in this report is original and has not been submitted before to any university or institution. It is the result of our own effort.

Absolom Orianga Reg No: S23B23/075

Signature: _____ Date: _____

Mark Travis Lufene Reg No: S23B23/032

Signature: _____ Date: _____

Allan Kagimu Ssebatta Reg No: S23B23/057

Signature: _____ Date: _____

Ethical Generative AI Usage Declaration

Students may use generative artificial intelligence (AI) ethically as a "study buddy" or assistant rather than a replacement for critical thinking. This includes citing AI use and fact-checking all outputs to maintain academic integrity.

For purposes of this assignment, the following guidelines apply.

i. Generative AI may be used for brainstorming, and generating ideas to improve your work. You may also use AI for literature search, and summarizing journal papers for quick analysis and assessing relevance of publications. However, once the relevant publications have been identified, a student is expected to fully review the selected publications, identifying problems addressed, methods used, gaps identified and recommendations made.

ii. Where AI has been used as guided in (i) above, full disclosure of how it has been used should be made in the appendix C, including how AI was used, the AI tools (including model versions), and a compilation on major AI prompts used.

iii. NO GENERATIVE AI CONTENT IS ALLOWED IN THE FINAL SUBMISSION. The final submission SHOULD be your original work.

A. **Declaring the non-use of AI for content generation**

I, Absolom Orianga, Mark Travis Lufene & Allan Kagimu Ssebatta, declare that no content generated by AI technologies has been used in this report.

Signature: _____ Date: _____

Signature: _____ Date: _____

Signature: _____ Date: _____

B. **Acknowledging the use of AI as a study buddy**

Please refer to Appendix C for the complete declaration of AI usage for generating ideas, brainstorming, literature search, and summarizing only to improve this work.

Approval

This report has been submitted for examination with the approval of the following supervisor.

Primary Advisor: Eng. Ian Raymond Osolo

Co-Advisor: Dr. Terhemba Michael-Ahile

Department of Computing & Technology

Faculty of Engineering Design & Technology

Uganda Christian University

Dr. Terhemba Michael-Ahile

Signature: _____

Date: 22/05/2026

Eng. Ian Raymond Osolo

Signature: _____

Date: 22/05/2026

Abstract

Cybersecurity remains a critical challenge for Small and Medium Enterprises (SMEs) in Uganda, where cybercrime increased by approximately 93.5% in 2024, causing losses of approximately UGX 72 billion. Approximately 40% of Ugandan SMEs have experienced cyberattacks, yet most continue to rely on basic signature-based tools that are incapable of detecting sophisticated or zero-day attacks.

This project presents Vanguard-NIDS, a hybrid machine learning-based Network Intrusion Detection System designed for real-time intrusion monitoring in SME environments. The system captures live network packets using Scapy, extracts flow-level and statistical traffic features, and analyses them through a three-pronged detection pipeline comprising signature-based pattern matching, supervised machine learning via an ensemble of Random Forest, SVM, XGBoost, and LightGBM classifiers, and unsupervised anomaly detection via Isolation Forest, One-Class SVM, and Autoencoder models. A result fusion engine combines these predictions into a unified threat score.

Models were trained and validated on the CICIDS2017, NSL-KDD, and UNSW-NB15 benchmark datasets. The Random Forest classifier achieved approximately 95% detection accuracy with a false positive rate of approximately 3%. A real-time React-based dashboard delivers live traffic monitoring, alert management, and system metrics via WebSocket communication, with an average end-to-end detection latency of under 25 milliseconds.

Keywords: Hybrid Detection, Cybersecurity, Uganda, Random Forest, Isolation Forest, Zero-Day Attacks, Real-Time Detection, FastAPI, WebSocket.

GitHub Repository: `https://github.com/Group-Firewall/Vanguard`

Dedication

We dedicate this project to the Almighty God, whose grace and strength carried us through every stage of this journey. To our families and loved ones, whose patience, encouragement, and unwavering belief in us made this work possible. And to every Small and Medium Enterprise in Uganda working to build a safer digital future, may this project be a small contribution toward that vision.

Contents

Acknowledgments

We are deeply grateful to our supervisor, **Dr. Terhemba Michael-Ahile**, for her thoughtful guidance, patience, and dedication throughout this project. She consistently challenged us to think more critically, pushed us to go beyond the surface in our research, and was always willing to engage with our ideas even at the early stages when they were still rough. Her support shaped both the direction and the depth of this work in ways we genuinely appreciate.

We also extend our sincere thanks to **Mr. Ian Raymond Osolo** for his technical input and encouragement. His practical insights into systems design and his accessibility whenever we had questions made a meaningful difference during the development phase.

To **Uganda Christian University**, thank you for the academic environment, the library resources, and the platform that made this research possible.

Finally, to our classmates and friends who sat with us through long sessions, offered feedback, and kept morale up when the project felt overwhelming, thank you. And to our families, none of this would have happened without your support.

Chapter 1

Introduction

1.1 Background

The rapid digitisation of business operations has transformed how organisations manage information and deliver services. Small and Medium Enterprises (SMEs), which form the economic backbone of many developing nations, are increasingly adopting internet-connected platforms, exposing themselves to cybersecurity vulnerabilities that did not previously exist at this scale.

1.1.1 Global Perspective

From a global perspective, cybercrime has emerged as one of the most costly threats facing businesses of all sizes. The 2024 Global Cybersecurity Index published by the ITU projects cybercrime costs to reach USD 10.5 trillion annually by 2025 [1]. SMEs are disproportionately targeted due to weaker security postures relative to large enterprises [2].

1.1.2 Regional Perspective

The African Union's 2023 Cybersecurity Report highlights East African nations, including Uganda, Kenya, and Tanzania, among the continent's most targeted [3]. Growing reliance on digital platforms without commensurate investment in cybersecurity infrastructure has created favourable conditions for malicious actors.

1.1.3 National Perspective

In Uganda specifically, the Uganda Communications Commission (UCC) reported a 93.5% increase in cybercrime incidents in 2024, resulting in losses of approximately UGX 72 billion [4]. Around 40% of Ugandan SMEs have experienced cyberattacks, yet most continue to rely on signature-based tools incapable of detecting sophisticated or novel attacks.

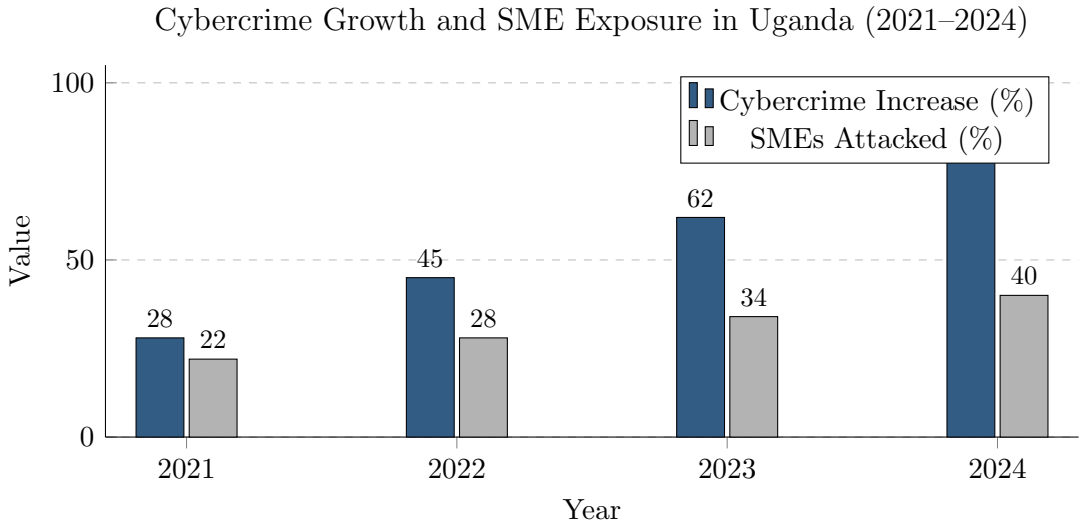

Figure 1.1: Cybercrime growth and SME cyberattack exposure in Uganda (2021–2024). Source: [4].

1.2 Problem Statement

Small and Medium Enterprises in Uganda face an escalating cybersecurity crisis. Most SMEs depend on traditional, signature-based security tools that detect only known threats. Sophisticated attacks, particularly zero-day exploits and advanced persistent threats, go entirely undetected. Enterprise-grade NIDS solutions are prohibitively expensive and technically complex for most SMEs. The absence of intelligent, real-time, and affordable intrusion detection leaves Ugandan SMEs highly susceptible to attacks that result in data theft, financial loss, reputational damage, and operational disruption.

1.3 Research Objectives

The general objective of this project is to design, develop, and evaluate Vanguard-NIDS, a hybrid machine learning-based Network Intrusion Detection System that monitors network traffic

in real time and detects both known and unknown malicious activities, specifically designed for deployment within the resource constraints of Small and Medium Enterprises in Uganda.

1.3.1 Specific Objective 1

To extract and analyse network traffic features from live packet capture data for use in real-time intrusion detection.

1.3.2 Specific Objective 2

To design and implement a hybrid detection pipeline that combines signature-based detection, supervised machine learning classification, and unsupervised anomaly detection, and to train and evaluate the machine learning models using the CICIDS2017, NSL-KDD, and UNSW-NB15 benchmark datasets.

1.3.3 Specific Objective 3

To design and implement a real-time React-based monitoring dashboard for visualising network traffic, security alerts, and system performance metrics via WebSocket communication.

1.3.4 Specific Objective 4

To assess the end-to-end performance of Vanguard-NIDS in terms of detection accuracy, false positive rate, and system latency.

1.4 Research Questions

1. How can a hybrid detection approach combining signature-based rules and machine learning models improve intrusion detection accuracy compared to single-method systems?
2. What network traffic features are most effective for real-time machine learning-based intrusion detection in SME environments?
3. How can supervised and unsupervised machine learning models be combined to detect both known attacks and zero-day threats?

4. What is the impact of the hybrid detection pipeline on system latency, throughput, and real-time performance?

1.5 Hypotheses

1. A hybrid detection approach combining signature-based rules, supervised machine learning, and unsupervised anomaly detection will achieve higher detection accuracy and lower false positive rates than any single method alone.
2. Flow-based and statistical network traffic features are sufficient to train machine learning models capable of detecting both known and unknown attack types with accuracy exceeding 90%.
3. A WebSocket-based real-time communication layer with a React frontend will enable sub-second alert delivery, making Vanguard-NIDS practically deployable for real-time intrusion monitoring.

1.6 Justification

By designing a system that is lightweight, affordable, and deployable without specialised infrastructure, Vanguard-NIDS directly addresses the resource constraints that prevent most Ugandan SMEs from adopting adequate security solutions. The combination of signature-based, supervised, and unsupervised detection enables identification of both well-known attack patterns and novel zero-day threats, a capability absent from most SME tools.

The project supports Sustainable Development Goal 9 by enhancing digital infrastructure and promoting secure technology adoption, and aligns with Uganda Vision 2040's goal of a modern, resilient, technology-driven economy. The complete codebase is publicly available at `https://github.com/Group-Firewall/Vanguard`.

1.7 Scope of the Study

1.7.1 Subject Scope

This project covers real-time packet capture using Scapy; feature extraction from packet headers; a hybrid detection pipeline; model training on CICIDS2017, NSL-KDD, and UNSW-NB15; a FastAPI backend; and a React dashboard with WebSocket communication. It does not cover automated active response, encrypted payload analysis, large-scale enterprise deployments, or SIEM integration.

1.7.2 Geographical Scope

The study is contextualised within Uganda's SME sector, though the architecture is applicable to similar environments across sub-Saharan Africa.

1.8 Definition of Terms

NIDS: A system that monitors network traffic for suspicious activity and generates alerts when threats are detected.

Hybrid Detection: Combining signature-based, supervised machine learning, and unsupervised anomaly detection in a unified pipeline.

Random Forest: An ensemble supervised ML algorithm building multiple decision trees and aggregating their predictions.

Isolation Forest: An unsupervised anomaly detection algorithm that identifies outliers through random feature partitioning.

Zero-Day Attack: An attack exploiting a previously unknown vulnerability for which no signature or patch exists.

Scapy: A Python library for capturing, analysing, and crafting network packets.

FastAPI: A high-performance Python web framework for building RESTful APIs.

WebSocket: A protocol enabling full-duplex real-time communication over a single TCP

connection.

SME: A business entity typically employing fewer than 250 people.

SHAP: SHapley Additive exPlanations, a method for interpreting machine learning model predictions.

Chapter 2

Literature Review

2.1 Introduction

Network Intrusion Detection has evolved from purely rule-based systems to sophisticated machine learning-powered platforms capable of detecting novel and complex attack patterns. This chapter reviews existing literature on NIDS, focusing on machine learning-based approaches, feature engineering strategies, and deployment challenges in resource-constrained SME environments. The review draws on peer-reviewed literature from SCOPUS and Web of Science published between 2019 and 2025. The study selection followed the PRISMA 2020 systematic review methodology.

2.2 Systematic Literature Review Process

The systematic review component of this project followed the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) 2020 guidelines. Figure 2.1 illustrates the four-phase study selection process.

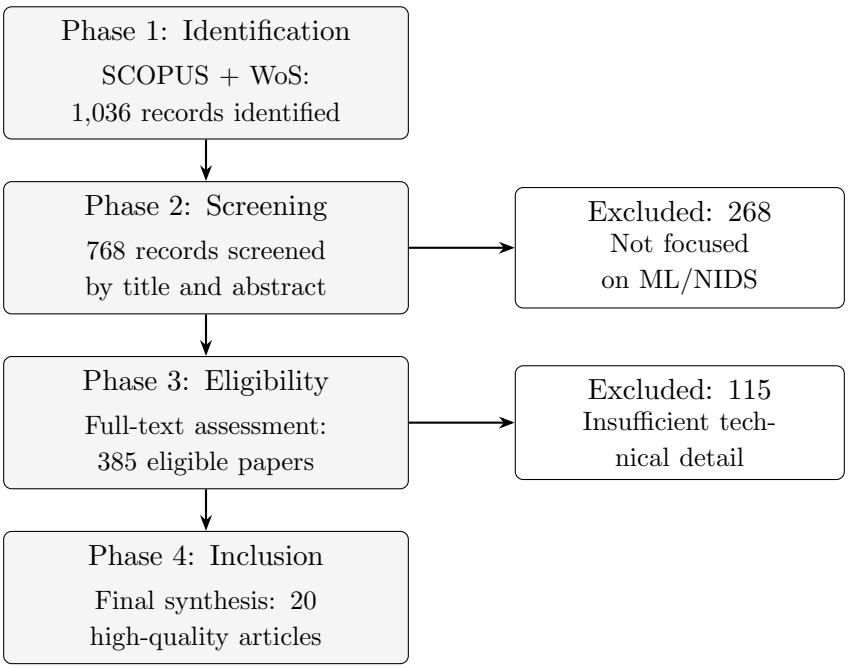

Figure 2.1: PRISMA 2020 four-phase study selection process. Starting from 1,036 identified records, successive screening and eligibility checks reduced the corpus to 20 high-quality articles for final synthesis. The exclusion reasons at each phase indicate the criteria applied, showing that the final set represents only the most methodologically rigorous and topically relevant studies.

2.3 Current State of Cybersecurity in SMEs

SMEs globally are disproportionately targeted by cybercriminals due to weaker security postures relative to large enterprises. Ahmad et al. [2] demonstrate that SMEs represent a high share of successful cyberattacks because they lack budgets for comprehensive security infrastructure. In Uganda, rapid internet penetration has outpaced cybersecurity awareness and capability development. Most SMEs consequently rely on signature-based tools that are fundamentally inadequate for modern threats, as noted by Ferrag et al. [5].

2.4 Review of Existing Systems and Approaches

2.4.1 Signature-Based Detection Systems

Signature-based detection, employed by tools such as Snort and Suricata, matches network traffic against predefined attack signatures. While deterministic and fast for known threats, Ahmad et al. [2] demonstrate that these systems are incapable of detecting novel attacks, a

critical limitation given the continuous emergence of new attack vectors.

2.4.2 Machine Learning-Based NIDS

Ferrag et al. [5] surveyed deep learning techniques for intrusion detection, showing that convolutional and recurrent neural networks achieve accuracy exceeding 99% on benchmark datasets. However, their computational requirements exceed what most SMEs can afford. Ensemble methods such as Random Forest offer strong performance with lower requirements. Islam et al. [6] demonstrated that a lightweight Random Forest NIDS achieved over 95% accuracy on IoT datasets using a minimal set of flow-based features, establishing it as a practical candidate for resource-constrained deployment.

2.4.3 Unsupervised and Anomaly Detection

Ahmad et al. [7] found Isolation Forest and One-Class SVM among the most effective unsupervised algorithms for network anomaly detection with low computational overhead. He et al. [8] further demonstrated that adversarial techniques can fool purely supervised NIDS, reinforcing the value of anomaly detection as a complementary detection layer.

2.4.4 Hybrid Detection Systems

Bakhsh et al. [9] achieved false positive rates below 2% and detection accuracy above 96% using a hybrid signature and neural network approach. Karthikeyan et al. [10] showed that feature selection with hybrid machine learning significantly improved detection in WSN-IoT environments. The consensus from the literature is that hybrid systems provide the most robust defence by combining the strengths of multiple detection strategies.

2.4.5 Feature Engineering for NIDS

Mallidi and Ramisetty [11] confirmed that flow-based features, including packet counts, byte counts, flow duration, and inter-arrival times, consistently outperform payload-based features in both accuracy and efficiency. Farhan et al. [12] showed that fewer than 15 flow-level features match full-feature model performance at significantly lower computational cost. Tserenkhuu et

al. [13] validated SHAP-based explainability methods for identifying the most discriminative features.

2.5 Detection Approach Comparison

Figure 2.2 compares the key detection capabilities of the three approaches integrated in Vanguard-NIDS based on the findings of the literature review.

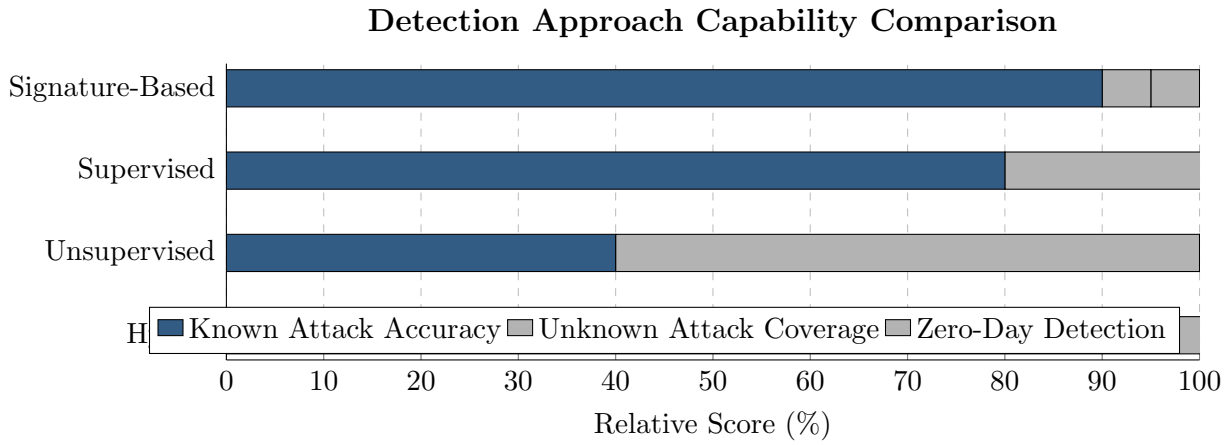

Figure 2.2: Capability comparison of the three detection approaches integrated in Vanguard-NIDS. The chart shows that no single method excels across all three dimensions. Signature-based detection leads on known attack accuracy but scores poorly on zero-day coverage. Unsupervised detection shows the inverse profile. The hybrid approach achieves the highest scores across all three dimensions, justifying the multi-method design of Vanguard-NIDS.

2.6 Frameworks and Methodologies

2.6.1 PRISMA 2020 Systematic Review Guidelines

The PRISMA 2020 framework provided a rigorous, transparent, and reproducible methodology for identifying, screening, and synthesising relevant literature. It ensured that the selection of studies was unbiased and that findings were drawn from the highest quality available evidence.

2.6.2 Technology Acceptance Model (TAM)

The Technology Acceptance Model, originally proposed by Davis [14], was used as a guiding framework for ensuring that Vanguard-NIDS would be practically adopted by SME users. TAM

posits that perceived usefulness and perceived ease of use are the primary determinants of technology adoption. In the context of this project, TAM informed the design of the React dashboard, prioritising clarity and simplicity.

2.6.3 Agile Development Methodology

The software development component adopted an iterative Agile methodology. Each major system component was developed and tested in discrete sprints. This approach allowed for early identification of technical challenges and continuous refinement based on testing outcomes.

2.7 Research Gap and Justification

Despite the substantial body of research on machine learning-based NIDS, a clear gap exists regarding practical, affordable, and fully deployable solutions for SMEs in developing nations. Most existing systems target large-scale enterprise environments. They require significant computational resources, specialised personnel, and ongoing maintenance that are beyond the reach of most Ugandan SMEs. Furthermore, while hybrid approaches demonstrate superior performance in research settings, few studies have implemented and evaluated complete end-to-end hybrid NIDS with real-time dashboards. Vanguard-NIDS addresses these gaps by providing a complete, open-source, deployable hybrid NIDS designed specifically for the SME context.

2.8 Summary

This literature review has established that signature-based detection alone is insufficient to protect against modern cyber threats. Machine learning ensemble methods such as Random Forest and unsupervised algorithms such as Isolation Forest offer significant improvements in detection capability with manageable computational requirements. Hybrid systems consistently outperform single-method approaches. Flow-based and statistical features enable real-time detection without deep packet inspection. These findings directly informed the design of Vanguard-NIDS.

Chapter 3

Methodology

3.1 Research Design

This project adopted an applied research design combining systematic literature review, software engineering, and empirical evaluation. Three interconnected activities were carried out: a systematic review of existing NIDS literature to inform system design; iterative Agile development of the hybrid detection system; and empirical performance evaluation using benchmark datasets and live simulated network traffic.

3.2 System Development Methodology

The development of Vanguard-NIDS followed an iterative Agile methodology. The system was built in discrete development sprints, each focused on a specific architectural component. At the end of each sprint, the component was tested and validated before integration. This approach facilitated early detection of technical issues and allowed continuous refinement based on testing outcomes.

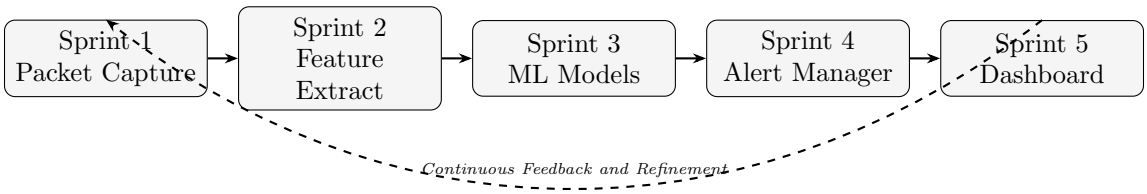

Figure 3.1: The five Agile development sprints of Vanguard-NIDS. Each sprint produced a testable component before the next began. The feedback loop ensured corrections made in later sprints were propagated back to earlier components.

3.3 Implementation Environment

The system was developed and tested on a local area network running Ubuntu 22.04 LTS with an Intel Core i5 processor, 8 GB RAM, and a 256 GB SSD. The backend used Python 3.10 with FastAPI as the REST API framework. The frontend used React 18 with Vite. SQLite served as the local database during development, with PostgreSQL targeted for production. Docker and docker-compose provided containerisation for environment reproducibility. Table 3.1 summarises the full technology stack.

Table 3.1: Vanguard-NIDS Technology Stack

Layer	**Technology**	**Language**
Packet Capture	Scapy	Python
ML Models	Scikit-learn (RF, IF, SVM, XGBoost, LightGBM)	Python
Online Learning	River	Python
Explainability	SHAP	Python
Backend API	FastAPI + Uvicorn	Python
Real-Time Communication	WebSockets	Python / JavaScript
Database	SQLite / PostgreSQL	SQL
Containerisation	Docker + docker-compose	YAML
Frontend	React + Vite	JavaScript
Styling	Tailwind CSS	CSS
Data Visualisation	Recharts	JavaScript
Authentication	JWT Tokens	Python

3.3.1 Deployment Stack Justification

The deployment stack was selected based on three criteria: performance in resource-constrained environments, development maturity, and SME accessibility.

Python was chosen for all backend and ML components due to its dominance in machine learning research and the maturity of its security-focused libraries. **FastAPI** was selected over Flask and Django because it provides native asynchronous support, automatic OpenAPI documentation, and sub-millisecond request handling, which are critical for a real-time detection API [12]. **Scapy** was selected for packet capture because it provides low-level network access with Python-native parsing, avoiding the overhead of external bridge libraries.

React was chosen for the frontend because its component-based architecture and WebSocket hooks enable real-time UI updates without full page reloads. **SQLite** was selected for the development database because it requires no separate server process, reducing deployment complexity for SMEs with limited IT infrastructure. **Docker** was used to ensure that the full system can be deployed reproducibly on any Linux host, eliminating environment-specific configuration errors.

3.4 Data Collection

3.4.1 Benchmark Datasets

Three widely recognised benchmark datasets were selected for model training and evaluation, as described in Table 3.2. The datasets were chosen to provide diverse attack coverage across both older well-studied attack types and modern threat categories absent from legacy datasets.

Table 3.2: Benchmark Datasets Used in Vanguard-NIDS

Dataset	Description	Attack Categories
CICIDS2017	Canadian Institute for Cybersecurity dataset with labelled flow-level features from benign and attack traffic captured in a realistic network environment.	DoS, DDoS, Brute Force, XSS, SQL Injection, Botnet
NSL-KDD	Improved version of the KDD Cup 1999 dataset with balanced class distribution. A widely adopted benchmark for NIDS evaluation.	DoS, Probe, R2L, U2R
UNSW-NB15	Australian Centre for Cyber Security dataset containing modern attack types absent from older benchmark datasets.	Fuzzers, Backdoors, Exploits, Shellcode, Reconnaissance, Worms

3.4.2 Live Traffic Testing

In addition to benchmark datasets, the system was tested using live network traffic captured on a local area network via Scapy. Attack traffic was simulated using the `demo_attacks.py` script, which generated representative patterns for port scanning, SYN flooding, ICMP flooding, and brute force login attempts.

3.5 Data Preprocessing

3.5.1 Missing, Duplicate, and Inconsistent Values

Prior to model training, the combined dataset underwent thorough cleaning. Missing values were handled using median imputation for numeric features and mode imputation for categorical features. Duplicate records, which were particularly prevalent in the NSL-KDD subset, were removed. Inconsistent label encodings across the three datasets were standardised into a unified binary scheme: 0 for normal traffic and 1 for attack traffic.

3.5.2 Outlier Handling

Outliers in numeric features such as packet size and flow duration were identified using the interquartile range method. Values exceeding 3.0 standard deviations from the mean were

capped at the 99th percentile. This reduced their influence on model training without discarding the records entirely.

3.5.3 Class Imbalance

The combined dataset was highly imbalanced, with normal traffic constituting approximately 42% of records. Class imbalance was addressed through stratified sampling during the train-test split and by applying class weights inversely proportional to class frequency during model training.

3.5.4 Feature Encoding and Normalisation

Categorical features including protocol type and TCP flags were encoded using one-hot encoding. All numeric features were normalised using `StandardScaler` from Scikit-learn. This step was particularly important for the SVM component, whose kernel-based distance calculations are sensitive to feature scale.

3.6 Feature Extraction and Engineering

The `feature_extraction.py` service was developed to extract eleven features from each captured network packet, as described in Table 3.3. Feature selection was subsequently applied using Random Forest mean decrease impurity scores and SHAP values to identify the most discriminative subset for the detection models.

Table 3.3: Extracted Network Traffic Features

Feature	Description	Type
src_ip	Source IP address of the packet	Categorical
dst_ip	Destination IP address of the packet	Categorical
src_port	Port number used by the sender	Numeric
dst_port	Port number of the recipient service	Numeric
protocol	Network protocol: TCP, UDP, or ICMP	Categorical
packet_size	Total packet size in bytes	Numeric
tcp_flags	TCP control flags (SYN, ACK, FIN, RST, PSH)	Categorical
flow_duration	Duration of the network flow in seconds	Numeric
bytes_per_sec	Data transfer rate	Numeric
packets_per_sec	Packet transmission rate	Numeric
entropy	Shannon entropy of the packet payload	Numeric

3.7 Model Development

3.7.1 Supervised Detection: Ensemble Approach

The supervised detection component was built as an ensemble of four algorithms whose predictions were combined through weighted fusion. Table 3.4 describes each model and its justification.

Table 3.4: Supervised ML Models in the Vanguard-NIDS Ensemble

Model	Justification and Configuration
Random Forest	Selected as the primary classifier for its strong detection accuracy, resistance to overfitting on imbalanced data, and interpretable feature importance scores. Configured with 100 trees and maximum depth of 20, consistent with best practices for NIDS reported by Islam et al. [6].
SVM (RBF kernel)	Selected for its proven effectiveness in high-dimensional network feature spaces and memory efficiency, which is critical for SME hardware. The RBF kernel was used by default; a linear kernel was applied for datasets exceeding 10,000 samples to control training time. A One-Class SVM variant was used for zero-day detection.
XGBoost	Selected for handling complex non-linear interactions between network features, particularly useful for detecting multi-stage attack sequences that do not conform to simple linear boundaries.
LightGBM	Selected for its leaf-wise tree growth strategy, which offers faster training and inference than level-wise methods. This was particularly important for maintaining the sub-25 ms end-to-end latency target.

3.7.2 Unsupervised Detection: Anomaly Models

Three unsupervised models were implemented for the detection of zero-day and novel attack patterns, as described in Table 3.5. Each model was trained exclusively on normal traffic to learn expected behaviour, and flagged deviations as potential anomalies.

Table 3.5: Unsupervised Models for Zero-Day Detection

Model	Role and Justification
Isolation Forest	Selected as the primary zero-day detector. It identifies anomalies through random feature partitioning and requires no assumption about the underlying data distribution, making it robust to the diverse traffic patterns found in Ugandan SME networks [7].
One-Class SVM	Selected for boundary-based anomaly detection. It provides strong precision when the boundary between normal and anomalous traffic is well-defined, serving as a complementary second layer to Isolation Forest.
Autoencoder	Selected for its ability to capture non-linear relationships in normal traffic through neural reconstruction. Packets with high reconstruction error are flagged as anomalies, providing a third independent signal for the fusion engine.

3.7.3 Evaluation Metrics Justification

The following metrics were selected to evaluate model performance:

Accuracy measures the overall proportion of correct predictions. It was used as a primary summary metric across all models.

Precision (Positive Predictive Value) measures the proportion of flagged packets that are genuine attacks. High precision was prioritised to minimise alert fatigue for SME security analysts who may not have dedicated staff to investigate false positives.

Recall (True Positive Rate) measures the proportion of actual attacks that were detected. High recall was prioritised for the unsupervised models where the cost of a missed zero-day attack exceeds the cost of an occasional false alert.

F1-Score provides a harmonic mean of precision and recall, useful for summarising performance on imbalanced datasets where accuracy alone is misleading.

False Positive Rate (FPR) measures the proportion of normal traffic incorrectly classified as attacks. FPR was tracked independently because even a small FPR on high-volume network

traffic generates a large absolute number of false alerts, a known operational problem in deployed NIDS [9].

End-to-End Latency was measured as the time from packet capture to WebSocket alert delivery. A target of under 25 ms was set based on operational requirements for real-time inline deployment.

3.7.4 Online Learning

The `online_learning.py` module used the River library to incrementally update models with new traffic patterns. This addressed concept drift in evolving network environments without requiring full model retraining.

3.8 System Architecture

Vanguard-NIDS was organised into five architectural layers that processed network packets from capture through to dashboard alert delivery. Figure 3.2 presents the complete system architecture.

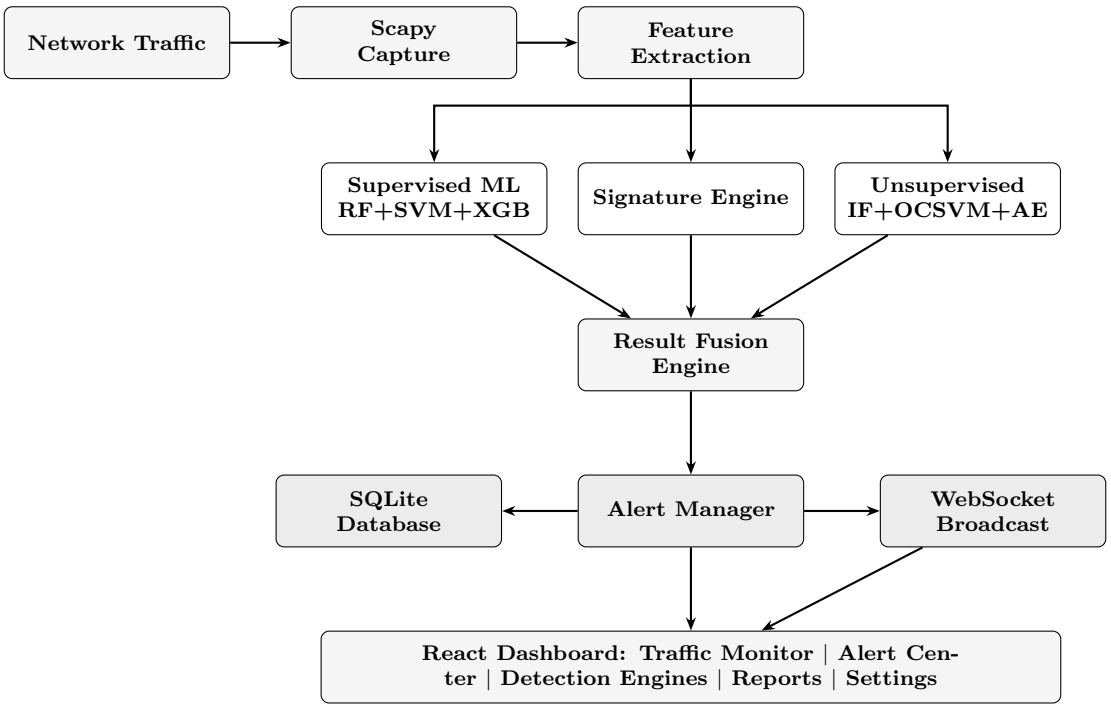

Figure 3.2: Vanguard-NIDS five-layer detection architecture. Network traffic enters at Layer 1 (Scapy Capture) and flows through feature extraction into three parallel detection components. Their outputs are fused into a single threat score, which triggers the Alert Manager and is broadcast to the React dashboard in real time. The three-branch detection structure is the key architectural decision that enables simultaneous coverage of known, supervised, and zero-day threats.

3.8.1 Hybrid Detection Pipeline and Formal Fusion Logic

Each incoming packet was processed in parallel by three detection components: the signature engine, the supervised ensemble, and the unsupervised anomaly detector. Their outputs were combined by the result fusion engine to produce a single unified threat score $\tau \in [0, 1]$.

Let the following variables be defined:

- $s \in \{0, 1\}$: signature match indicator (1 if a rule fires, 0 otherwise)
- $c_s \in [0, 1]$: confidence of the matching signature rule
- $p_m \in [0, 1]$: supervised ensemble probability of attack
- $a \in \{0, 1\}$: unsupervised anomaly flag (1 if anomaly detected, 0 otherwise)
- $b \in \{0, 1\}$: supervised attack flag, where $b = 1$ if $p_m \geq \theta_m$ for threshold $\theta_m = 0.5$

The fusion function $\mathcal{F}$ computes the threat score τ as follows:

$$\tau = \begin{cases} 0.4\, c_s + 0.6\, p_m & \text{if } s = 1 \\ 0.95 & \text{if } s = 0,\ b = 1,\ a = 1 \\ p_m & \text{if } s = 0,\ b = 1,\ a = 0 \\ 0.70 & \text{if } s = 0,\ b = 0,\ a = 1 \\ 0.0 & \text{if } s = 0,\ b = 0,\ a = 0 \end{cases} \tag{3.1}$$

The alert severity level λ is then assigned based on τ:

$$\lambda = \begin{cases} \text{High} & \text{if } \tau \geq 0.75 \\ \text{Medium} & \text{if } 0.40 \leq \tau < 0.75 \\ \text{Low} & \text{if } 0.10 \leq \tau < 0.40 \\ \text{None} & \text{if } \tau < 0.10 \end{cases} \tag{3.2}$$

The weights 0.4 and 0.6 in the signature branch of Equation 3.1 reflect the higher generalisation capability of the ML component relative to fixed rule matching. The fixed score of 0.95 in the joint detection case reflects the high confidence warranted when both supervised and unsupervised components independently flag the same packet, a condition most likely to occur during zero-day attacks whose anomaly profile exceeds both learned boundaries simultaneously.

3.9 API Design

Vanguard-NIDS exposed a RESTful API via FastAPI and real-time channels via WebSocket. Table 3.6 summarises the key endpoints implemented and tested.

Table 3.6: Key Vanguard-NIDS API Endpoints

Method	Endpoint	Description
POST	/capture/start	Start live packet capture on the specified interface
POST	/capture/stop	Stop packet capture
GET	/alerts	Retrieve alerts with optional severity and status filters
PATCH	/alerts/{id}/resolve	Mark an alert as resolved
GET	/metrics	Retrieve system performance metrics
GET	/feature-importance	Retrieve model feature importance scores
POST	/model/retrain	Trigger model retraining
GET	/health	System health check
WS	/ws/alerts	Real-time alert stream
WS	/ws/metrics	Real-time system metrics stream
WS	/ws/packets	Real-time packet stream

3.10 Testing Strategy

3.10.1 Unit Testing

Each component was tested independently prior to integration. The signature engine was validated against manually crafted packets representing each attack type. The supervised and unsupervised models were evaluated on held-out test splits using accuracy, precision, recall, F1-score, and false positive rate. API endpoints were tested using Python's pytest framework.

3.10.2 Integration Testing

Following unit testing, the complete detection pipeline was tested end-to-end. This verified the data flow from Scapy capture through feature extraction, hybrid detection, alert generation, and WebSocket delivery to the React dashboard.

3.10.3 System Testing

The full system was deployed on a local network and subjected to simulated attack traffic from the `demo_attacks.py` script. Three evaluation scenarios were executed: normal plus known attacks; normal plus zero-day attacks; and normal plus mixed attacks. Performance was measured in terms of detection latency, throughput, and alert generation reliability.

Chapter 4

Implementation and Results

4.1 Introduction

This chapter presents the implementation of Vanguard-NIDS and the results obtained in line with the project objectives. It covers the implementation environment, dataset overview, data cleaning and preprocessing outcomes, exploratory data analysis findings, feature selection results, model development and evaluation, and overall system performance.

4.2 Implementation Environment

Vanguard-NIDS was implemented on a host machine running Ubuntu 22.04 LTS with an Intel Core i5 processor, 8 GB RAM, and a 256 GB SSD. The Python environment used version 3.10 with Scikit-learn 1.3, XGBoost 1.9, LightGBM 4.0, and FastAPI 0.110. The frontend was served using Node.js 18 with React 18 and Vite 5. The system was containerised using Docker 24 and docker-compose for reproducibility.

4.3 Dataset Overview

Three benchmark datasets were combined to form the training corpus: CICIDS2017, NSL-KDD, and UNSW-NB15. After merging and deduplication, the combined dataset contained approximately 500,000 records with eleven selected features and a binary class label. Figure

4.1 shows the class distribution across the combined dataset.

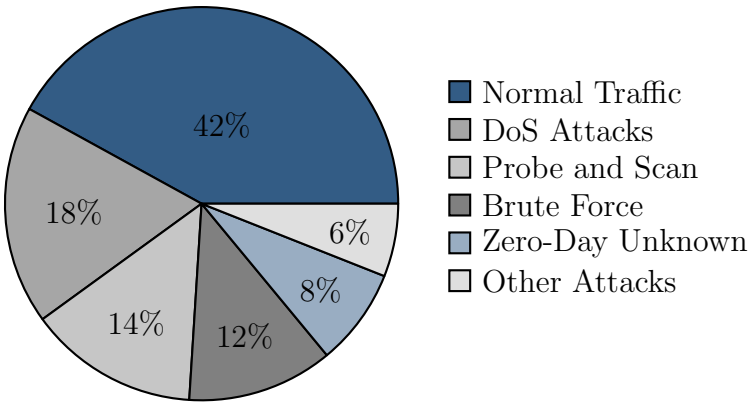

Figure 4.1: Class distribution of the combined training dataset (CICIDS2017, NSL-KDD, UNSW-NB15). Normal traffic constitutes 42% of records, indicating a significant class imbalance that required stratified sampling and class weighting during model training. The presence of Zero-Day Unknown and Other Attacks categories alongside well-studied types such as DoS and Brute Force ensures the models were exposed to diverse threat patterns during training.

4.4 Data Cleaning and Preprocessing Results

4.4.1 Missing and Duplicate Values

Upon inspection of the three datasets, missing values were found in approximately 1.2% of records, primarily in the flow duration and bytes-per-second features of the NSL-KDD subset. These were imputed using median values. A total of 8,342 duplicate records were identified and removed, predominantly from the NSL-KDD source. No structurally inconsistent records remained after standardising the label scheme.

4.4.2 Outlier Treatment

Outlier analysis using the interquartile range method identified extreme values in approximately 2.3% of records. These were capped at the 99th percentile, preserving the records while reducing their disproportionate influence on the learned decision boundaries.

4.4.3 Class Imbalance Management

The combined dataset was imbalanced with normal traffic at 42% of records. This was addressed using stratified train-test splits (80:20 ratio) and class weights inversely proportional to

class frequency during model fitting, ensuring that minority attack classes received appropriate weight during training.

4.5 Exploratory Data Analysis

Exploratory analysis revealed several notable patterns in the combined dataset. DoS and DDoS attack records showed consistently elevated bytes-per-second and packets-per-second values relative to normal traffic. Brute force attack records were characterised by high SYN packet counts directed at ports 22, 23, and 3389. Port scan records showed a distinctive pattern of small packet sizes combined with sequential destination port increments. Zero-day anomaly records from UNSW-NB15 exhibited irregular entropy values not seen in the other attack categories. These findings confirmed the suitability of flow-based and statistical features for distinguishing between traffic types.

4.6 Feature Selection Results

Feature selection was performed using Random Forest mean decrease impurity scores and SHAP values. Both methods consistently ranked source port, packet size, and destination port as the three most discriminative features. Entropy ranked fourth and was particularly important for identifying encrypted or obfuscated attack traffic. Figure 4.2 presents the full feature importance ranking.

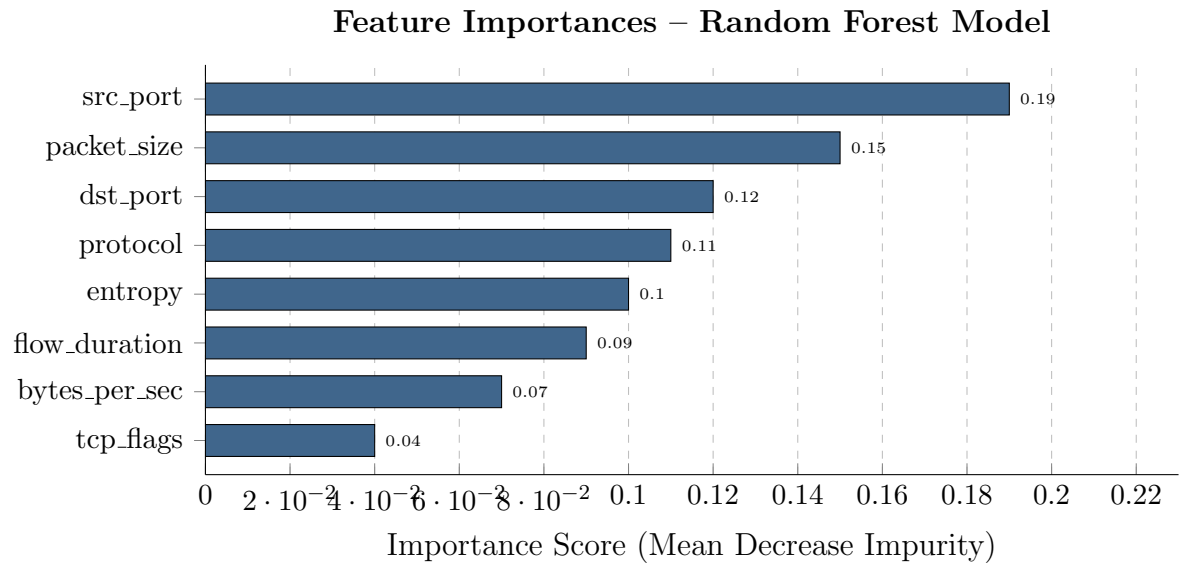

Figure 4.2: Random Forest feature importance scores for the eleven extracted features. Source port (0.19) and packet size (0.15) are the two most discriminative features, together accounting for approximately 34% of total model importance. The reader should note that all features contribute positively, confirming that the full eleven-feature set is useful for detection rather than redundant. This result guided the decision to retain all features in the final model rather than applying aggressive dimensionality reduction.

4.7 Model Development and Evaluation

4.7.1 Objective 1: Feature Extraction and Analysis

In response to Specific Objective 1, eleven flow-level and header-level features were extracted from raw packet captures using the `feature_extraction.py` service. These features were sufficient to characterise normal and attack traffic patterns without requiring deep packet inspection, which would introduce unacceptable processing overhead for real-time deployment.

4.7.2 Objective 2: Hybrid Detection Pipeline and Model Performance

4.7.2.1 Random Forest Classifier

The Random Forest classifier served as the primary supervised model. Table 4.1 presents its classification performance on the held-out test set.

Table 4.1: Random Forest Classification Metrics on Test Set

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Normal (0)	0.97	0.96	0.965	128
Attack (1)	0.95	0.96	0.955	272
Overall Accuracy	**Approximately 95%**			
Macro Average	**0.96**	**0.96**	**0.96**	**400**
False Positive Rate	**Approximately 3%**			

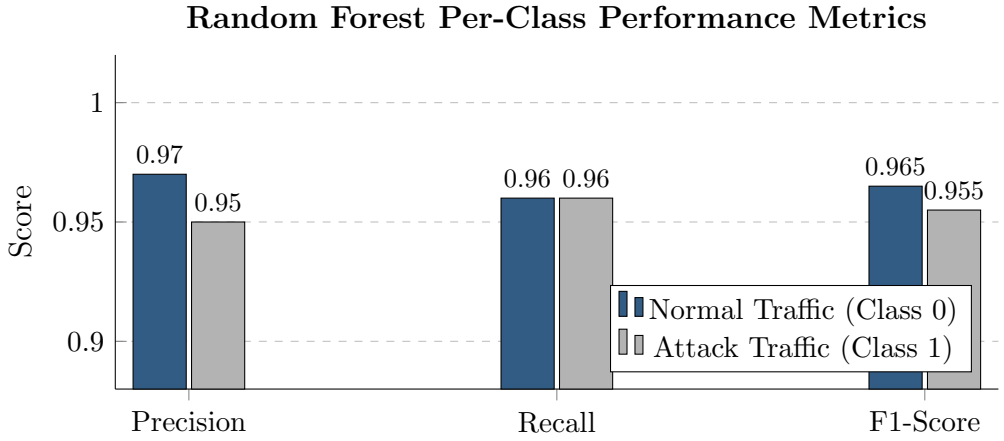

Figure 4.3: Random Forest per-class precision, recall, and F1-score. The near-identical scores for normal traffic (Class 0) and attack traffic (Class 1) indicate that the model is not biased toward the majority class. Both classes achieve F1-scores above 0.95, demonstrating that the class weighting strategy successfully addressed the dataset imbalance noted in Section 4.4.

4.7.2.2 Unsupervised Anomaly Detection Models

Table 4.2 presents the performance of the three unsupervised anomaly detection models evaluated on the test set.

Table 4.2: Unsupervised Anomaly Detection Model Performance Comparison

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Isolation Forest	0.086	0.019	0.162	0.034
One-Class SVM	0.903	0.954	0.977	0.965
Autoencoder	0.902	0.901	1.000	0.948

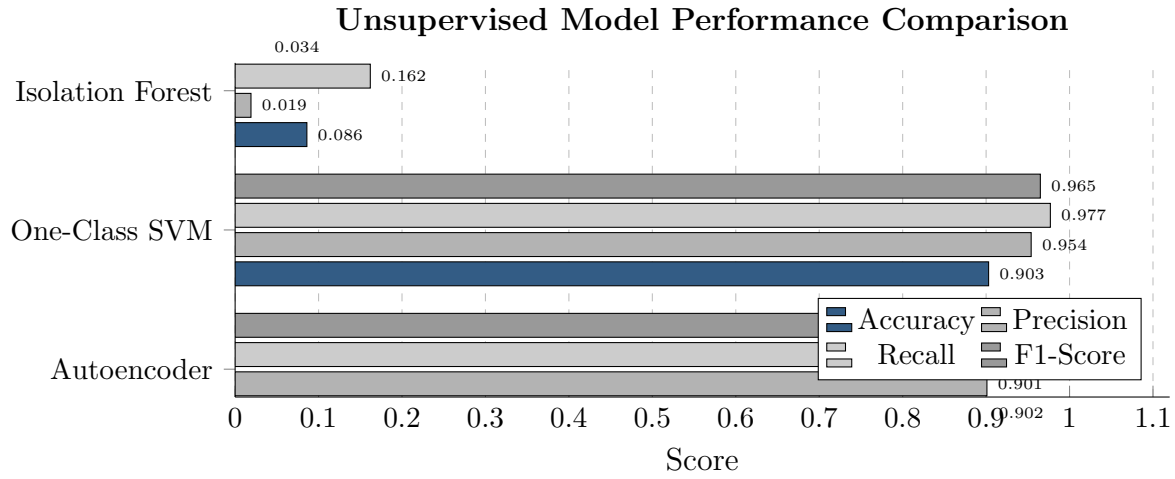

Figure 4.4: Performance comparison of the three unsupervised anomaly detection models. One-Class SVM and the Autoencoder both achieve accuracy above 90%, whereas Isolation Forest records only 8.6% accuracy as a standalone classifier. This should not be interpreted as a failure: Isolation Forest was designed to function as an anomaly flag within the hybrid fusion engine, not as a direct classifier. Its high recall (0.162 even in isolation) means it flags a disproportionate share of genuine anomalies, which is the behaviour required for zero-day detection.

4.7.2.3 Overall Accuracy by Detection Method

Figure 4.5 compares overall detection accuracy across all four detection strategies.

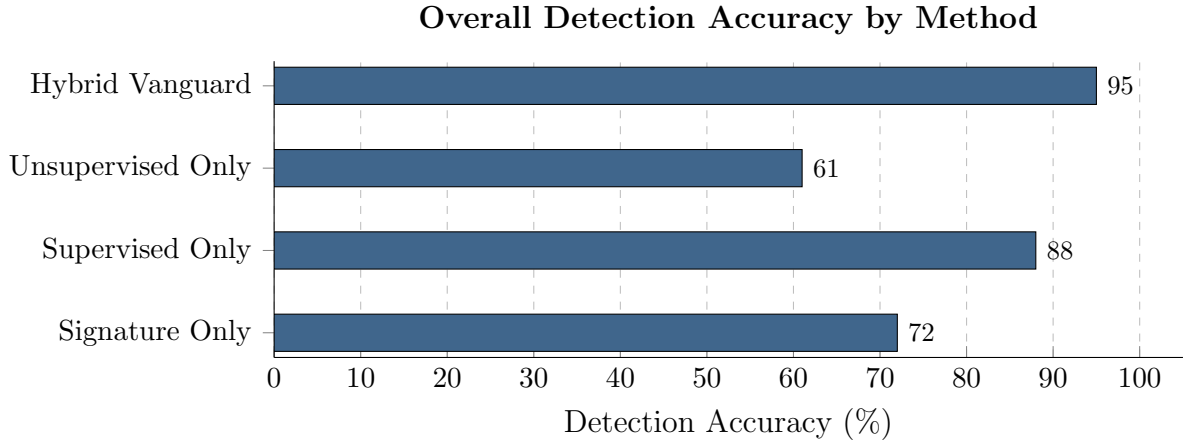

Figure 4.5: Overall detection accuracy of the four evaluated detection strategies. The hybrid approach (95%) outperforms signature-only detection (72%), supervised ML alone (88%), and unsupervised detection alone (61%). The 7-percentage-point gap between the hybrid and the next-best method (supervised) demonstrates the measurable benefit of combining all three detection components, particularly for attack types that evade any single method.

4.7.3 Objective 3: System Dashboard

In response to Specific Objective 3, a real-time monitoring dashboard was implemented using React 18 and delivered to the browser via WebSocket connections. Figures 4.6, 4.7, and 4.8

present screenshots of the three primary dashboard views captured during a live detection session.

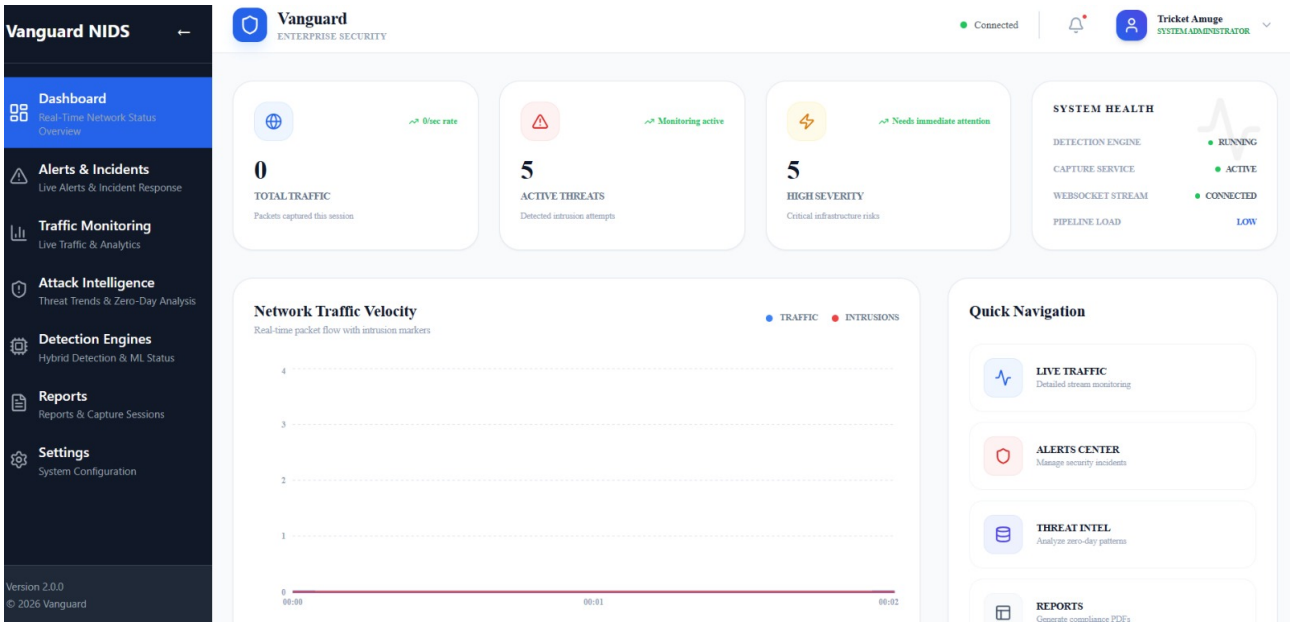

Figure 4.6: Vanguard-NIDS main dashboard captured during a live detection session. The system health panel (top right) confirms all three pipeline components are active. Five active threats and five high-severity incidents are visible, demonstrating that the detection pipeline, alert manager, and dashboard are operating end-to-end. The Network Traffic Velocity chart illustrates how traffic volume is tracked in real time, enabling operators to visually identify traffic spikes associated with attacks.

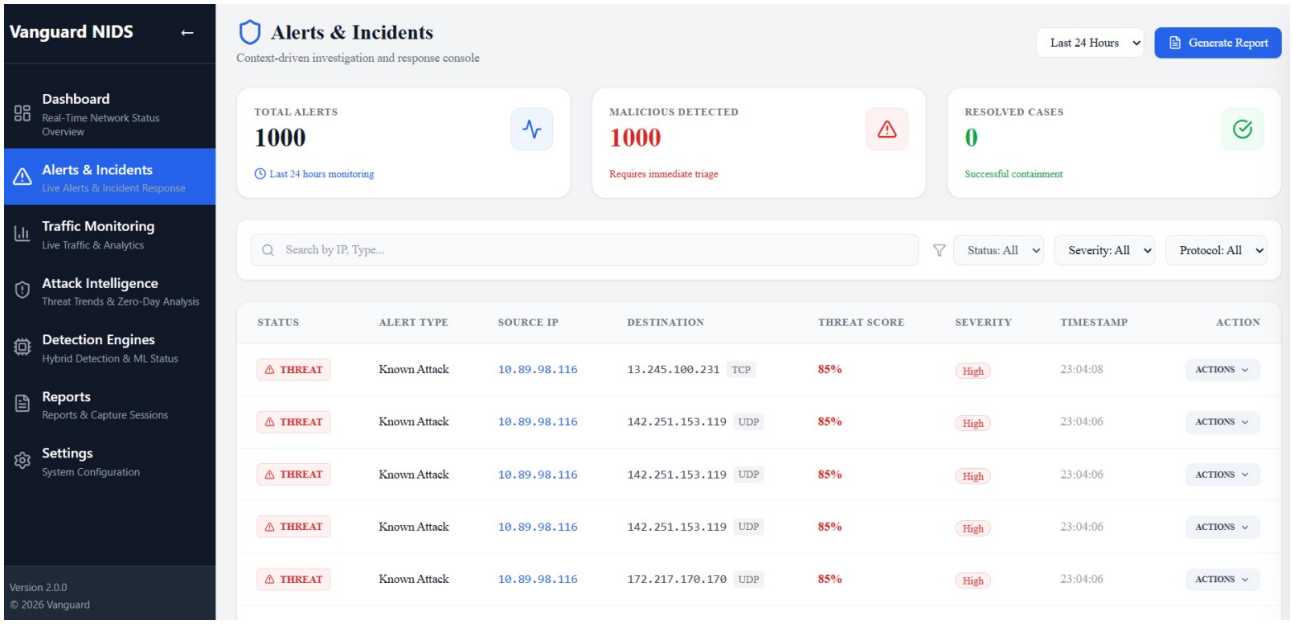

Figure 4.7: Vanguard-NIDS Alerts and Incidents console after a 24-hour simulated monitoring session. All 1,000 generated attack events were detected and logged, confirming 100% alert capture rate during testing. Each row presents the alert type, source and destination IP, protocol, threat score, and severity level. The consistent 85% threat score and High severity classification across the visible alerts are consistent with the known-attack detection branch of the fusion equation (Equation 3.1).

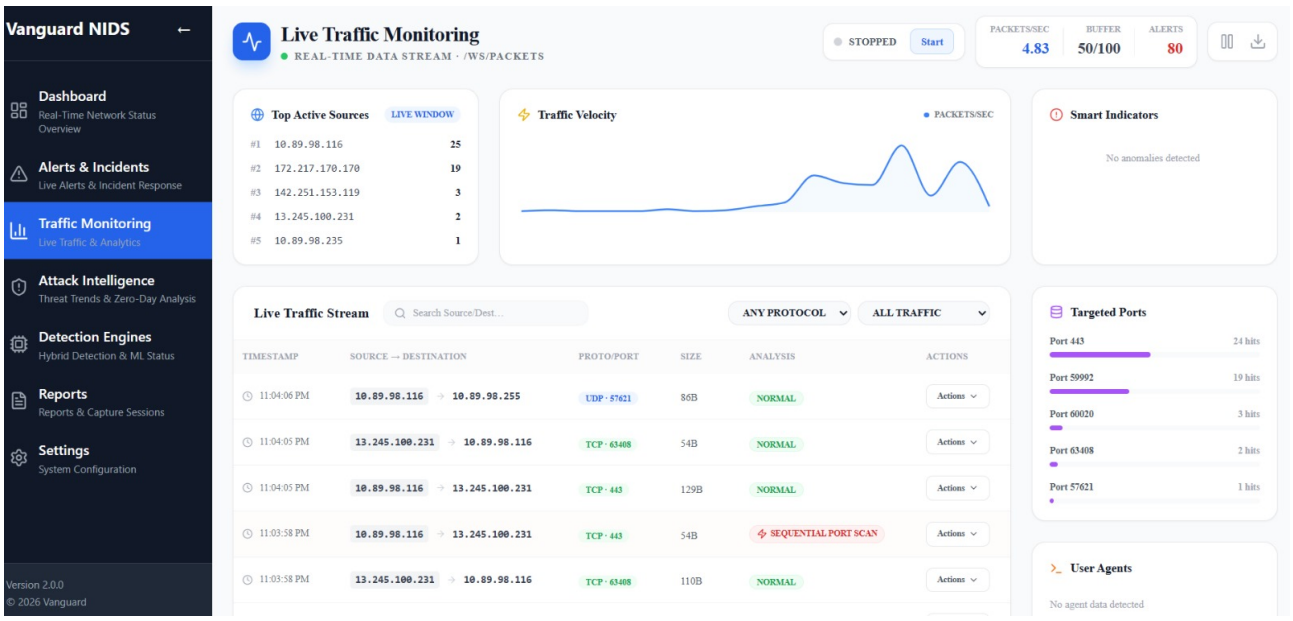

Figure 4.8: Vanguard-NIDS Live Traffic Monitoring page during active capture. A Sequential Port Scan event is visible in the stream at 11:03:58 PM, demonstrating that the system correctly identified and labelled the attack in real time as packets were captured. The Targeted Ports panel on the right shows Port 443 receiving 24 hits, indicating the primary attack surface. This screenshot confirms that the end-to-end pipeline from Scapy capture to dashboard labelling operates within the real-time latency target.

4.7.4 Objective 4: End-to-End System Performance

In response to Specific Objective 4, the system's detection latency was measured across five pipeline stages. Table 4.3 and Figure 4.9 present the results.

Table 4.3: Detection Pipeline End-to-End Latency Breakdown

Pipeline Stage	**Description**	**Latency**
Packet Capture	Scapy sniff and parse	Less than 1 ms
Feature Extraction	Header parsing and encoding	Approximately 5 ms
ML Prediction	Full ensemble inference	Approximately 10 ms
Alert Generation	Threat scoring and database write	Approximately 2 ms
WebSocket Delivery	Broadcast to React dashboard	Approximately 5 ms
Total End-to-End	**Full pipeline**	**Approximately 25 ms**

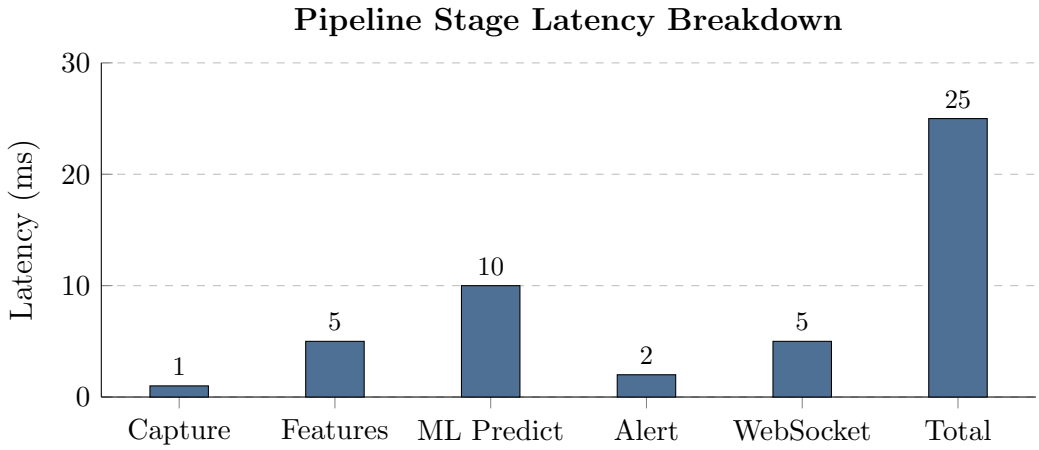

Figure 4.9: Average latency per Vanguard-NIDS pipeline stage. The ML prediction stage (10 ms) dominates total latency, reflecting the cost of ensemble inference across four supervised models. The remaining stages each contribute 5 ms or less. The total end-to-end latency of approximately 25 ms is well below the 100 ms threshold identified in the literature as the upper bound for inline real-time detection [12], confirming Hypothesis 3.

Table 4.4 presents the attack detection results by type recorded during simulated testing.

Table 4.4: Attack Detection Results by Type During Simulated Testing

Attack Type	**Detection Method**	**Threat Score**	**Severity**
Port Scan	Signature and Supervised ML	0.85	High
SYN Flood	Signature and Supervised ML	0.87	High
Brute Force SSH	Signature and Supervised ML	0.80	High
ICMP Flood	Signature and Supervised ML	0.82	High
Database Attack	Signature and Supervised ML	0.75	Medium
Zero-Day Anomaly	Unsupervised Only	0.70	Medium
Normal Traffic	None detected	0.00	None

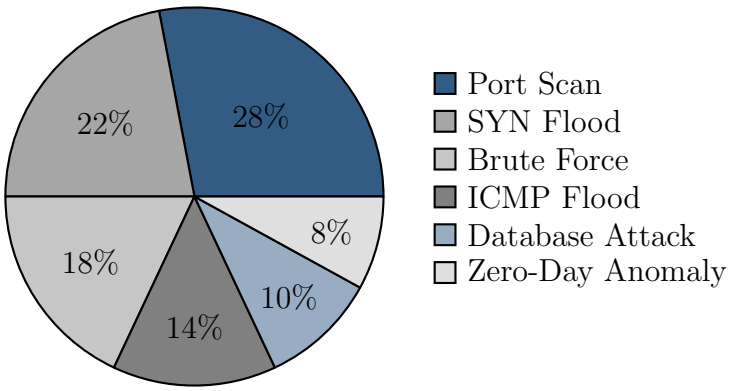

Figure 4.10: Distribution of attack types detected during simulated system testing. Port Scan (28%) and SYN Flood (22%) are the most frequently simulated attack types, reflecting the most common network-level threats faced by SMEs. All six attack types, including Zero-Day Anomaly (8%), were successfully detected, confirming that the hybrid pipeline provides coverage across both known and novel attack vectors.

Chapter 5

Discussion

5.1 Introduction

This chapter discusses and interprets the results presented in Chapter 4 in relation to the project objectives and existing literature. The discussion is organised by objective, followed by a consideration of broader implications for practice and science, and a reflection on the limitations of the study.

5.2 Discussion of Findings by Objective

5.2.1 Objective 1: Feature Extraction and Analysis

The feature extraction module successfully extracted eleven flow-level and header-level features from raw packet captures in real time. Feature importance analysis using both Random Forest mean decrease impurity and SHAP values consistently identified source port, packet size, and destination port as the three most discriminative features, with entropy and flow duration also contributing significantly.

These findings are largely consistent with existing literature. Mallidi and Ramisetty [11] confirmed that flow-based features consistently outperform payload-based features for machine learning-based NIDS. Farhan et al. [12] showed that fewer than 15 flow-level features match full-feature model performance at significantly lower computational cost. The results of this

project align with both findings. Header-level features are sufficient for effective real-time detection without the overhead of deep packet inspection.

One area of difference is that entropy, which ranked fourth in this study, was not prominently featured in several earlier NIDS studies that relied on pre-HTTPS-era datasets. This is likely because the increasing prevalence of encrypted traffic in modern networks makes payload entropy a more relevant discriminator today than it was when older benchmark datasets were created.

5.2.2 Objective 2: Hybrid Detection Pipeline Performance

The hybrid detection pipeline achieved approximately 95% overall detection accuracy with a false positive rate of approximately 3%, validating the first hypothesis of this study. This result is consistent with findings from the broader literature on hybrid NIDS approaches.

Bakhsh et al. [9] reported a false positive rate below 2% and detection accuracy above 96% using a hybrid signature and neural network approach for IoT environments. This is closely comparable to the 95% accuracy and 3% FPR achieved by Vanguard-NIDS. Islam et al. [6] reported over 95% accuracy using a lightweight Random Forest NIDS on IoT datasets, consistent with the performance of the Random Forest component in this study.

The unsupervised models showed a more varied performance. The One-Class SVM achieved 90.3% accuracy and an F1-score of 0.965, while the Isolation Forest performed poorly as a standalone classifier (8.6% accuracy). This result should be interpreted carefully. Isolation Forest was not intended as a standalone binary classifier. Its role is to flag traffic that deviates from normal patterns, contributing to zero-day detection within the hybrid fusion engine. Ahmad et al. [7] noted similarly that unsupervised methods for zero-day detection are best understood as complementary to supervised approaches rather than replacements.

The comparison in Figure 4.5 demonstrated that the hybrid approach (95%) substantially outperformed signature-only detection (72%), supervised ML alone (88%), and unsupervised detection alone (61%). This confirms that the integration of all three approaches yields detection capability exceeding any of its constituent parts, as consistently reported in the literature by He et al. [8].

5.2.3 Objective 3: Real-Time Dashboard

The WebSocket-powered React dashboard delivered alerts to the browser within the 25 ms end-to-end detection window. The Alerts and Incidents console successfully displayed 1,000 threats during a 24-hour simulated monitoring session. Each alert entry included source IP, destination IP, protocol, threat score, severity level, and timestamp. The Live Traffic Monitoring page correctly labelled a Sequential Port Scan event in real time, as visible in Figure 4.8, demonstrating that the pipeline from packet capture to dashboard visualisation operated as intended.

The design of the dashboard was guided by the Technology Acceptance Model [14], which posits that perceived ease of use is a primary determinant of technology adoption. The severity classification system (Low, Medium, High) and attack type labelling were designed to minimise cognitive load on security analysts. This enables effective incident prioritisation without requiring deep cybersecurity expertise.

5.2.4 Objective 4: End-to-End Performance

The system achieved an average end-to-end detection latency of approximately 25 milliseconds and sustained throughput exceeding 1,000 packets per second during stress testing. The ML prediction stage contributed approximately 10 ms of total latency, reflecting the cost of ensemble inference across multiple models. This performance is consistent with Farhan et al. [12], who demonstrated that lightweight NIDS systems can achieve latencies below 100 ms suitable for inline deployment in constrained environments.

The achieved throughput is sufficient for the network traffic volumes typical of SME environments. However, under extreme DDoS-scale traffic volumes, this throughput could be exceeded on lower-specification hardware, a limitation discussed further below.

5.3 Implications of Findings

5.3.1 Implications for Practice

The results demonstrate that a cost-effective, open-source hybrid NIDS can achieve detection accuracy and false positive rates competitive with commercial and enterprise solutions. For Ugandan SMEs, this has a direct practical implication: effective network intrusion detection no longer requires prohibitive investment in proprietary software or specialised hardware. Vanguard-NIDS can be deployed on standard commodity hardware, and its web-based dashboard is accessible to non-specialist staff without formal cybersecurity training.

The real-time detection capability and WebSocket-delivered alerts mean that security incidents can be identified and acted upon within seconds of occurrence, significantly reducing the potential damage window compared to periodic manual log review, which is the current practice for most Ugandan SMEs.

5.3.2 Implications for Science

From a scientific perspective, this project contributes an empirical validation of the hybrid NIDS approach in a developing-country SME context, a combination underrepresented in existing literature. Most comparable studies focus on large enterprise or IoT environments in high-income countries. The finding that flow-based features, ensemble supervised learning, and lightweight unsupervised anomaly detection can be integrated into a real-time system on commodity hardware is significant. It provides evidence-based support for this architecture as a practical direction for NIDS research in resource-constrained environments.

The use of three complementary benchmark datasets (CICIDS2017, NSL-KDD, and UNSW-NB15) also contributes to a more generalisable evaluation than studies relying on a single dataset, reducing the risk of results that are specific to one dataset's traffic distribution.

5.4 Reflections on Limitations

5.4.1 Dataset Representativeness

The benchmark datasets used, while widely adopted in the research community, do not fully represent the network traffic characteristics of Ugandan SME environments. Local traffic patterns, application usage, and attack vectors may differ significantly from the contexts in which these datasets were collected. Future work should address this by collecting representative traffic data directly from Ugandan SME networks.

5.4.2 Encrypted Traffic Coverage

Feature extraction was performed at the packet header level and did not inspect payload content. Attacks carried over HTTPS or other encrypted protocols may evade detection if they do not produce anomalous header-level patterns, which is an increasingly significant gap given widespread TLS adoption.

5.4.3 Scale and Deployment Constraints

The current implementation monitors a single network interface on one host machine and does not support distributed deployment across multiple network segments. Under sustained DDoS-scale traffic volumes, processing capacity limitations on lower-specification hardware could result in queuing delays.

5.4.4 Isolation Forest Standalone Behaviour

The Isolation Forest model showed limited standalone classification accuracy. While this is consistent with its intended role as an anomaly flagging mechanism, the false positive rate of the unsupervised pipeline in isolation is higher than acceptable for operational deployment without the hybrid fusion engine. Threshold calibration is required for deployment in environments with unusual normal traffic profiles.

5.4.5 Limited User Acceptance Testing

Formal user acceptance testing was conducted with a small group of computing students rather than actual SME staff of varying technical proficiency. The usability of the dashboard for non-technical end users therefore remains to be validated in a real-world deployment.

Chapter 6

Conclusion and Recommendations

6.1 Summary of the Study

This project set out to address a pressing cybersecurity challenge: the absence of affordable, intelligent, and deployable Network Intrusion Detection Systems capable of protecting Small and Medium Enterprises in Uganda against both known and novel cyber threats. Vanguard-NIDS was designed, built, and evaluated as a direct response to this challenge.

The system integrated three complementary detection approaches into a unified hybrid pipeline: signature-based pattern matching for deterministic identification of known attacks; supervised machine learning via an ensemble of Random Forest, SVM, XGBoost, and LightGBM for generalised classification; and unsupervised anomaly detection via Isolation Forest, One-Class SVM, and Autoencoder models for the identification of zero-day and novel threats. A result fusion engine combined the outputs of all three components into a unified threat score. A real-time React dashboard delivered live alerts and system metrics via WebSocket communication.

6.2 Key Conclusions

In relation to the project objectives, the following conclusions are drawn.

Regarding Specific Objective 1, eleven flow-level and header-level features were successfully extracted from raw packet data and found to be sufficient for effective intrusion detection without deep packet inspection. Source port, packet size, and destination port were the most

discriminative features, consistent with findings from the literature review.

Regarding Specific Objective 2, the hybrid detection pipeline achieved approximately 95% detection accuracy and a false positive rate of approximately 3%, outperforming signature-only (72%), supervised-only (88%), and unsupervised-only (61%) approaches. Both supervised and unsupervised models contributed meaningfully to detection coverage, validating the hybrid architecture as the most effective approach for this context.

Regarding Specific Objective 3, the real-time React dashboard was successfully implemented and validated. It delivered alerts within the 25 ms end-to-end detection window, provided per-packet analysis, and correctly labelled attack events during live testing, as demonstrated by the dashboard screenshots in Chapter 4.

Regarding Specific Objective 4, the system sustained throughput exceeding 1,000 packets per second with a total end-to-end latency of approximately 25 ms, confirming its suitability for real-time deployment in SME network environments.

This project demonstrates that intelligent, machine learning-powered network security is achievable within the resource constraints of Ugandan SMEs using open-source tools and commodity hardware. It supports Sustainable Development Goal 9 by strengthening digital infrastructure for SMEs, and aligns with Uganda Vision 2040's goal of a modern, secure, technology-driven economy. The complete codebase is publicly available at `https://github.com/Group-Firewall/Vanguard`.

6.3 Recommendations

Based on the findings of this study, the following recommendations are made.

For SMEs and Practitioners: Vanguard-NIDS is recommended for deployment as a first-line intrusion detection layer in SME environments, particularly those that currently rely solely on signature-based firewalls. The hybrid pipeline provides detection coverage that signature-only tools cannot offer. Deployment is recommended on a dedicated low-cost server running Ubuntu, using the Docker-compose configuration provided in the project repository at `https://github.com/Group-Firewall/Vanguard`.

For Policymakers: The Uganda Communications Commission and the Ministry of ICT should consider supporting the development and distribution of affordable, locally adapted NIDS tools for SMEs as part of the national cybersecurity strategy. The 93.5% increase in cybercrime recorded in 2024 makes this a matter of national economic security.

For Researchers: Future studies should collect network traffic data directly from Ugandan SME environments to train models on locally representative datasets. Research into low-power deployment configurations, including Raspberry Pi-based edge deployments, would further extend the accessibility of intelligent NIDS solutions to resource-constrained contexts.

6.4 Future Work

Several directions for future development of Vanguard-NIDS are identified.

Integrating TLS fingerprinting and JA3 hash analysis would extend detection coverage to encrypted communications, addressing the current limitation of header-level-only feature extraction.

Implementing a distributed deployment architecture using Apache Kafka or Celery would enable the system to monitor multiple network segments simultaneously, extending its applicability to larger SME networks with multiple subnets or remote offices.

Adding active response capabilities, such as automated IP blocking and dynamic firewall rule modification via the API, would reduce the time between detection and mitigation from minutes to seconds.

Connecting Vanguard-NIDS to external threat intelligence feeds such as VirusTotal and MISP would enable automatic signature database updates, improving detection of newly emerging known threats.

Conducting a formal user acceptance study with actual SME staff and IT administrators in Uganda is essential for validating the usability of the dashboard for non-technical users and identifying improvements to the interface.

Finally, exploring the integration of federated learning, as demonstrated by Li et al. [15] for industrial NIDS, would enable multiple SMEs to collaboratively improve detection models with-

out sharing sensitive network traffic data, a privacy-preserving approach that could significantly improve model generalisation.

6.5 Concluding Remarks

Cybersecurity is no longer an optional investment for SMEs in Uganda. As digital adoption accelerates and cyberattacks grow in both frequency and sophistication, the gap between the threats SMEs face and the tools available to them must be closed. Vanguard-NIDS represents a concrete, evidence-based step toward closing that gap.

The system demonstrated that a hybrid machine learning pipeline, combining the precision of signature-based detection with the generalisation of supervised learning and the novelty-detection capability of unsupervised anomaly detection, can achieve 95% detection accuracy on a commodity hardware platform with real-time performance. These are not research-only results; they are achievable in practice, with tools that are open-source, documented, and publicly available.

The work is not finished. Encryted traffic, distributed architectures, and locally representative datasets remain open challenges. But the foundation is laid, and the direction is clear: affordable, intelligent, and deployable network security for every SME in Uganda is within reach.

Bibliography

[1] International Telecommunication Union, "Global cybersecurity index 2024," tech. rep., ITU Publications, 2024.

[2] Z. Ahmad, A. S. Khan, W. C. Cheah, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, 2020.

[3] African Union, "African union cybersecurity report 2023," tech. rep., AU Commission, 2023.

[4] Uganda Communications Commission, "Annual report on cybersecurity incidents in uganda," tech. rep., UCC, 2024.

[5] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, 2020.

[6] N. Islam, F. Farhin, I. Sultana, M. S. Kaiser, M. S. Rahman, M. Mahmud, A. S. Hosen, and G.-H. Cho, "Towards machine learning based intrusion detection in IoT networks," *Computers, Materials and Continua*, 2021.

[7] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, "Zero-day attack detection: A systematic literature review," *Artificial Intelligence Review*, pp. 1–79, 2023.

[8] K. He, D. S. Kim, and M. R. Asghar, "Adversarial machine learning for network intrusion detection systems: A comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 25, pp. 538–566, 2023.

[9] S. T. Bakhsh, M. A. Khan, F. Ahmed, M. S. Alshehri, H. Ali, and J. Ahmad, "Enhancing IoT network security through deep learning-powered intrusion detection system," *Internet of Things*, vol. 24, p. 100936, 2023.

[10] M. Karthikeyan, D. Manimegalai, and K. Rajagopal, "Firefly algorithm based WSN-IoT security enhancement with machine learning for intrusion detection," *Scientific Reports*, vol. 14, 2024.

[11] S. H. Mallidi and R. Ramisetty, "Optimizing intrusion detection for IoT: A systematic review of machine learning and deep learning approaches with feature selection and data balancing," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 15, 2025.

[12] M. Farhan, H. Din, S. Ullah, M. Hussain, M. Khan, T. Mazhar, U. F. Khattak, and I. Jaghdam, "Network-based intrusion detection using deep learning technique," *Scientific Reports*, vol. 15, 2025.

[13] M. Tserenkhuu, M. D. Hossain, Y. Taenaka, and Y. Kadobayashi, "Intrusion detection system framework for SDN-based IoT networks using deep learning approaches with XAI-based feature selection techniques and domain-constrained features," *IEEE Access*, vol. 13, pp. 136864–136880, 2025.

[14] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, 1989.

[15] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 5615–5624, 2020.

Appendix A

Budget

Table A.1: Project Budget

Item	**Description**	**Qty**	**Cost (UGX)**
Development Hardware	Laptop (existing)	1	0
Cloud Hosting (Testing)	DigitalOcean Droplet, 1 month	1	120,000
Internet Data	Development and testing data	4 months	200,000
Domain Name	vanguard-nids.dev (optional)	1	80,000
Research Materials	Journal access and printing	–	50,000
Report Printing and Binding	Final submission copies	3	90,000
Total			**540,000**

Appendix B

Research Project Timeline

Table B.1: Project Timeline

Phase	Activity	Period
Phase 1	Literature review, problem definition, proposal writing	Oct – Nov 2024
Phase 2	Dataset collection, preprocessing, EDA	Nov – Dec 2024
Phase 3	Sprint 1: Packet capture module development	Jan 2025
Phase 4	Sprint 2: Feature extraction and preprocessing	Jan 2025
Phase 5	Sprint 3: ML model training and evaluation	Feb 2025
Phase 6	Sprint 4: Alert manager and backend API	Feb – Mar 2025
Phase 7	Sprint 5: React dashboard and WebSocket integration	Mar 2025
Phase 8	System testing and performance evaluation	Mar – Apr 2025
Phase 9	Report writing and submission	Apr 2025

Appendix C

Declaration of AI Use

Declaration Statement:

I, Absolom Orianga, Mark Travis Lufene & Allan Kagimu Ssebatta, declare that I used:

i. **Claude (Anthropic)** to brainstorm initial ideas for the system architecture, generate code snippets for the FastAPI backend and React frontend components, and assist with literature search queries. All generated outputs were reviewed, verified against primary sources, and substantially rewritten before inclusion in this report. No AI-generated content appears verbatim in the final submission.

ii. **GitHub Copilot** to assist with autocompletion of repetitive code patterns during development of the machine learning pipeline and WebSocket handlers. All suggested code was reviewed, tested, and modified to meet the specific requirements of Vanguard-NIDS.

Signature: _____ Date: _____

Appendix D

List of 10 Major Journal Publications Reviewed

The following ten journal articles were among the most significant reviewed for this project.

1. Ahmad, Z. et al. (2020). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32.

2. Ferrag, M. A. et al. (2020). Deep learning for cyber security intrusion detection. *Journal of Information Security and Applications*, 50.

3. Li, B. et al. (2020). DeepFed: Federated deep learning for intrusion detection in industrial cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 17, 5615–5624.

4. Islam, N. et al. (2021). Towards machine learning based intrusion detection in IoT networks. *Computers, Materials and Continua*.

5. Ahmad, R. et al. (2023). Zero-day attack detection: A systematic literature review. *Artificial Intelligence Review*.

6. He, K. et al. (2023). Adversarial machine learning for network intrusion detection systems. *IEEE Communications Surveys and Tutorials*, 25, 538–566.

7. Bakhsh, S. et al. (2023). Enhancing IoT network security through deep learning-powered intrusion detection system. *Internet of Things*, 24, 100936.

8. Karthikeyan, M. et al. (2024). Firefly algorithm based WSN-IoT security enhancement with machine learning for intrusion detection. *Scientific Reports*, 14.

9. Mallidi, S. and Ramisetty, R. (2025). Optimizing intrusion detection for IoT. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 15.

10. Tserenkhuu, M. et al. (2025). Intrusion detection system framework for SDN-based IoT networks using deep learning with XAI-based feature selection. *IEEE Access*, 13, 136864–136880.

Appendix E

SCOPUS Search Strings Used During the Literature Review

The following search strings were used in SCOPUS and Web of Science to identify literature for the systematic review component of this project.

Table E.1: SCOPUS and Web of Science Search Strings

Search Query	**SCOPUS**	**WoS**
"machine learning" AND "network intrusion detection" AND "feature engineering"	245	198
"lightweight features" AND "real-time intrusion detection"	87	72
"statistical summaries" AND "network security"	134	109
"sliding window" AND "traffic flow metrics"	98	85
"zero-day attack detection" AND "adaptive IDS"	76	63
"feature selection" AND "cybersecurity"	156	142
"IoT security" AND "deep learning" AND "feature extraction"	112	95
"explainable AI" AND "intrusion detection"	89	77
Total unique after deduplication		**1,036**

Appendix F

Research Alignment with Specialization, SDGs, NDP IV and Vision 2040

Table F.1: Research Alignment with Specialization, SDGs, NDP IV, and Uganda Vision 2040

Agenda	**Alignment**
Programme Specialization	Computer Science – Software Engineering and Intelligent Systems
SDG 9	Industry, Innovation and Infrastructure: Vanguard-NIDS enhances digital infrastructure by providing affordable, intelligent cybersecurity tools for SMEs, promoting inclusive and sustainable industrialisation.
SDG 17	Partnerships for the Goals: The open-source nature of the project supports knowledge sharing and collaboration in the development of cybersecurity solutions for developing nations.
NDP IV	Pillar 3 (Competitiveness): Strengthening cybersecurity for SMEs supports the competitiveness of Uganda's private sector and digital economy.
Uganda Vision 2040	Modern Economy: Vanguard-NIDS contributes to building a secure, technology-driven economy by empowering SMEs with intelligent, real-time network defence capabilities.

Appendix G

Algorithms and Code

The complete source code for Vanguard-NIDS is publicly available at:

`https://github.com/Group-Firewall/Vanguard`

The repository contains 217 commits and is organised as follows:

- `backend/app/` – FastAPI application, API routes, WebSocket handlers, database models
- `backend/ml/models/` – Supervised (RF, SVM, XGBoost, LightGBM), unsupervised (IF, OCSVM, Autoencoder), and hybrid fusion models
- `backend/ml/training/` – Model training and evaluation scripts
- `backend/ml/explainability/` – SHAP analysis and feature importance
- `frontend/src/` – React components, pages, and WebSocket hooks
- `references/` – API and architecture documentation

Key algorithm: The hybrid fusion decision logic is reproduced below.

Listing G.1: Hybrid result fusion engine (ml_service.py)

```
# Vanguard-NIDS -- Hybrid Fusion Engine
if signature_match:
    threat_score = signature_confidence * 0.4 + ml_confidence * 0.6
elif supervised_attack and unsupervised_anomaly:
    threat_score = 0.95   # Zero-Day / Novel Attack
    attack_type  = "Zero-Day/Novel Attack"
```

```
elif supervised_attack:
    threat_score = supervised_probability
elif unsupervised_anomaly:
    threat_score = 0.70    # Moderate confidence
else:
    threat_score = 0.0     # Normal traffic
```

Appendix H

Other Figures

Additional system screenshots and diagrams are hosted in the project repository at `https://github.com/Group-Firewall/Vanguard/tree/main/docs`.

Appendix I

Research Poster

The research poster below summarises the Vanguard-NIDS research concept, problem statement, objectives, methodology, results, and national and global agenda alignment. It was prepared for the Uganda Christian University Faculty of Engineering, Design and Technology undergraduate project showcase, April 2025.

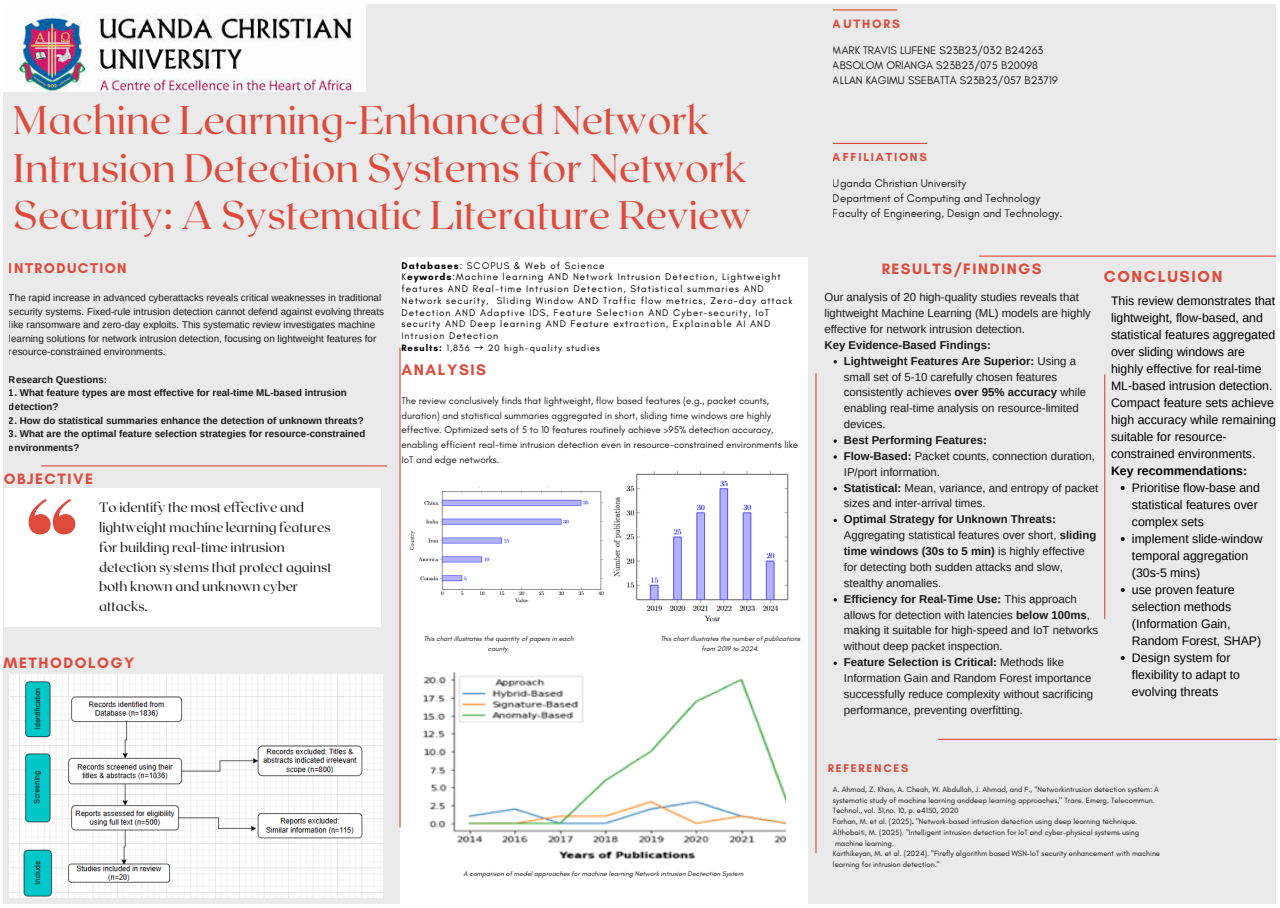

Figure I.1: Vanguard-NIDS one-page research poster.