Vigilante

Final Year Project Report

Group Members

Zaid Dandia	2021719
Sarim Ahmad	2021572
Muhammad Abdullah	2021317
Muhammad Sabeer Faisal	2021447

Supervisor:
Dr. Rashid M. Jillani

Co-Supervisor:
Dr. Khurram Khan Jadoon

Faculty of Computer Sciences & Engineering
GIK Institute of Engineering Sciences & Technology

2025

Ghulam Ishaq Khan Institute of Engineering Sciences & Technology

Faculty of Computer Science and Engineering

Vigilante

Final Year Project Report

Zaid Dandia (2021719)
Sarim Ahmad (2021572)
Muhammad Abdullah (2021317)
Muhammad Sabeer Faisal (2021447)

Advisor: Dr. Rashid M. Jillani
Co-Advisor: Dr. Khurram Khan Jadoon

2025

Certificate of Approval

It is certified that the work presented in this report was performed by **Zaid Dandia, Sarim Ahmad, Muhammad Abdullah, Muhammad Sabeer Faisal** under the supervision of **Dr. Rashid M. Jillani**. The work is adequate and lies within the scope of the BS degree in Computer Science at Ghulam Ishaq Khan Institute of Engineering Sciences and Technology.

Dr. Rashid M. Jillani

(Advisor)

Dr. Khurram Khan Jadoon

(Co-Advisor)

Prof. Dr. Qadeer Ul Hassan

(Dean)

SDGs Inclusion

Vigilante's design and deployment are intrinsically aligned with the broader mission of promoting sustainable development, particularly in the context of enhancing technological resilience, building smart infrastructures, and securing urban digital ecosystems. By integrating intelligent monitoring, dynamic anomaly detection, and adaptive security solutions, Vigilante directly supports the construction of safer, more robust digital environments. Its contributions play a pivotal role in advancing key United Nations Sustainable Development Goals (SDGs), most notably SDG 9: Industry, Innovation, and Infrastructure, and SDG 11: Sustainable Cities and Communities. Through the application of innovative technologies and forward-looking cybersecurity strategies, Vigilante exemplifies how digital platforms can contribute to sustainable, inclusive growth in the modern world.

In alignment with SDG 9, Vigilante fosters the development of resilient and intelligent infrastructure by empowering academic institutions and enterprises to establish high-availability, fault-tolerant network ecosystems. Its modular, scalable design ensures that digital infrastructures remain robust against evolving cybersecurity threats while maintaining operational efficiency. Vigilante also acts as a catalyst for technological innovation by incorporating cutting-edge AI models, federated learning techniques, and real-time threat intelligence into its framework, creating a next-generation cybersecurity platform that leverages automation and intelligent decision-making. Furthermore, Vigilante accelerates industrial digitization efforts by supporting the secure adoption of Industry 4.0 practices. Its secure-by-design philosophy guarantees that organizations can confidently expand their digital operations while safeguarding data privacy, ensuring operational integrity, and maintaining regulatory compliance, thus driving sustainable technological transformation.

Aligned with SDG 11, Vigilante significantly enhances urban digital resilience by securing the increasingly interconnected infrastructures of modern smart cities. As urban environments become more reliant on IoT devices, public service networks, and real-time communication systems, Vigilante provides the critical cybersecurity foundation needed to maintain reliability, integrity, and trust in these ecosystems. Its mobile-first administration and user-centric interfaces promote inclusive access to security management, ensuring that even resource-constrained municipalities can effectively safeguard their digital infrastructures. Moreover, Vigilante's advanced analytics and reporting capabilities enable city planners, policymakers, and institutional leaders to implement data-driven governance models. By facilitating informed decision-making on infrastructure upgrades, threat response strategies, and cybersecurity policy enforcement, Vigilante supports the creation of sustainable, secure, and resilient urban communities capable of thriving in the digital era.

Dr. Rashid M. Jillani	**Dr. Khurram Khan Jadoon**
(Advisor)	(Co-Advisor)

FINAL YEAR PROJECT MAPPING TO COMPLEX ENGINEERING/COMPUTING PROBLEM ATTRIBUTES

It is to certify here that the final year design project (FYDP) entitled **VIGILANTE** is categorized as a complex engineering/computing problem (CEP) based on the preamble (in-depth engineering knowledge) and involvement of the following attributes:

Table 1: Mapping to Complex Engineering/Computing Problem Attributes

Attribute	**Justification**
Range of Conflicting Requirements	Vigilante balances real-time performance, security, usability, scalability, and system modularity—often with trade-offs between detection accuracy and latency.
Depth of Analysis Required	Involves detailed traffic inspection, anomaly detection, and performance tuning using ML, along with threat pattern evaluation.
Depth of Knowledge Required	Involves AI/ML, network protocols, cybersecurity, real-time systems, and microservices—requiring cross-disciplinary technical understanding.
Familiarity of Issues	Deals with unfamiliar, evolving threats (e.g., zero-day attacks), requiring adaptive models instead of relying solely on known patterns.

(continued on next page)

Attribute	**Justification**
Extent of Applicable Codes	Uses cybersecurity standards (e.g., encrypted protocols, RBAC) and integrates licensed open-source tools like Zeek and Stratosphere modules.
Extent of Stakeholder Involvement and Level of Conflicting Requirements	Must satisfy institutional IT policies, academic infrastructure needs, and user-level access control, which may involve conflicting system expectations.
Consequences	Improper detection or failure can lead to system compromise, data breaches, or degraded network performance—posing operational and security risks.
Interdependence	Highly modular and event-driven system where components (e.g., AI models, alert engine, APIs) depend on each other for real-time communication and decision-making.

Dr. Rashid M. Jillani
(Advisor)

Dr. Khurram Khan Jadoon
(Co-Advisor)

Abstract

Vigilante is an advanced AI-driven network monitoring and security solution designed to address the challenges of modern institutional infrastructures, particularly at Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (GIKI). Traditional network management techniques, relying on static rules and manual interventions, often fall short in coping with the rapidly evolving landscape of cybersecurity threats and performance demands. Vigilante introduces a comprehensive, real-time system that combines network topology mapping, intelligent intrusion prevention, and dynamic traffic optimization using cutting-edge machine learning techniques.

The system adopts a hybrid microservices and event-driven architecture, ensuring modularity, scalability, and responsiveness. Vigilante's core components include a passive network traffic monitoring engine powered by Zeek, a symbolic flow encoder for efficient anomaly detection, a GRU-based Recurrent Neural Network for real-time threat classification, and an AI Optimization Module employing clustering and reinforcement learning for adaptive network performance enhancement. Integrated mobile interface empower administrators with intuitive control, live visualizations, and instant threat alerts, ensuring seamless network oversight from any location.

Extensive evaluation demonstrated Vigilante's effectiveness in improving network security and performance, achieving a classification accuracy of 97.3% in anomaly detection and enhancing bandwidth utilization by 12% while reducing network latency by 18%. By leveraging honeypots, symbolic abstractions, and AI models, Vigilante advances proactive threat mitigation and operational resilience. The project not only strengthens cybersecurity postures but also contributes towards the Sustainable Development Goals (SDGs) by

fostering resilient, secure, and smart digital infrastructures. Future directions include incorporating advanced deep learning models, SDN-based automation, and external threat intelligence integration to further extend Vigilante's capabilities.

ACKNOWLEDGMENTS

The development of *Vigilante* would not have been possible without the support and guidance of several individuals. We would like to express our sincere gratitude to:

- **Dr. Rashid M. Jillani (Advisor):**
  Thank you for your continuous guidance, mentorship, and insightful feedback throughout this project. Your expertise and support were instrumental in shaping the direction and ensuring the quality of *Vigilante*.
- **Dr. Khurram Khan Jadoon (Co-Advisor):**
  A special thanks to Dr. Jadoon, whose enthusiasm and constructive criticism greatly enriched our work. Your encouragement and technical insights played a significant role in the successful completion of this project.
- **Hafiz Syed Muhammad Muslim Shah (FYP Coordinator):**
  We are grateful to the Final Year Project coordinators for their administrative support, timely guidance, and for steering us through the project milestones with clarity and encouragement.

We are also thankful anyone else who offered assistance or encouragement during the development of *Vigilante*.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

In today's interconnected world, the importance of robust network security cannot be overstated [16]. Organizations and institutions heavily rely on networked systems for daily operations, which inherently makes them vulnerable to a variety of cyber threats [7]. These threats range from unauthorized access, malware, and Distributed Denial of Service (DDoS) attacks, to complex intrusion attempts and advanced persistent threats (APTs). Ensuring the security and integrity of network infrastructure is thus critical, as breaches can lead to substantial financial losses, compromised sensitive information, operational disruptions, and severe reputational damage [12].

Educational institutions, such as the Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (GIKI), host complex networks serving numerous users and handling sensitive academic and administrative data. These institutions must manage large-scale network traffic, optimize performance, and secure their infrastructure against evolving cybersecurity threats. Effective network management involves not only traditional security measures but also advanced analytical capabilities to detect and respond to sophisticated attacks rapidly.

Recognizing these challenges, Vigilante was conceptualized as an advanced, AI-driven Intrusion Prevention System (IPS) tailored specifically for the network infrastructure of GIKI. Vigilante is designed to provide real-time monitoring, anomaly detection, proactive threat mitigation, and network traffic optimization. Leveraging artificial intelligence (AI) and machine learning (ML), Vigilante aims to address the critical need for enhanced visibility and control over network activities, thus significantly strengthening network security and operational efficiency.

The primary motivation behind Vigilante stems from the necessity of safeguarding institutional network assets while maintaining optimal network performance. Existing network management solutions often rely on static rules and signatures, which fail to adapt quickly enough to emerging threats or network changes. Vigilante seeks to fill this gap by dynamically analyzing network traffic, detecting patterns and anomalies indicative of security threats, and automatically adjusting to optimize network resources.

Therefore, the Vigilante project responds to the urgent requirement of a scalable, responsive, and intelligent system that can effectively manage network security and performance challenges faced by modern educational institutions.

1.2 Purpose and Scope

The purpose of this document is to clearly outline the requirements and detailed description of the Vigilante system, an advanced networking solution designed specifically for real-time monitoring, management, and security enhancement of the network infrastructure at Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (GIKI). This comprehensive documentation is intended to serve as a foundational reference for software developers, stakeholders, and third-party vendors involved in the project implementation process.

The product scope of Vigilante encompasses robust real-time monitoring and detailed management of the network. Its primary aim is to enhance network visibility, optimize overall network performance, and significantly strengthen security measures through sophisticated AI-driven Intrusion Prevention System (IPS) capabilities. Vigilante's key functionalities include:

- **Real-time Device Monitoring:** Automatically discovers and displays detailed information of all connected devices, such as IP and MAC addresses.
- **Network Traffic Analysis:** Employs advanced analytical tools to monitor and visualize network traffic flows, bandwidth usage, and identify bottlenecks.
- **AI-driven Intrusion Prevention and Detection:** Uses machine learning algorithms to detect and respond proactively to anomalous behaviors and potential security threats, providing real-time alerts and automated responses to mitigate risks.
- **Mobile Accessibility:** Features a mobile application enabling network administrators to remotely manage network activities, view network status, receive real-time alerts, and configure network settings.
- **User Management and Integration:** Implements Role-Based Access Control (RBAC) to manage user permissions effectively, ensuring security and tailored access.
- **Scalability and Performance:** Designed to be scalable to accommodate various network sizes and handle high data volumes efficiently without performance degradation.

Vigilante is designed specifically as a software-based solution and does not include hardware components or in-depth network configuration management. Its target users primarily include network administrators, IT security teams, and managed service providers (MSPs), all of whom will significantly benefit from the enhanced monitoring capabilities and proactive security management offered by Vigilante.

1.3 Objectives

The primary objectives of the Vigilante project are as follows:

- **Real-time Device Monitoring:** Continuously monitor and provide visibility of the network's real-time status, identifying connected devices, traffic patterns, and potential network bottlenecks.
- **AI-driven Optimization:** Employ advanced artificial intelligence algorithms to analyze network traffic data, optimize resource allocation, reduce latency, and enhance overall network performance.
- **Intrusion Prevention:** Utilize sophisticated AI-based Intrusion Prevention Systems (IPS) to detect and proactively mitigate potential security threats and anomalies, ensuring robust network security.
- **Mobile Accessibility:** Develop and implement a user-friendly mobile application that allows network administrators to manage network operations, monitor real-time status, configure settings, and receive instant security alerts remotely.

1.4 Problem Statement

The existing network infrastructure at Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (GIKI) faces significant challenges related to network security and performance optimization. Given the complexity and scale of the network serving numerous students, faculty, and administrative personnel, the system frequently experiences issues such as unauthorized access attempts, malware intrusions, inefficient network traffic routing, bottlenecks, latency issues, and ineffective bandwidth utilization. Traditional network security systems, which rely on static rules and predefined signatures, are insufficient to cope with dynamically evolving cybersecurity threats and often result in delayed responses to potential breaches [2].

Moreover, the absence of real-time visibility and automated optimization solutions exacerbates the difficulty of swiftly detecting and mitigating network anomalies, intrusions, and performance inefficiencies. Network administrators require advanced tools that not only detect and prevent security incidents proactively but also optimize network resources dynamically to maintain seamless operations.

Vigilante is specifically designed to address these critical challenges by integrating AI-driven analytical capabilities, proactive threat detection, real-time monitoring, and automated performance optimization within a user-friendly platform accessible through mobile applications. The implementation of Vigilante will empower network administrators at GIKI to swiftly identify, respond to, and prevent network security issues, while simultaneously ensuring optimized network performance and resource allocation.

Table 1.1: Terms used in this document and their description

Name	Description
IPS	Intrusion Prevention System
IDS	Intrusion Detection System
AI	Artificial Intelligence
ML	Machine Learning
IP	Internet Protocol
MAC	Media Access Control
RBAC	Role-Based Access Control
MSP	Managed Service Provider
GUI	Graphical User Interface
TCP	Transmission Control Protocol
HTTPS	HyperText Transfer Protocol Secure
SSL	Secure Sockets Layer
TLS	Transport Layer Security
SNMP	Simple Network Management Protocol

1.5 Project Significance

The Vigilante project holds significant importance as it directly addresses essential aspects of network security and operational efficiency within educational institutions, particularly at Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (GIKI). Aligning closely with industry-recognized security best practices, Vigilante integrates AI-driven methodologies to proactively identify and mitigate security threats, providing continuous and comprehensive security coverage.

By employing real-time monitoring, Vigilante enhances the immediate visibility of network activities, enabling swift detection and management of anomalies, unauthorized access, and potential breaches [14]. The AI-driven optimization strategies significantly improve network performance by dynamically managing traffic flows, minimizing latency, reducing packet loss, and optimizing bandwidth utilization.

The stakeholders, including network administrators, IT security teams, and managed service providers (MSPs), will benefit extensively from the capabilities provided by Vigilante. The system ensures enhanced security, reduced operational overhead, and improved network reliability, ultimately contributing to the institution's academic and administrative efficiency [10].

2 Literature Survey

2.1 Overview

Network security encompasses practices and policies adopted to prevent and monitor unauthorized access, misuse, modification, or denial of computer networks and network-accessible resources. The increasingly sophisticated nature of cyber threats necessitates advanced security mechanisms and proactive strategies to protect sensitive data and maintain network integrity.

Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are critical components of modern network security strategies. An IDS monitors network traffic to identify suspicious activities or known threats, issuing alerts to network administrators for further investigation. In contrast, an IPS not only detects threats but actively takes preventive measures to block or mitigate malicious activities immediately. These systems rely heavily on detecting abnormal patterns or known threat signatures within the network data.

The integration of Artificial Intelligence (AI) into network security systems significantly enhances their capabilities. AI-driven systems employ machine learning algorithms and data analytics to identify anomalies and predict potential threats in real-time. This predictive capability enables faster detection and more accurate responses compared to traditional rule-based systems. Additionally, AI-based optimization techniques ensure efficient network resource management, reducing latency, improving bandwidth allocation, and enhancing overall network performance.

By combining IDS, IPS, and AI-driven methodologies, contemporary network security solutions offer robust, adaptive, and proactive measures necessary to counteract sophisticated and evolving cyber threats, thereby providing comprehensive protection and efficient network performance management.

2.2 Review of Existing Solutions

Several existing solutions utilize advanced technologies to manage network security effectively. The following are notable examples, each with distinct advantages and limitations:

2.2.1 Cisco DNA Center

Key Concepts

Cisco DNA Center is built around intent-based networking and centralized orchestration, which directly resonates with Vigilante's vision of unified control. Its ability to segment networks and automate policies demonstrates how software-defined networking principles can improve manageability, something Vigilante adapts through modular microservices for IDS, optimization, and device tracking [6].

AI Application

The predictive AI capabilities of Cisco DNA highlight the benefits of using historical and live data to preemptively mitigate failures—an approach mirrored in Vigilante's AI optimization module. Cisco's assurance engine, which uses machine learning to spot deviations, aligns with Vigilante's anomaly detection models that continuously learn from streaming data [6].

Security Measures

Stealthwatch integration showcases effective behavioral analytics in encrypted traffic—an increasingly vital function. Vigilante draws from this by incorporating AI to monitor encrypted sessions and deploy honeypots for deception-based defense, covering encrypted and unencrypted channels [6].

Network Optimization

Cisco DNA's traffic telemetry and application-aware path selection parallel Vigilante's AI-based routing enhancements, ensuring minimal delay and adaptive bandwidth utilization based on real-time performance indicators [6].

Advantages

Offers centralized management, leveraging AI for predictive analytics and anomaly detection, thus significantly enhancing network reliability and reducing operational costs.

Limitations

Highly optimized for Cisco devices, potentially limiting compatibility and integration capabilities with non-Cisco equipment.

2.2.2 Juniper Mist AI

Key Concepts

Juniper Mist's cloud-native architecture and focus on end-user experience set an important precedent for user-centric design. Vigilante applies this mindset by prioritizing real-time visualization of network events and mobile accessibility for administrators through its dedicated app [9].

AI Application

The Marvis assistant exemplifies how natural language processing and AI-powered recommendations can simplify network management. Vigilante plans to incorporate intelligent alert summarization and potentially conversational interfaces for reporting and response [9].

Security Measures

Although Juniper Mist relies on third-party integration for full-spectrum security, its proactive performance diagnostics are foundational to anomaly detection. Vigilante combines this strength with embedded IDS/IPS modules powered by CNNs and Autoencoders to ensure holistic threat visibility [9].

Network Optimization

Juniper Mist's automatic RF tuning and Wi-Fi assurance offer a practical blueprint for Vigilante's optimization engine, which dynamically adapts routing and traffic shaping policies based on evolving conditions [9].

Advantages

Integrates AI-driven insights and autonomous troubleshooting to significantly enhance user experience, operational efficiency, and response times to security threats.

Limitations

Primarily focuses on performance optimization and may lack comprehensive intrusion prevention capabilities found in more security-focused solutions.

2.2.3 Palo Alto Prisma Access

Key Concepts

Prisma Access illustrates the potential of Secure Access Service Edge (SASE) models in supporting decentralized and mobile-first networks. Vigilante integrates this philosophy by ensuring full remote management and distributed logging capabilities across subnets and institutions [15].

AI Application

By integrating threat intelligence with AI-powered analytics (Cortex XDR, WildFire), Prisma Access underscores the importance of continuous model enrichment. Vigilante adapts this by allowing retraining of its IDS models via data pipelines sourced from honeypots and admin-labeled events [15].

Security Measures

With built-in IPS, DNS filtering, and ZTNA policies, Prisma sets the benchmark for layered defenses. Vigilante parallels this with tiered detection thresholds and a modular threat scoring system, escalating high-severity events for immediate mitigation [15].

Network Optimization

Prisma's ability to maintain seamless application delivery through dynamic WAN path adjustment is echoed in Vigilante's traffic prioritization logic and AI-based congestion avoidance, maintaining user experience even under anomalous load [15].

Advantages

Combines secure access and real-time visibility through cloud-based management, utilizing advanced analytics and AI for enhanced threat detection, effectively combating sophisticated cyber threats.

Limitations

Deployment and management complexity might be higher due to comprehensive features, potentially requiring extensive training for optimal use.

2.2.4 SolarWinds Network Performance Monitor

Key Concepts

As a classic example of centralized performance monitoring, SolarWinds NPM emphasizes vendor-neutral visibility and infrastructure diagnostics. Vigilante leverages this philosophy by offering multi-layer visibility (physical, logical, and behavioral) across diverse network devices and architectures [19].

AI Application

While AI capabilities in SolarWinds are limited, tools like PerfStack and NetPath enable a form of event correlation. Vigilante extends this idea with embedded ML pipelines that generate insights into long-term behavior changes and performance baselines [19].

Security Measures

SolarWinds is not a dedicated security solution but complements others via syslog ingestion and SNMP alerts. Vigilante advances this by offering direct intrusion response, threat classification, and an AI-led alerting system that prioritizes accuracy and minimal false positives [19].

Network Optimization

The system's strength lies in its detailed visualization of performance issues and fault domains. Vigilante expands on this by using AI models to not only diagnose but proactively adjust network behavior, improving reliability and response time.

These systems provide valuable context for Vigilante's development, highlighting how enterprise-grade solutions utilize AI, analytics, and automation. Vigilante draws from their strengths while offering a modular, open-source, and research-backed alternative tailored for academic and institutional use cases [19].

Advantages

Provides robust performance management and troubleshooting, utilizing AI-driven anomaly detection for proactive issue resolution.

Limitations

Primarily designed for performance monitoring, necessitating additional modules for complete security management capabilities, potentially leading to increased costs and complexity.

2.2.5 Paper 1: Enabling Intrusion Detection Systems with Dueling Double Deep Q-Learning

Key Concepts

Badr's work emphasizes a shift from static, data-driven models to adaptive learning systems. Traditional IDS often struggle with concept drift, outdated signatures, and limited generalization. Badr introduces the concept of using a dueling double deep Q-learning architecture to enhance IDS responsiveness by learning directly from environmental interactions rather than relying solely on historical datasets. This aligns closely with Vigilante's goal of incorporating adaptive intelligence within a modular IDS framework [3].

AI Application

The proposed architecture in Badr's paper combines reinforcement learning (RL) with deep learning (DL) via a dueling double deep Q-network (DDDQN). This architecture enables Vigilante to detect anomalies in real time and learn optimal feature sets dynamically through trial-and-error interaction. Unlike purely supervised models, RL models in Vigilante are trained to maximize cumulative rewards, improving their decision-making capabilities even in unfamiliar network environments [3].

Security Measures

Security effectiveness is enhanced through cost-aware classification using costly feature classification (CwCF) logic. This allows the RL agent to weigh performance against computation or resource constraints, only acquiring high-cost features when necessary. Vigilante's future work roadmap includes implementing this intelligent trade-off mechanism, thereby optimizing performance and detection accuracy while maintaining system efficiency [3].

Network Optimization

Badr's architecture also supports resource-aware adaptation by dynamically selecting relevant features for classification, a concept that Vigilante applies in its AI optimization module. This not only conserves computational power but also reduces false positives and ensures bandwidth-efficient decision-making, reinforcing Vigilante's role as a proactive and intelligent network management platform [3].

Advantages

- Adaptive learning based on real-time interaction with the environment.
- Strong generalization and resistance to concept drift.
- Efficient decision-making under resource constraints using CwCF.

Limitations

- High computational requirements for training RL models.
- Real-world effectiveness depends heavily on quality and diversity of interaction data.
- Reward design and exploration strategies can significantly impact performance and convergence.

This paper forms a critical theoretical backbone for Vigilante's AI module design and validates its event-driven, reinforcement-learning-enhanced direction.

2.2.6 Paper 2: Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey

Key Concepts

This survey explores the foundational principles and comparative frameworks of machine learning (ML) and deep learning (DL) in the context of IDS. It categorizes ML-based IDS into signature-based and anomaly-based systems, which Vigilante also leverages in its hybrid detection pipeline. Liu & Lang highlight the importance of generalization, real-time responsiveness, and adaptive learning—three pillars deeply embedded in Vigilante's system design [13].

AI Application

Liu & Lang's survey reviews a wide spectrum of AI techniques—such as CNNs for pattern recognition, RNNs for sequential data modeling, and Autoencoders for unsupervised anomaly detection. These models are instrumental in Vigilante's AI module, where CNNs classify incoming traffic, RNNs analyze long-term behavior patterns, and Autoencoders detect outliers. The paper's recommendation for ensemble and hybrid AI systems reinforces Vigilante's strategy of combining supervised and unsupervised learning for robust threat detection [13].

Security Measures

The survey emphasizes balancing precision, recall, and false-positive rates while maintaining computational efficiency. Vigilante reflects this balance by integrating its AI models with real-time traffic sampling and honeypot feedback loops. This ensures that alerts are both relevant and actionable. Liu &

Lang's analysis of IDS datasets and evaluation criteria supports Vigilante's methodical testing and validation process [13].

Network Optimization

Though primarily focused on threat detection, Liu & Lang also touch on the role of ML in network behavior prediction and performance management. Vigilante capitalizes on this by integrating ML-driven traffic classifiers with real-time network statistics, enabling proactive congestion control and bandwidth optimization [13].

This paper substantiates Vigilante's multi-layered AI approach and affirms its alignment with state-of-the-art trends in intelligent intrusion detection systems.

Advantages

- Offers a comprehensive overview of diverse ML/DL techniques and architectures.
- Validates hybrid approaches for high-accuracy, low-latency IDS.
- Encourages evaluation metrics and standard datasets for performance benchmarking.

Limitations

- Survey scope remains theoretical without offering implementation-specific insights.
- Limited guidance on real-time model deployment challenges.
- Minimal discussion on reinforcement learning or adaptive agents.

2.3 AI in Network Security

Artificial Intelligence significantly improves the capabilities of Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) by leveraging advanced machine learning algorithms [18] [8]. These AI methodologies enhance threat detection accuracy by analyzing large datasets to recognize patterns and anomalies indicative of malicious activities. The utilization of AI reduces false positives, effectively handles unknown zero-day vulnerabilities, and provides rapid responses to detected threats, thus improving the overall security posture [5].

Literature highlights various machine learning and deep learning techniques employed in IDS, such as Deep Reinforcement Learning (DRL) for action-selection strategies, Deep Q-Networks (DQN), and Autoencoders for anomaly detection. These advanced methods significantly boost detection accuracy, facilitate proactive threat mitigation, and ensure rapid response to security incidents.

Table 2.1: Existing Systems

Product/Research Paper	**Key Concepts**	**AI Application**	**Security Measures**	**Network Optimization**
Cisco DNA Center [6]	Network mapping, automation, security, AI-driven analytics	AI for intelligent issue detection and analysis	Integrates with Cisco's broader security ecosystem	Offers intrusion-detection and prevention features
Juniper Mist AI [9]	Network analytics, performance optimisation, security features	AI for network analytics and performance optimisation	Basic security features	Monitors traffic and adapts to changing conditions
Palo Alto Networks Prisma Access [15]	Network monitoring, intrusion prevention, AI-based analytics	AI-based analytics for threat detection and policy optimisation	Combines network monitoring with IPS	Detects threats and tunes security policies
SolarWinds Network Performance Monitor [19]	Security / traffic-analysis modules, network mapping	Customisable AI-based optimisation features	Extendable with additional security modules	Provides comprehensive network management
Enabling IDS with Dueling Double DQN [3]	Intrusion-detection systems, network security	Deep-RL (DQL) to improve IDS action selection	Dueling DDQN architecture for network intrusion detection	Learns optimal policies for network optimisation
ML and DL Methods for IDS: A Survey [13]	IDS taxonomy; ML and DL techniques in IDS	ML for normal / abnormal traffic, autoencoders for anomalies	AI in IDS (under AI Application)	Traffic grouping, packet-parsing & feature-engineering detection

3 Design

3.1 System Requirements

External Interfaces for the Vigilante system are categorized into User Interfaces, Software Interfaces, and Communication Interfaces, each critical to the effective operation and user interaction of the system.

3.1.1 User Interfaces

- **Admin Dashboard:** Provides a centralized, mobile-based interface for real-time network traffic monitoring, alert management, user and role administration, and comprehensive reporting capabilities.
- **Login Interface:** Ensures secure system access with option for Role-Based Access Control (RBAC), restricting system use to authorized personnel.
- **Configuration Settings Interface:** Allows administrators to customize system settings such as monitoring intervals, alert thresholds, and AI optimization parameters, ensuring flexibility and alignment with institutional requirements.

3.1.2 Software Interfaces

- **Database Integration:** Employs NoSQL databases like Firebase to effectively manage and store unstructured network traffic data.
- **Machine Learning and AI Models:** Integrates external AI/ML libraries

such as TensorFlow and PyTorch for traffic data analysis and application of optimization strategies.

3.1.3 Communication Interfaces

- **Network Protocols:** Utilizes standard TCP/IP for general network communication and HTTPS/SSL/TLS for secure data exchanges with user interfaces. SNMP is employed for network device monitoring and troubleshooting.
- **Notification Protocols:** Incorporates services like Firebase Cloud Messaging (FCM) and Apple Push Notification Service (APNs) for real-time push notifications.
- **API Communication:** Provides RESTful APIs for interaction with third-party systems and WebSockets for instant, two-way communication between frontend and backend components.
- **External Device Communication:** Implements Syslog for log management and integrates third-party IDS/IPS and firewalls through standard security protocols like IPSec.

3.1.4 Functional Requirements

- **FR-001:** Continuous real-time network traffic monitoring and data recording.
- **FR-002:** Analyze captured network traffic to identify patterns and anomalies.
- **FR-003:** Detect anomalies based on traffic analysis and flag suspicious behavior.

- **FR-004:** Generate immediate alerts upon detecting potential intrusions.
- **FR-005:** Continuously monitor and adjust AI optimization strategies.
- **FR-006:** Analyze traffic data to optimize network performance.
- **FR-007:** Log all intrusion events, including detection time, intrusion type, and preventive actions.
- **FR-008:** Allow administrators to configure network monitoring intervals, alert thresholds, and data retention policies.

3.1.5 Non-Functional Requirements

- **NFR-001 (Performance):** Process and analyze network traffic with minimal latency, targeting a response time under 500 milliseconds.
- **NFR-002 (Scalability):** Ability to scale horizontally to support networks from 1,000 to 10,000 devices without performance degradation.
- **NFR-004 (Availability):** Maintain nearly 100% uptime, limiting downtime to less than 8 hours per year.
- **NFR-005 (Reliability):** Ensure data integrity and consistency during network operations.
- **NFR-006 (Usability):** Provide intuitive and easy-to-navigate user interfaces, enabling quick adaptation and minimal training.
- **NFR-007 (Maintainability):** Implement modular architecture for easy system updates within a maintenance window of 2-4 hours without significant downtime.

3.2 System Architecture

The Vigilante system architecture is structured using the 4+1 Architecture View Model, which provides a comprehensive framework illustrating different aspects and interactions within the system. The model includes the Use Case View, Logical View, Development View, Process View, and Physical View, each highlighting unique system dimensions critical for effective design and deployment.

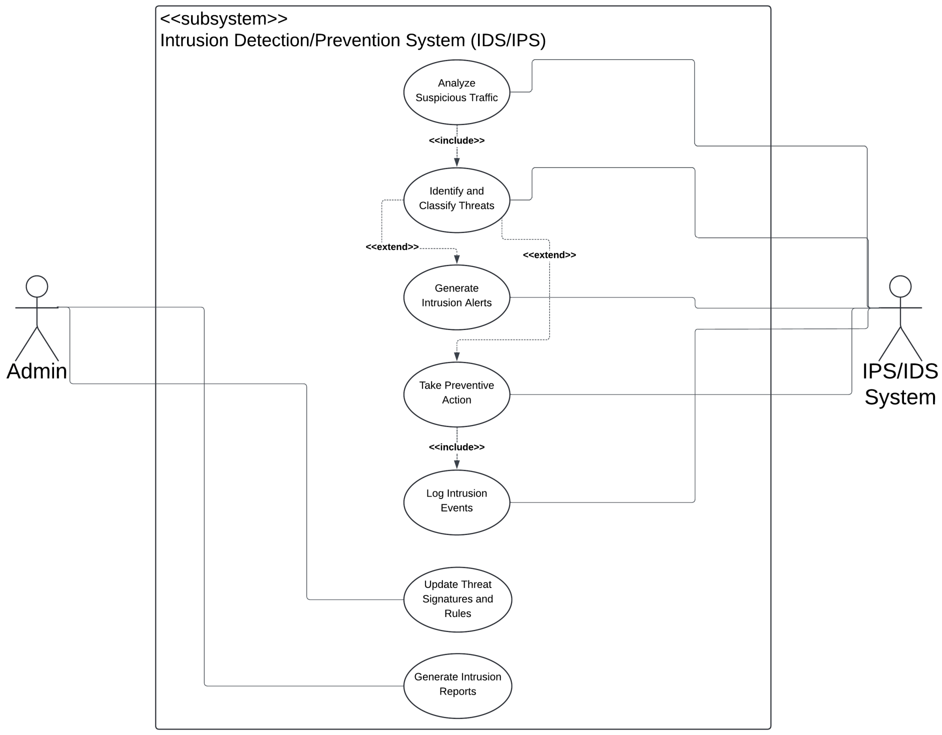
3.2.1 Use Case View

The use case views depict the primary interactions and functionalities Vigilante supports:

IPS/IDS Use Case

Manages detection and prevention of network intrusions, actively monitoring for anomalies and generating alerts.
Description: This use case diagram 3.1 represents the flow of actions involved in real-time network monitoring. Both the administrator and system user (e.g., automated system or technician) interact with the module.

- The process begins with Monitor Network Traffic, where the system passively observes all incoming and outgoing packets.
- It continues to Analyze Traffic Patterns, using statistical analysis and ML models to identify common behaviors.
- Detect Anomalies identifies deviations from the norm, which might indicate security threats or performance issues.

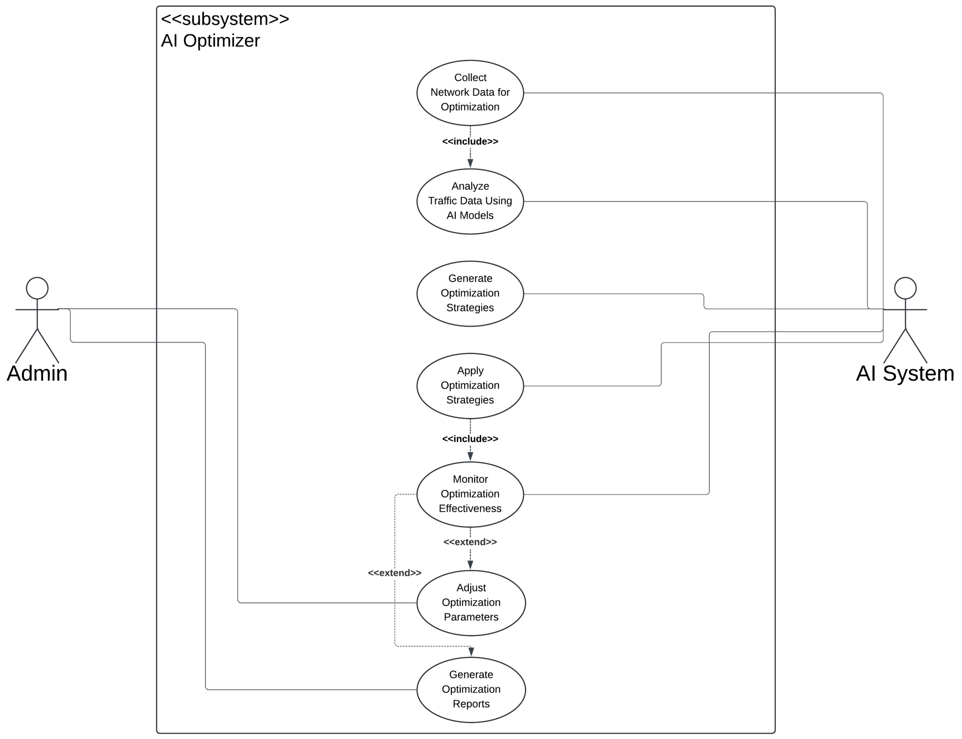


<<subsystem>>
Intrusion Detection/Prevention System (IDS/IPS)
Analyze Suspicious Traffic
<<include>>
Identify and Classify Threats
<<extend>>
<<extend>>
Generate Intrusion Alerts
Take Preventive Action
<<include>>
Log Intrusion Events
Update Threat Signatures and Rules
Generate Intrusion Reports
Admin
IPS/IDS System

Figure 3.1: Use Case View - IPS/IDS

- The system then Generates Network Reports that summarize usage trends, anomalies, and historical data.
- If necessary, the system Alerts Admin for Network Issues through notifications.
- The administrator can Configure Network Monitoring Settings to adjust thresholds, logging frequency, and analysis parameters.

This module ensures the network is constantly observed, and any irregularities are quickly reported and documented for further action.



<<subsystem>>
AI Optimizer
Collect Network Data for Optimization
<<include>>
Analyze Traffic Data Using AI Models
Generate Optimization Strategies
Apply Optimization Strategies
<<include>>
Monitor Optimization Effectiveness
<<extend>>
<<extend>>
Adjust Optimization Parameters
Generate Optimization Reports
Admin
AI System

Figure 3.2: Use Case View - AI Optimizer

AI Optimizer Use Case

Focuses on real-time network traffic analysis and optimization strategies to enhance network performance.

Description: This use case diagram 3.2represents the flow of actions involved in real-time network monitoring. Both the administrator and system user (e.g., automated system or technician) interact with the module.

- The process begins with Monitor Network Traffic, where the system passively observes all incoming and outgoing packets.
- It continues to Analyze Traffic Patterns, using statistical analysis and

ML models to identify common behaviors.

- Detect Anomalies identifies deviations from the norm, which might indicate security threats or performance issues.
- The system then Generates Network Reports that summarize usage trends, anomalies, and historical data.
- If necessary, the system Alerts Admin for Network Issues through notifications.
- The administrator can Configure Network Monitoring Settings to adjust thresholds, logging frequency, and analysis parameters.

This module ensures the network is constantly observed, and any irregularities are quickly reported and documented for further action.

Network Monitoring Use Case:

Oversees comprehensive monitoring, data collection, and visualization of network statuses and traffic flows.

Description: This use case 3.3 outlines the AI-driven network optimization functionality of Vigilante. It dynamically improves network performance using insights from traffic behavior.

- It starts by Collecting Network Data for Optimization, pulling performance metrics and traffic logs.
- The system then Analyzes Traffic Data Using AI Models, identifying bottlenecks, underutilized paths, and high-priority flows.
- It proceeds to Generate Optimization Strategies, such as load balancing or route prioritization.

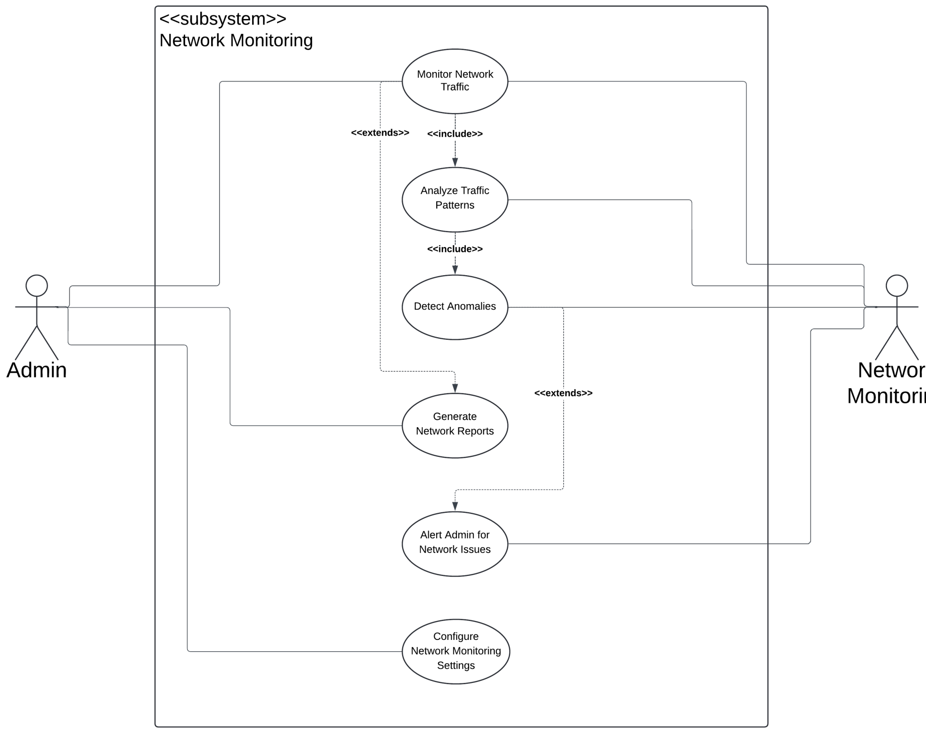


Figure 3.3: Use Case View - Network Monitor

- These strategies are then applied to the network via routing updates or QoS adjustments.
- The system continues to monitor Optimization Effectiveness in real time, ensuring changes result in performance improvement.
- If necessary, adjusts optimization parameters based on feedback or changing network conditions.
- It concludes with Generating Optimization Reports that provide analytics on throughput gains, latency reduction, and applied changes.

This module exemplifies Vigilante's smart automation, reducing human overhead while maximizing performance and reliability.

3.2.2 Logical View

The logical architecture provides a detailed representation of the system's object-oriented design:

Class Diagrams:

Figure 3.4 defines the structural relationships among different classes representing the Vigilante system's components.

Core Classes and Responsibilities

- Network, Subnet, and Device represent the foundational infrastructure components. The Network class aggregates subnets and connected devices, capable of mapping the network, identifying bottlenecks, and generating graphical representations. Each Subnet monitors traffic and detects local faults, while each Device can independently monitor traffic, detect intrusions, and optimize flows.

- Connection manages the communication between devices, with properties like bandwidth, latency, and health status. It enables the visualization and optimization of device interactions.

- User encapsulates administrator details and interactions. Through login, logout, and device configuration, this class secures and controls system access, reinforced by role-based access management.

- IntrusionDetectionSystem (IDS) performs deep traffic scans, threat detection, and alert generation. It maintains a list of active threats and issues alerts to relevant users for real-time response.

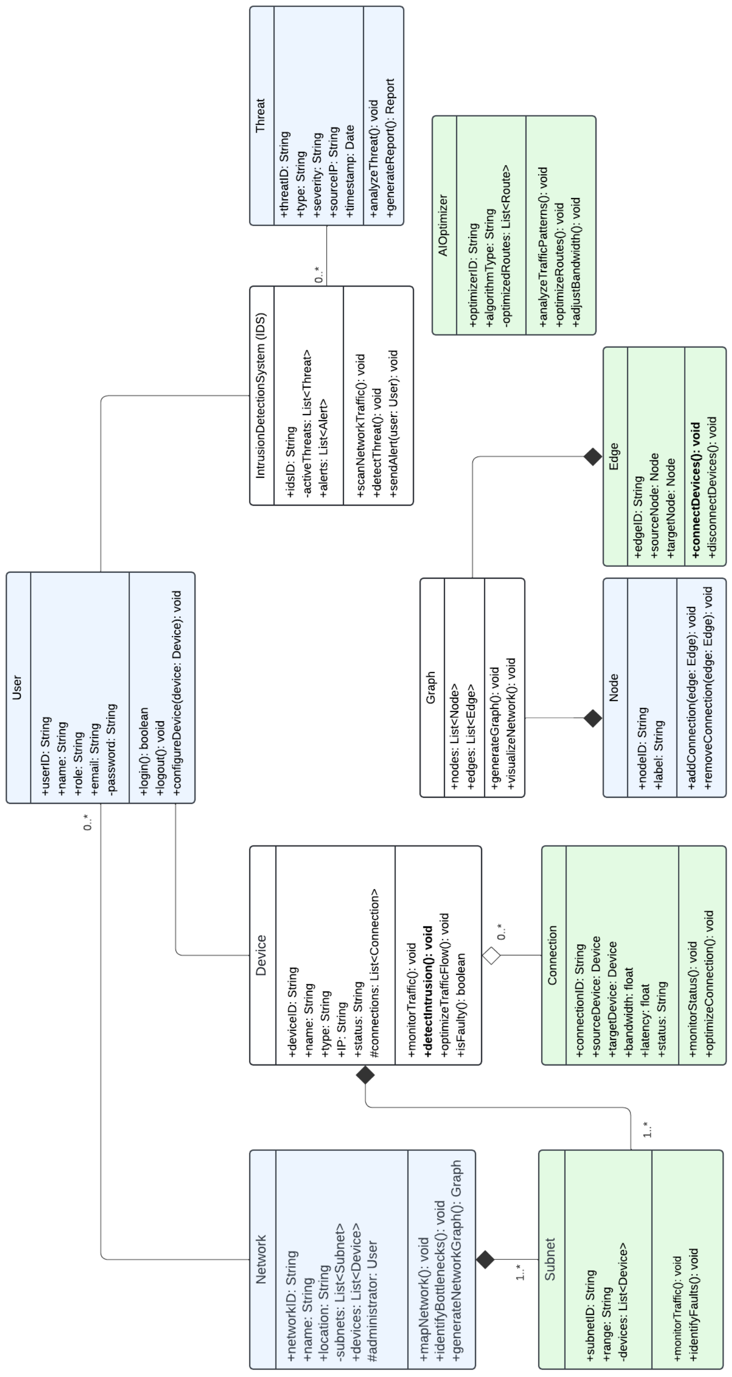


Figure 3.4: Class Diagram of Vigilante

- Threat captures characteristics of identified threats such as type, source IP, and severity. It includes methods to analyze and report on attack behavior for later study and countermeasure planning.

- AIOptimizer is central to Vigilante's intelligence. It analyzes traffic patterns using different algorithms (e.g., RL, DL) to adjust routing, optimize throughput, and allocate bandwidth efficiently.

- Graph, Node, and Edge represent the data structures used to model and visualize the network topology. Graph aggregates Node and Edge instances, and supports adding, removing, or updating network visual components.

Design Insights This structure allows Vigilante to maintain high modularity and clear separation of concerns. The relationships among Network, Device, and Connection classes provide the data backbone for real-time monitoring, while IDS and AIOptimizer build the decision-making core. The Graph structure enables live topology visualization, aiding administrators in diagnostics and optimization.

This class model also facilitates future extensibility, such as integrating blockchain for audit trails or adding federated learning to the optimizer module. Overall, the diagram supports Vigilante's objective of secure, intelligent, and scalable network defense.

Object Diagrams:

Figure 3.5 illustrates specific instances of classes and their interactions within the system. The object diagram for Vigilante represents a snapshot of the system's state at a specific moment in time, showcasing real instances of the

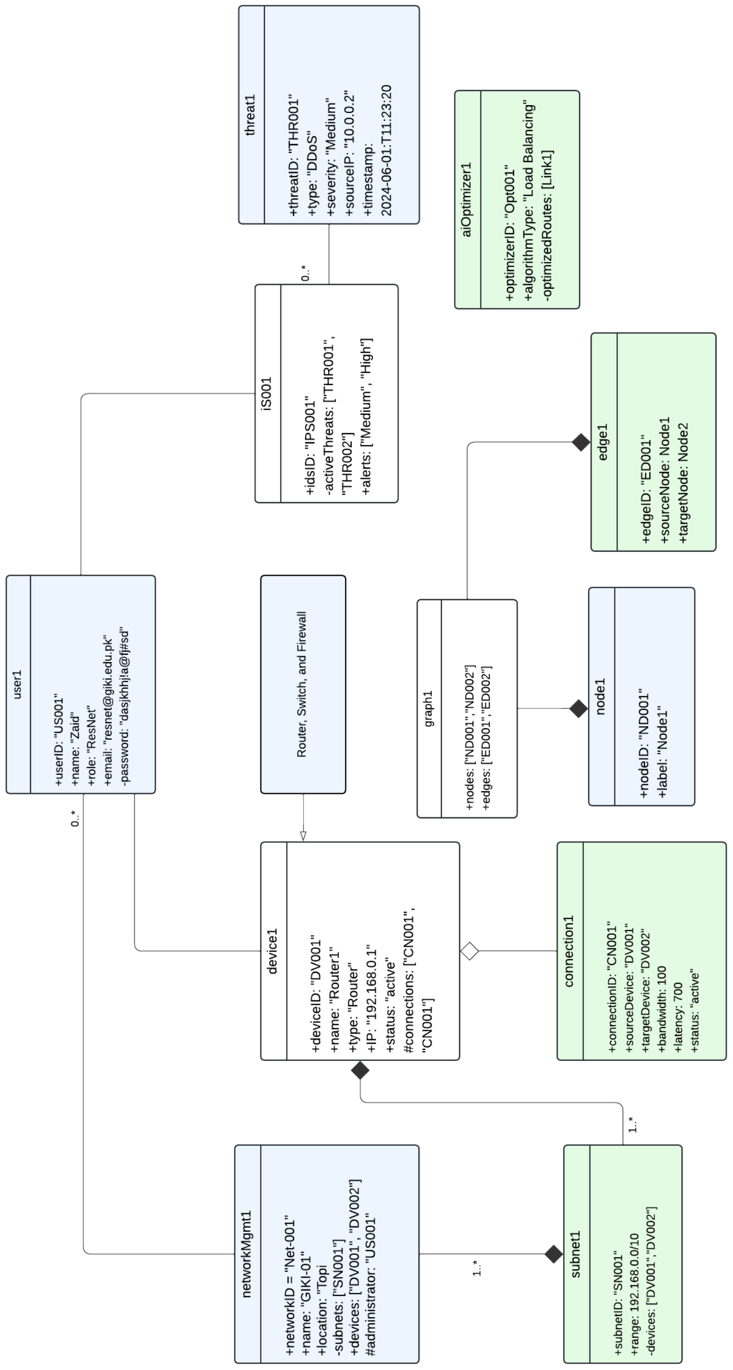


Figure 3.5: Object Diagram of Vigilante

classes defined in the class diagram. It offers clarity on how different components interact during runtime and illustrates how relationships are maintained.

Runtime Instances and Relationships

- networkMgmt1 defines an actual network instance titled ”GIKI-01”. It shows an active topology with associated subnets (subnet1), devices (device1), and an administrator (user1).
- device1 (e.g., ”Router1”) is actively managing connections and participating in network communication via connection1. This object has a specific IP, type, and status, showcasing how real-time attributes are tracked.
- subnet1 groups devices within a defined IP range (192.168.0.0/10) and references the specific devices it governs.
- connection1 illustrates a live link between two devices with defined properties such as latency and bandwidth, critical for AI-based optimization.

Security System in Action

- iS001 (an active IDS instance) is monitoring traffic and detecting threats (THR001, THR002), classified by threat1. These threats are associated with severity and timestamps, which help determine immediate countermeasures.
- threat1 (type: DDoS) is an example of a detected anomaly, including metadata such as severity, origin, and timing—important for correlation and logging.

Optimization Process at Runtime

- aiOptimizer1 is operational, applying the "Load Balancing" algorithm on a specific route (Link1). It reflects how the system dynamically adjusts traffic flows using its optimization logic.
- graph1, with its node1 and edge1, demonstrates the current network visualization model—where nodes and edges represent the devices and connections in real time.

Design Insight

This diagram helps understand Vigilante's live behavior—showing how threats are monitored, optimizations are applied, and relationships among components remain consistent. It also highlights the system's ability to provide contextual awareness through structured object interactions and adaptive intelligence.

This object-level perspective complements the class diagram by verifying the real-world feasibility of Vigilante's structural design.

Composite Structure Diagrams:

Detail complex interactions within the system, specifically for modules like User Management, AI Optimization, IPS/IDS, and Network Management.
The composite structure diagram for Vigilante breaks down the internal collaborations within major subsystems: Network Management, Intrusion Detection and Prevention System (IDPS), AI Optimization, and User Authentication. Each composite block reveals the internal roles and their interaction patterns, reinforcing Vigilante's modular architecture.

User Management 3.6 Description

- authenticator validates user credentials.

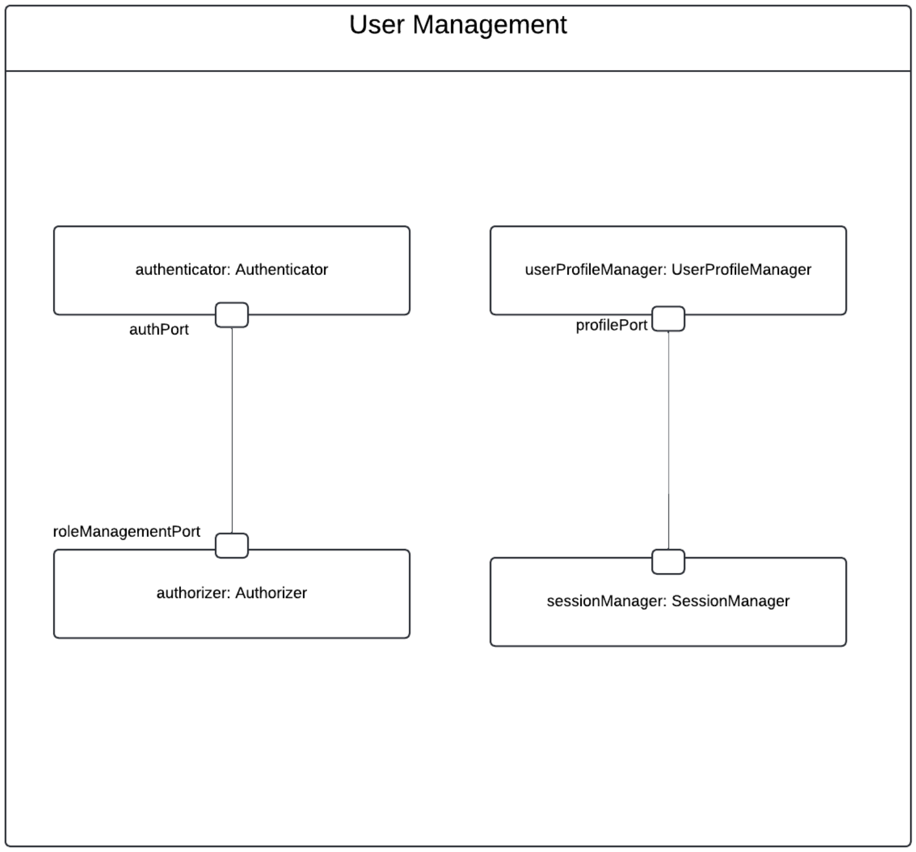

User Management
authenticator: Authenticator
authPort
userProfileManager: UserProfileManager
profilePort
roleManagementPort
authorizer: Authorizer
sessionManager: SessionManager

Figure 3.6: User Management Composite Structure Diagram

- authorizer grants access based on roles defined in the user profile.
- userProfileManager manages role, contact, and permission data.
- sessionManager maintains secure user sessions, ensuring traceability and timeout policies. This supports Vigilante's secure multi-user administration and RBAC model.

AI Optimization 3.7 Description

- trafficAnalyzer feeds summarized traffic behavior to the aiModelHan-

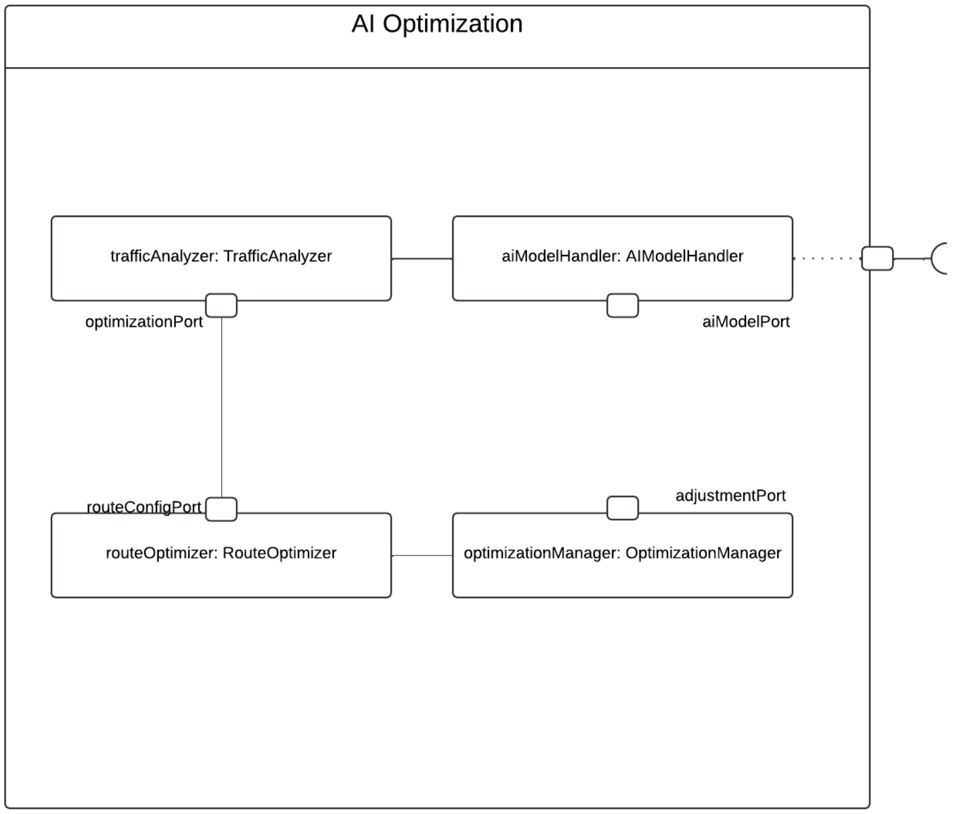

AI Optimization
trafficAnalyzer: TrafficAnalyzer
aiModelHandler: AIModelHandler
optimizationPort
aiModelPort
routeConfigPort
adjustmentPort
routeOptimizer: RouteOptimizer
optimizationManager: OptimizationManager

Figure 3.7: AI Optimization Composite Structure Diagram

dler, which selects and invokes the appropriate ML model.

- routeOptimizer evaluates different routing possibilities for better performance.
- optimizationManager applies the most effective strategies based on feedback. This modularity allows Vigilante to self-tune network performance with minimal admin input.

IPS/IDS System 3.8 Description

- trafficAnalyzer inspects data packets to detect suspicious activity

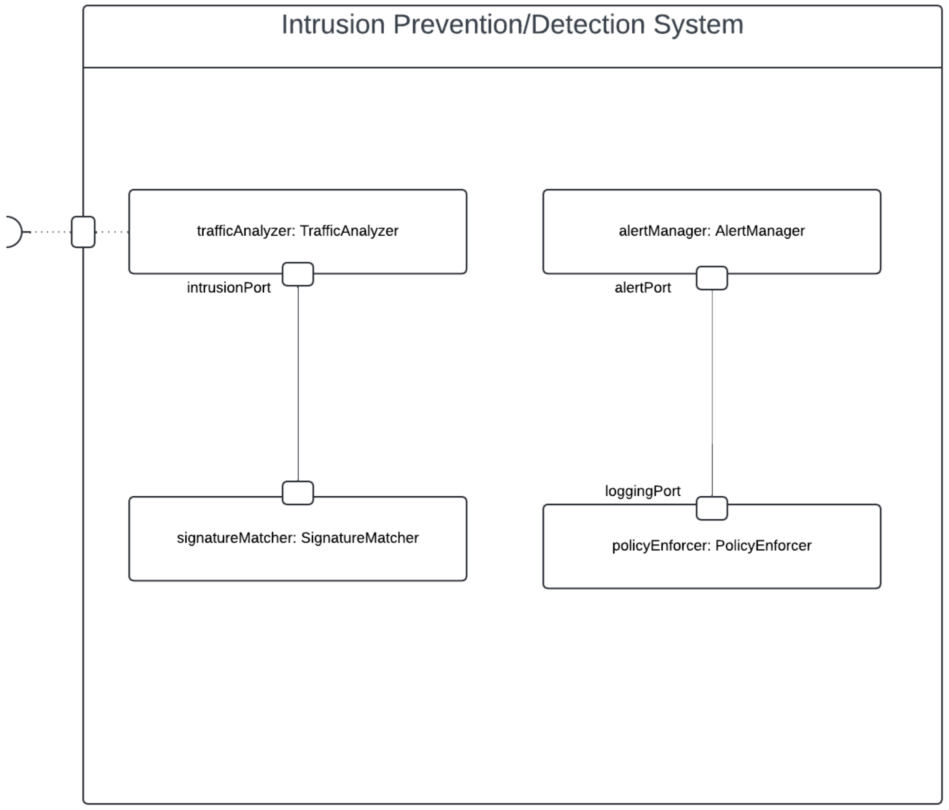

Intrusion Prevention/Detection System
trafficAnalyzer: TrafficAnalyzer
intrusionPort
alertManager: AlertManager
alertPort
loggingPort
signatureMatcher: SignatureMatcher
policyEnforcer: PolicyEnforcer

Figure 3.8: IPS/IDS Composite Structure Diagram

- signatureMatcher compares live traffic against known malicious patterns
- alertManager sends real-time alerts to admins based on detection rules.
- policyEnforcer automatically applies block or quarantine rules to suspicious connections. This collaboration reflects Vigilante's AI-augmented reactive defense mechanisms.

Network Management 3.9 Description

- networkScanner and deviceController handle discovery and inventory of all network elements.

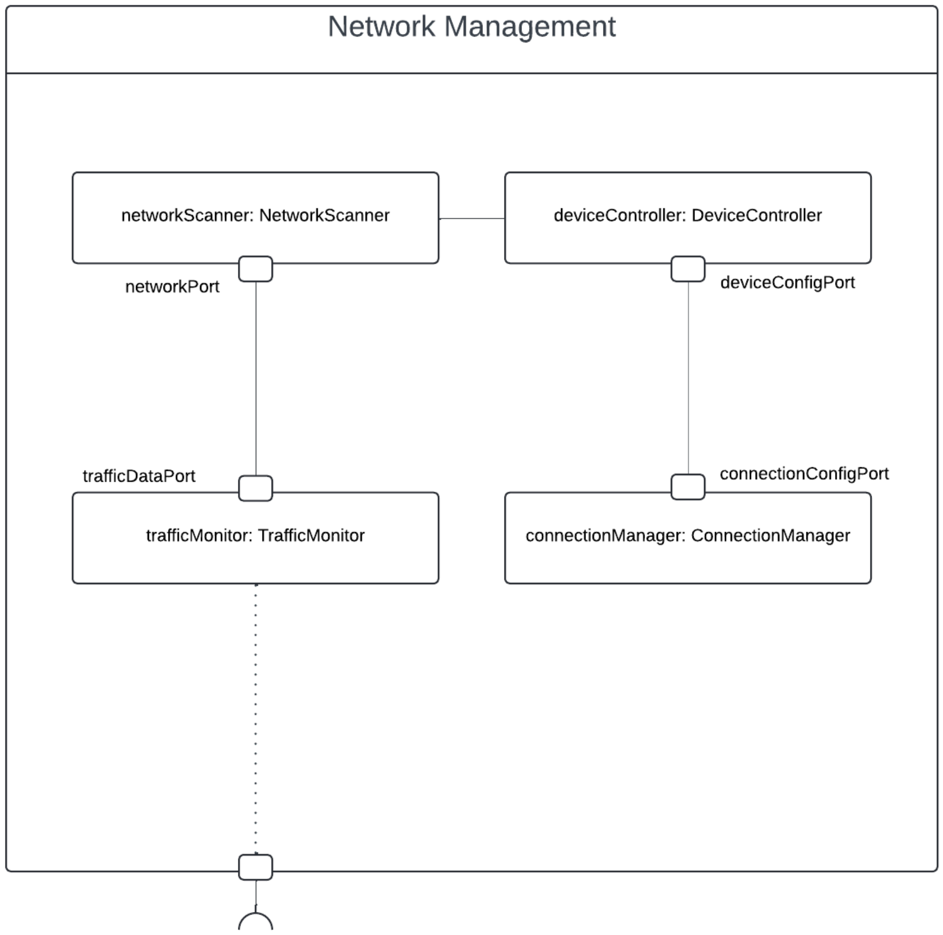

Network Management
networkScanner: NetworkScanner
deviceController: DeviceController
networkPort
deviceConfigPort
trafficDataPort
connectionConfigPort
trafficMonitor: TrafficMonitor
connectionManager: ConnectionManager

Figure 3.9: Network Management Composite Structure Diagram

- trafficMonitor continually observes packet flow for anomalies, feeding data into other components.
- connectionManager oversees the establishment, monitoring, and health status of logical and physical links. This structure ensures that network topology is actively managed and synchronized with live traffic states.

3.2.3 Development View

Component Diagram

Highlights major system components such as backend services, AI modules, databases, notification services, and their interconnections.

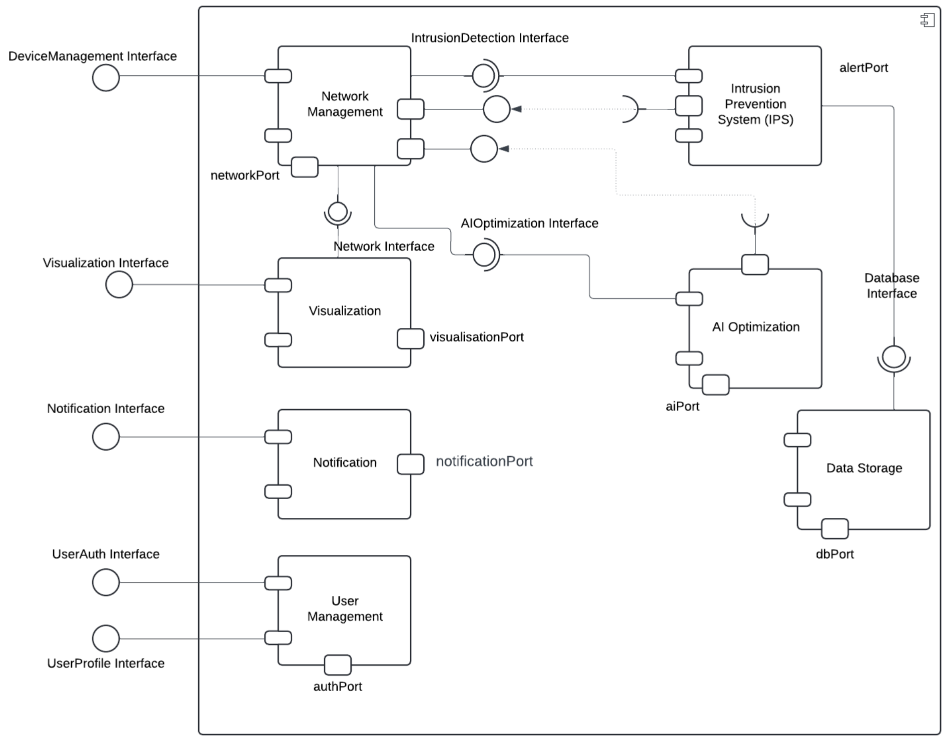


Figure 3.10: Component Diagram of Vigilante

Description The component diagram 3.10 illustrates the high-level architecture of Vigilante by representing each major functional module and how they communicate through well-defined interfaces and ports. It highlights the service-oriented and modular nature of the system.

- **Network Management:** The Network Management component interfaces with the rest of the system via networkPort and manages traffic data, device statuses, and routing configurations. It plays a central coordinating role and provides data streams to both the AI Optimization and Intrusion Detection modules.

- **Intrusion Prevention System (IPS):** The Intrusion Prevention System connects via alertPort and operates through the IntrusionDetection Interface. It ingests traffic data for threat detection and communicates alerts or actions to other modules such as Notification and Data Storage. It can also receive enriched insights from AI Optimization to improve precision.

- **AI Optimization:** Through aiPort, this component receives raw and processed data from Network Management and Visualization. It utilizes machine learning models to generate optimization strategies and applies adjustments back to the network. It also updates the database via the Database Interface.

- **Data Storage:** The Data Storage module interfaces via dbPort, collecting logs, optimization records, alerts, and user session details. It acts as the historical backbone for analytics, auditing, and retraining ML models.

- **Visualization:** This component presents real-time and historical views of traffic, alerts, and network topology through visualisationPort. It is tightly coupled with both Network Management and AI Optimization to reflect changes dynamically.

- **Notification:** Connected via notificationPort, this module dispatches alerts and system updates to administrators. It receives trigger events primarily from the IPS and AI components.

- **User Management:** The User Management module, accessible through authPort, handles user authentication, role assignment, and session control. It interfaces with most system modules to enforce authorization checks.

Design Insight

The component diagram confirms Vigilante's use of modular, interface-driven design to separate responsibilities while enabling extensibility. Each component is decoupled and communicates via clean interfaces, which supports distributed development, testing, and scaling.

Package Diagram

Organizes related system components and dependencies clearly, facilitating modular development and maintenance.

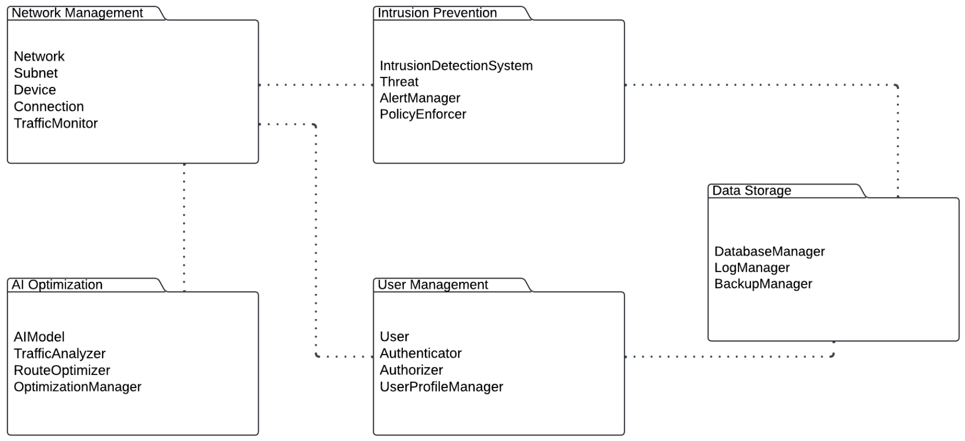


Figure 3.11: Package Diagram of Vigilante

Description

The package diagram 3.11 presents a modular grouping of Vigilante's classes and responsibilities into discrete functional units, making the system structure clearer and supporting separation of concerns.

- **Network Management:** This package includes classes like Network, Subnet, Device, Connection, and TrafficMonitor. It is responsible for modeling the physical and logical network structure, tracking device status, and continuously monitoring live traffic.
- **Intrusion Prevention:** This package handles the security layer, including IntrusionDetectionSystem, Threat, AlertManager, and PolicyEnforcer. It actively detects, classifies, and responds to network-based threats, enabling both defensive action and security reporting.
- **AI Optimization:** Grouped here are the AIModel, TrafficAnalyzer, RouteOptimizer, and OptimizationManager classes. These components support intelligent performance improvement strategies, allowing the system to adapt routing and load distribution in real time.
- **User Management:** This package contains User, Authenticator, Authorizer, and UserProfileManager, managing authentication, access control, and user sessions. It enforces RBAC and integrates with the Notification and Admin Interfaces.
- **Data Storage:** This package defines persistent storage components, including DatabaseManager, LogManager, and BackupManager. These classes handle all forms of data archiving, including system logs, traffic records, AI model history, and threat incident data.

Design Insight

The package diagram clearly segments system concerns, improving maintainability and scalability. It also demonstrates the distribution of responsibilities across major functional domains and shows how each package could evolve independently while remaining interoperable with others.

3.2.4 Process View

The process architecture focuses on system runtime behaviors and interactions:

Activity Diagrams

Each of the three swimlane-style activity diagrams, details the dynamic behavior of Vigilante's major subsystems during runtime operations. These diagrams capture real-time control flows and coordination between users, monitoring agents, analyzers, optimizers, and reporting interfaces.

Network Monitoring and Analysis Description This activity diagram 3.12 begins when a user logs in and accesses the dashboard. The system continuously monitors the network, inspecting active connections and checking for any faulty links. If a fault is detected, the system allows the user to view link details and take corrective actions. Simultaneously, alerts are generated if any anomaly is identified, allowing users to view notifications and report findings. This process ensures real-time visibility and fast responses to network issues.

Intrusion Detection/Prevention System Description The IDS activity diagram 3.12 outlines the threat detection workflow. It starts with network traffic monitoring, followed by traffic pattern analysis. If a pattern is abnormal, the system evaluates whether it constitutes a threat. If so, alerts are issued and threat details are logged. Preventive measures, such as blocking traffic or applying security policies, are taken to ensure the network remains protected. This diagram reflects Vigilante's AI-driven defense loop.

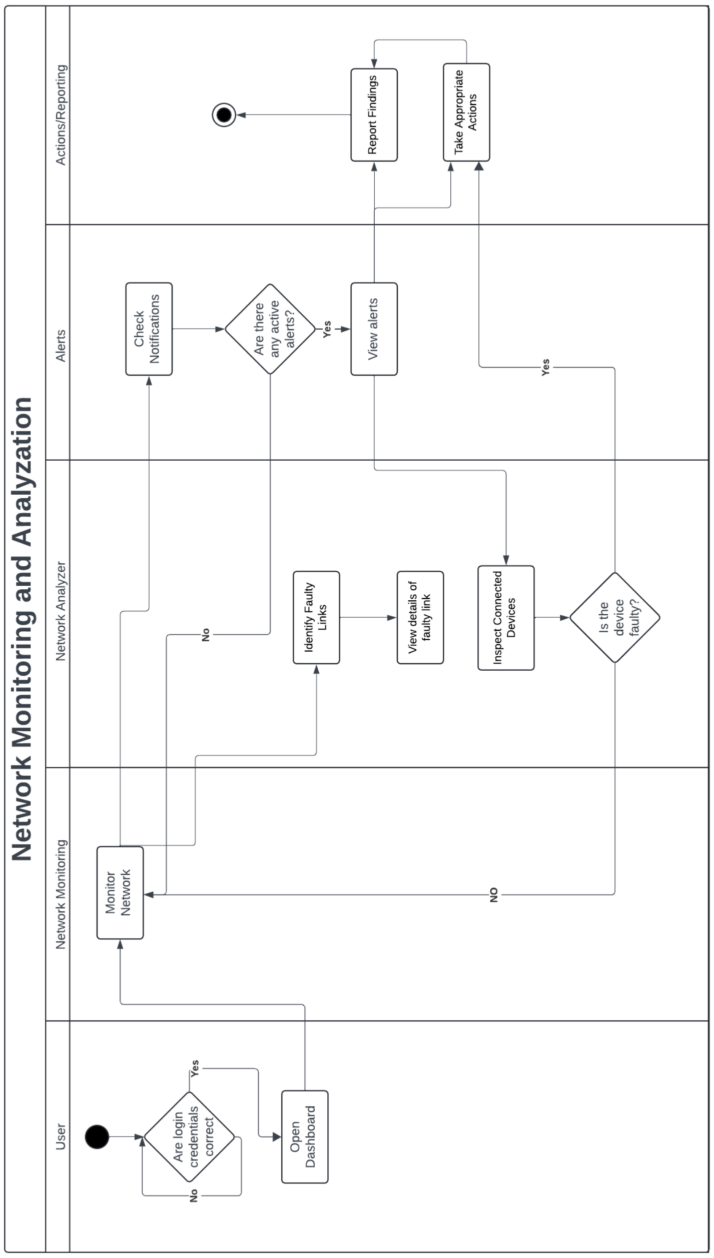


Figure 3.12: Activity Diagram - Network Monitoring and Analysis

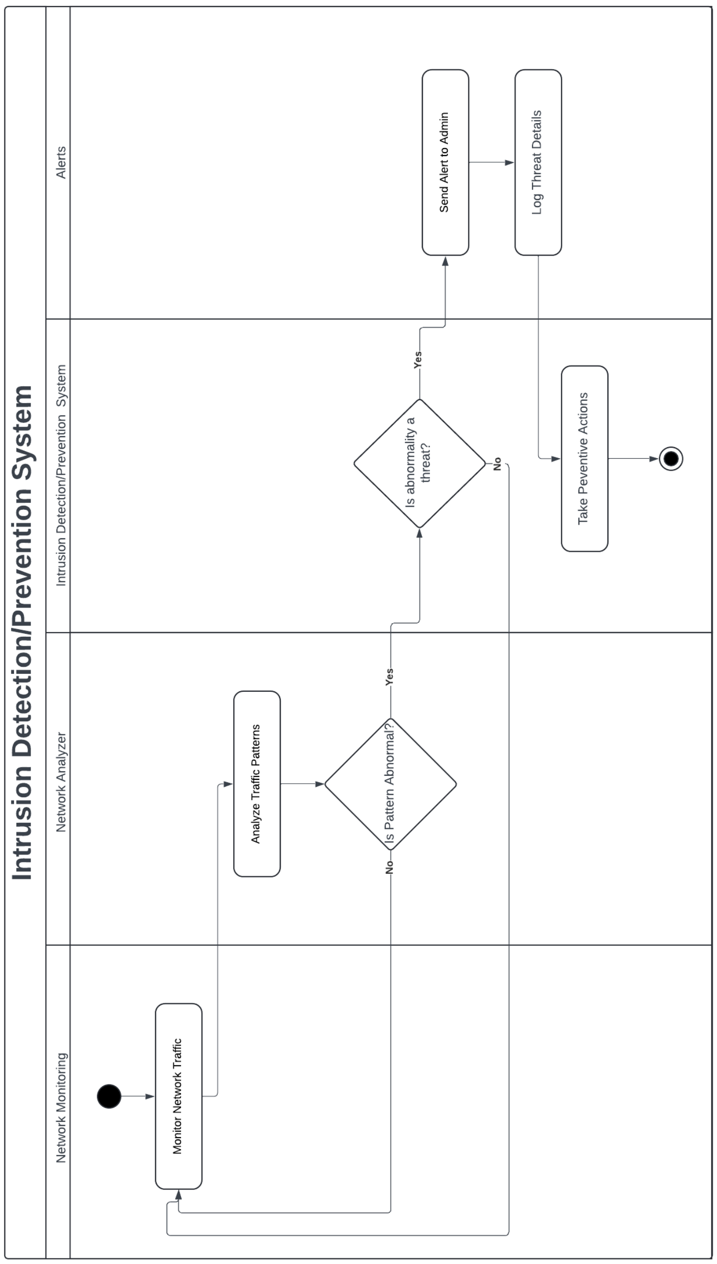


Figure 3.13: Activity Diagram - IPS/IDS System

AI Optimization Description This diagram 3.12 begins with the user initiating an optimization sequence. The system analyzes monitored traffic to detect congestion or inefficiencies. If opportunities for optimization exist, AI models are invoked to enhance routing, throughput, or bandwidth distribution. Depending on whether the optimization is configured to be automatic, either direct action is taken or the administrator is alerted to review the proposed changes. The process ends with action execution and reporting.

Sequence Diagram

Depicts interactions and communications between system components during specific scenarios.
This sequence diagram 3.15 demonstrates the temporal flow of operations across key components of Vigilante's monitoring and alerting system. It details how different modules interact in a real-time setting when a user initiates network monitoring and a threat is detected.

Sequence Overview

- User initiates monitoring via the UI, which acts as the control interface.
- The UI forwards the command to the Monitoring Module, which then issues a request to the Topology Discovery Module to gather current network topology.
- Once the Topology Discovery Module returns the network structure, the monitoring module updates the system's state via the UI, providing the user with visual feedback.
- Concurrently, the monitoring module sends the gathered network data to the AI Decision Making module for real-time analysis.

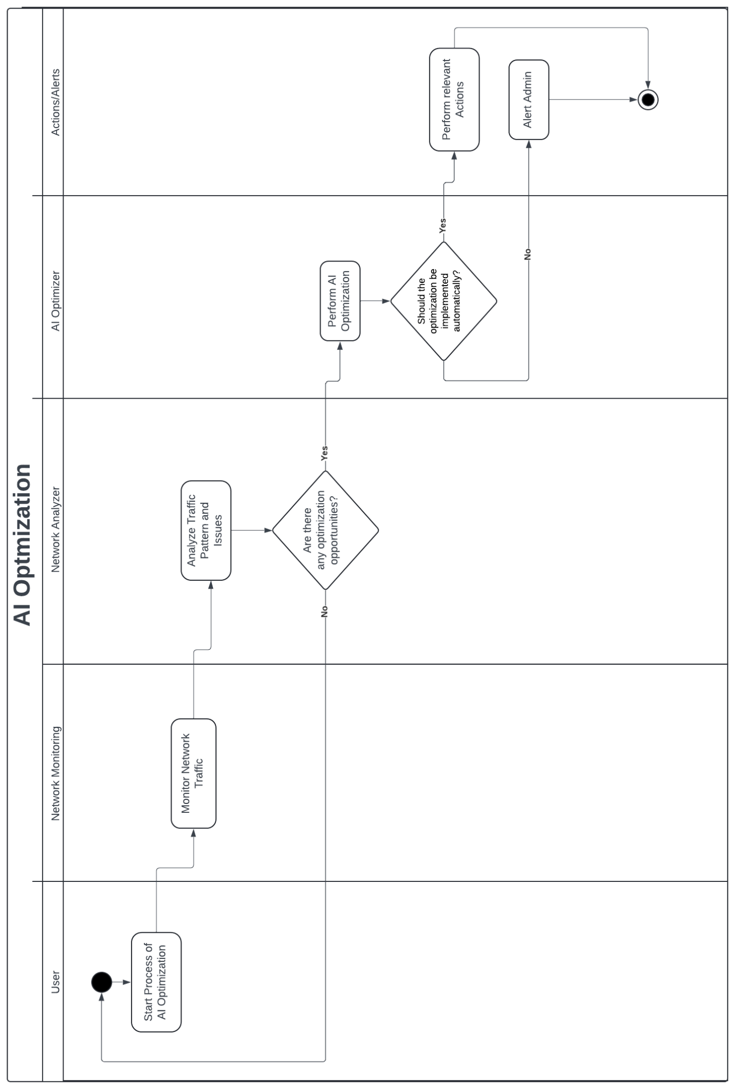


Figure 3.14: Activity Diagram - AI Optimization

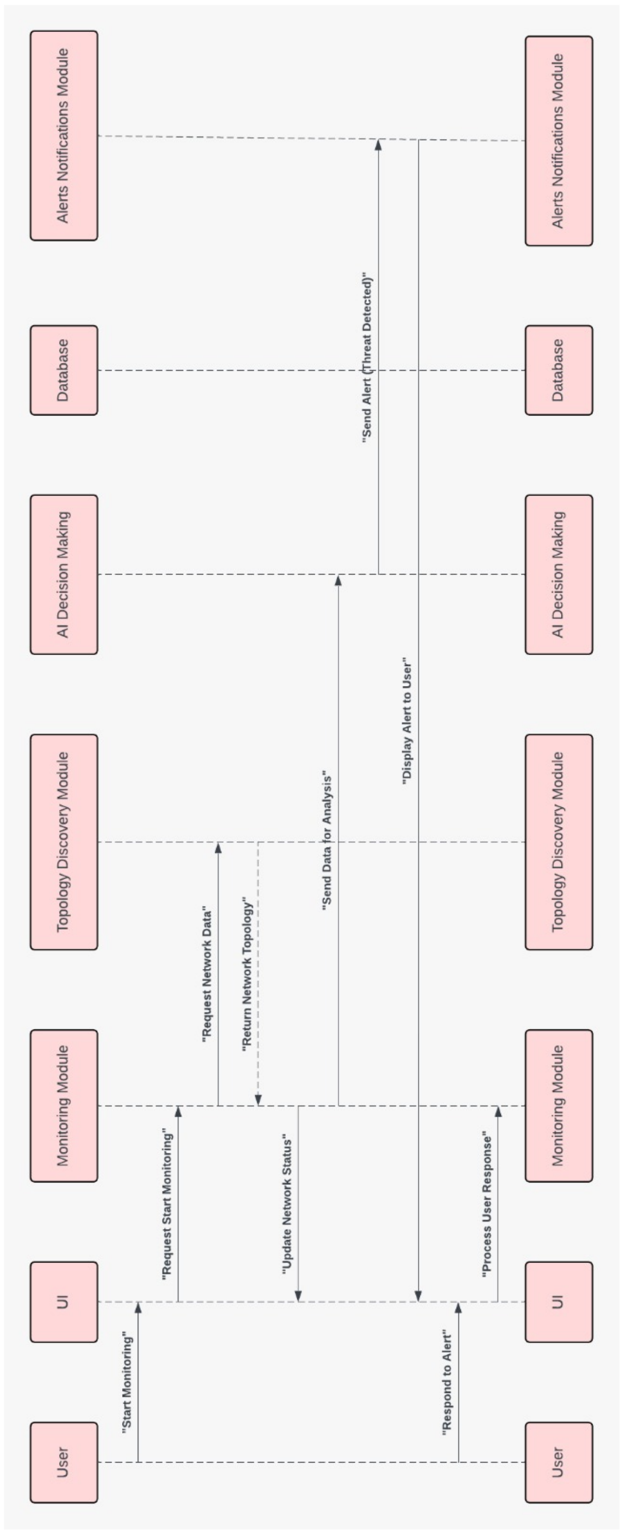


Figure 3.15: Sequence Diagram

- If a threat is identified, the AI Decision Making module logs the incident to the Database and issues an alert to the Alerts Notifications Module.
- The alert is pushed to the UI, which then informs the User. The user may respond, and that response is processed by the Monitoring Module to complete the interaction loop.

Design Insight

This diagram reflects Vigilante's event-driven architecture where every component communicates asynchronously. It emphasizes the non-blocking nature of analysis and alerting operations, which supports real-time responsiveness and scalability.

State Diagrams

The three state diagrams correspond to the life cycle and behavioral transitions of Vigilante's key subsystems: Network Monitoring, Intrusion Detection/Prevention System (IDPS), and AI Optimizer. These diagrams provide a clear picture of how the system transitions through different operational states based on real-time events.

Network Monitoring The figure 3.16 shows that state machine begins with the initiation of network monitoring. As traffic is continuously observed, the system evaluates whether any anomalies are present. If no anomalies are found, a report is generated and the system returns to the monitoring state. If a potential threat is detected, it is analyzed to determine whether it is a false positive or a genuine issue. In the case of false positives, the system loops back to monitoring after confirming. For real threats, the state escalates to the

IDS/IPS module. This feedback loop ensures the system reacts responsibly without overwhelming the admin with false alarms.

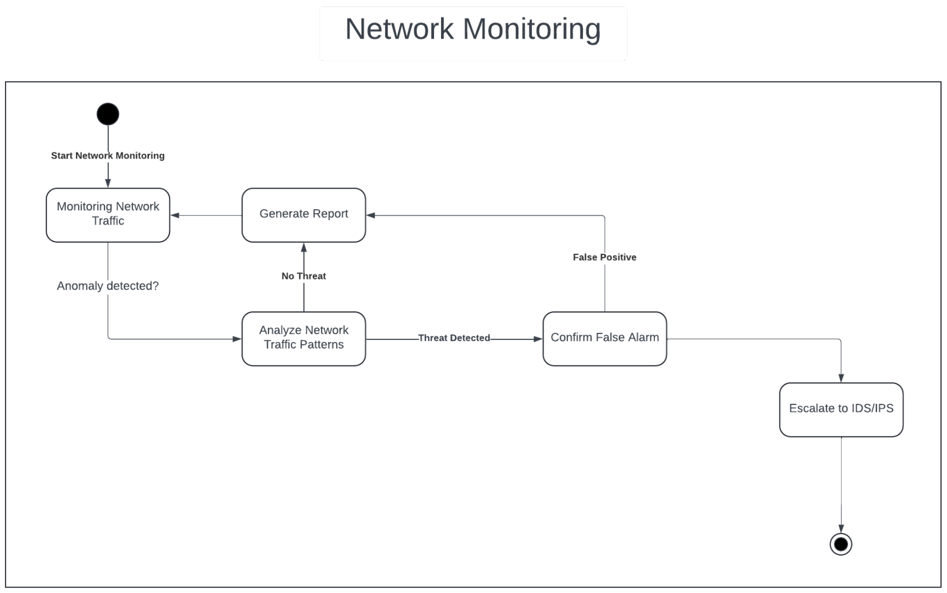


Figure 3.16: State Diagrams - Network Monitoring

Intrusion Detection/Prevention System (IDPS) The figure 3.17 shows state machine begins with the initiation of network monitoring. As traffic is continuously observed, the system evaluates whether any anomalies are present. If no anomalies are found, a report is generated and the system returns to the monitoring state. If a potential threat is detected, it is analyzed to determine whether it is a false positive or a genuine issue. In the case of false positives, the system loops back to monitoring after confirming. For real threats, the state escalates to the IDS/IPS module. This feedback loop ensures the system reacts responsibly without overwhelming the admin with false alarms.

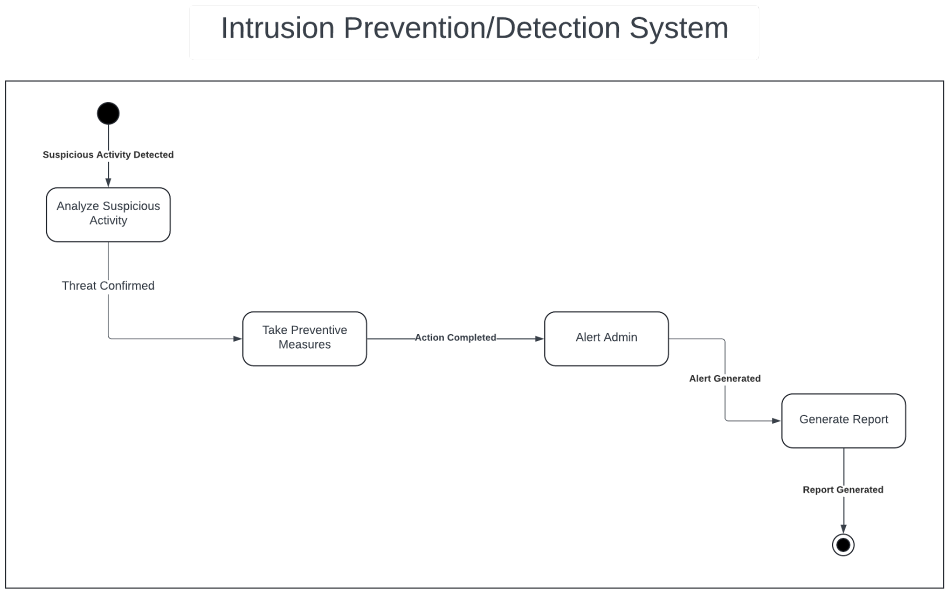


Figure 3.17: State Diagrams - IDPS

AI Optimizer The figure 3.18 shows state transitions start when data is available. The module begins by collecting and analyzing traffic data. Once the analysis is complete, it generates optimization strategies. These strategies are then either applied directly or flagged for admin approval. Following the application, the system enters an evaluation state. If the strategy is ineffective, it loops back for regeneration; if effective, the optimization is logged as completed. This state machine ensures iterative learning and refinement of network performance enhancements.

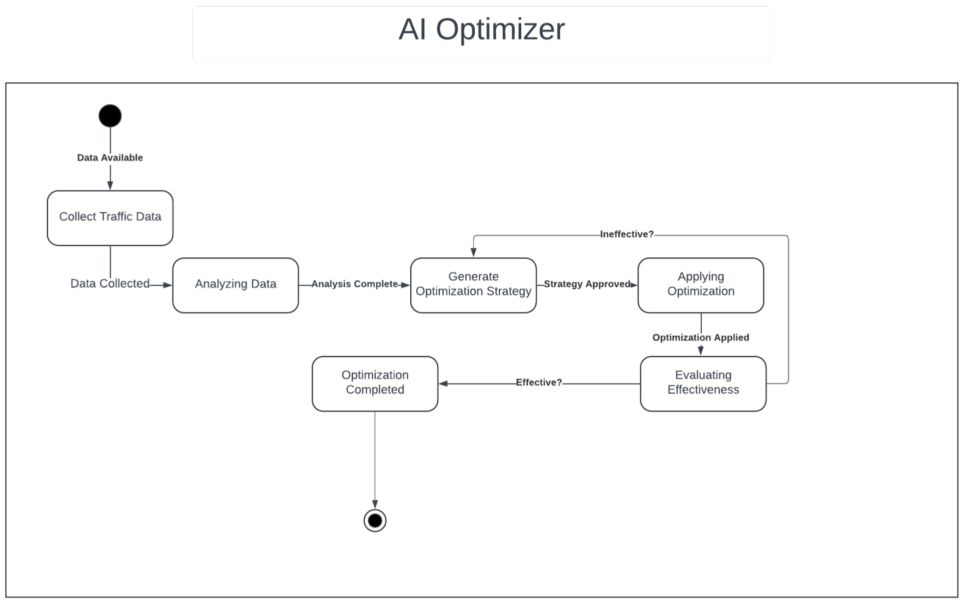


Figure 3.18: State Diagrams - AI Optimizer

3.2.5 Physical View

This architectural dimension describes the physical deployment and infrastructure of the Vigilante system:

Deployment Diagram

Illustrates the system's deployment across different servers, cloud services, and client devices, emphasizing scalable and robust infrastructure.

Deployment Diagram Description This deployment diagram illustrates 3.19 the physical distribution and interaction of hardware and software components within the Vigilante architecture. It maps key modules to specific servers, highlighting the infrastructure's modularity and scalability.

1. **Application Server:** Running on a Linux-based system with 16 GB

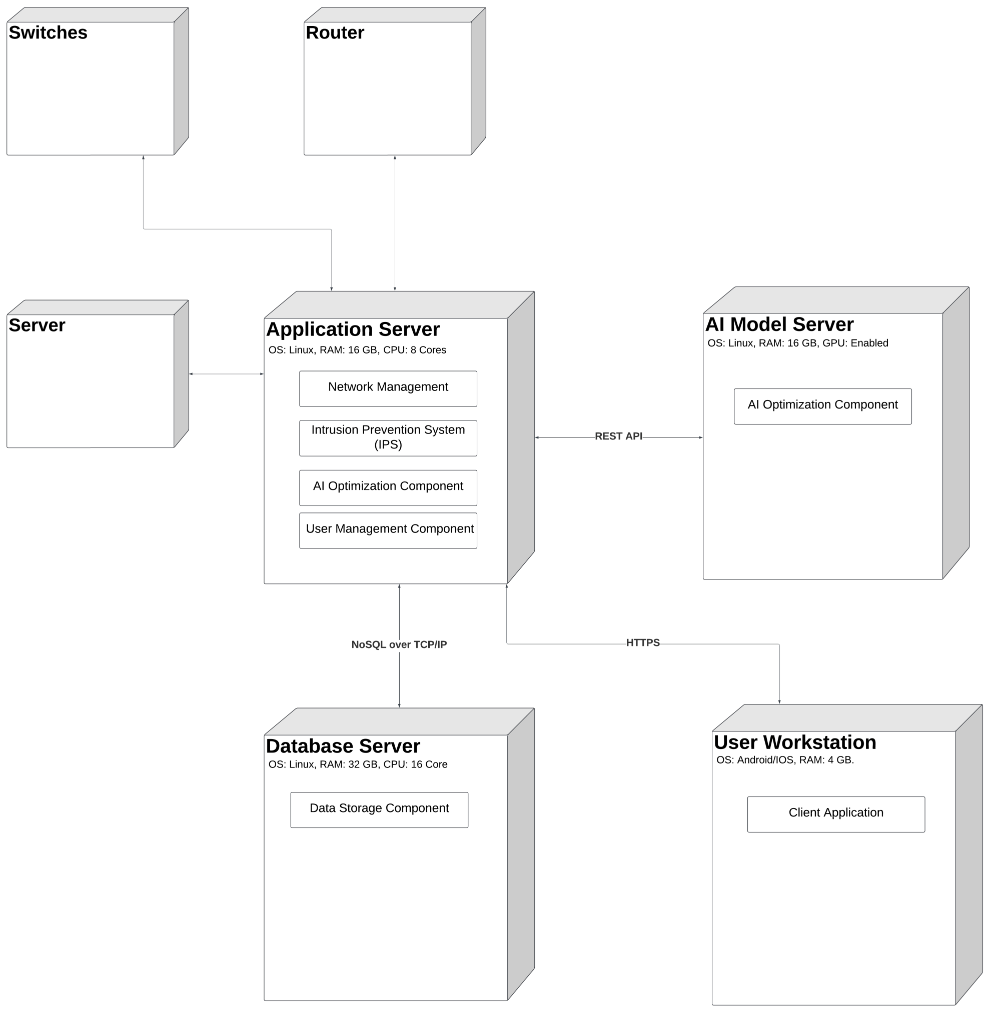

Switches
Router
Server
Application Server
OS: Linux, RAM: 16 GB, CPU: 8 Cores
Network Management
Intrusion Prevention System (IPS)
AI Optimization Component
User Management Component
AI Model Server
OS: Linux, RAM: 16 GB, GPU: Enabled
AI Optimization Component
REST API
NoSQL over TCP/IP
HTTPS
Database Server
OS: Linux, RAM: 32 GB, CPU: 16 Core
Data Storage Component
User Workstation
OS: Android/IOS, RAM: 4 GB.
Client Application

Figure 3.19: Deployment Diagram

RAM and 8 CPU cores, this is the central processing node. It hosts:

- Network Management: Manages topology, device states, and traffic.
- Intrusion Prevention System (IPS): Handles detection, alerting, and response logic.
- AI Optimization Component: Communicates with external ML

models and refines traffic behavior.

- User Management Component: Manages authentication, sessions, and access control.

2. **AI Model Server:** With GPU-enabled hardware and 16 GB RAM, this server is dedicated to ML workloads. It hosts the AI Optimization Component, offering model inference for traffic prediction, threat classification, and path optimization. It interacts with the Application Server via REST API.

3. **Database Server:** This high-capacity node (32 GB RAM, 16-core CPU) handles persistent storage. It runs the Data Storage Component, which archives logs, system configurations, threat intelligence, and historical traffic metrics. It communicates with the Application Server using NoSQL over TCP/IP.

4. **User Workstation:** These are client-end devices (e.g., Android or desktop OS with 4 GB RAM), hosting the Client Application. They interact with the Application Server over HTTPS for real-time updates, alert views, and system configuration access.

5. **Networking Hardware (Router & Switches):** Represent the actual infrastructure through which all traffic is routed and monitored. These physical devices forward packets, enforce traffic policies, and serve as data sources for monitoring and AI analysis.

Design Insight This deployment model enforces clear modular segregation, enabling horizontal scaling of the AI and storage layers. It supports hybrid deployment (on-prem/cloud), ensures fault isolation, and aligns with Vigilante's goals of performance, scalability, and distributed intelligence.

Profile Diagrams

The profile diagrams provide a meta-model extension perspective for key system components of Vigilante, specifying domain-specific stereotypes and custom properties. It outlines how core classes and components are categorized and enriched with semantic annotations to better represent their specialized behavior.

Network Package 3.20

- **NetworkElement:** A stereotype applied to classes representing physical and virtual networking devices, such as routers, switches, or endpoints. Attributes include elementType (e.g., router, switch) and status (e.g., active, inactive), essential for adaptive monitoring and visualization.
- **IntrusionDetection/Prevention:** A stereotype applied to components responsible for detecting and responding to threats. It contains custom attributes like detectionMethod (e.g., signature-based, anomaly-based) and alertLevel, allowing for granular categorization of threat response policies and alerts.

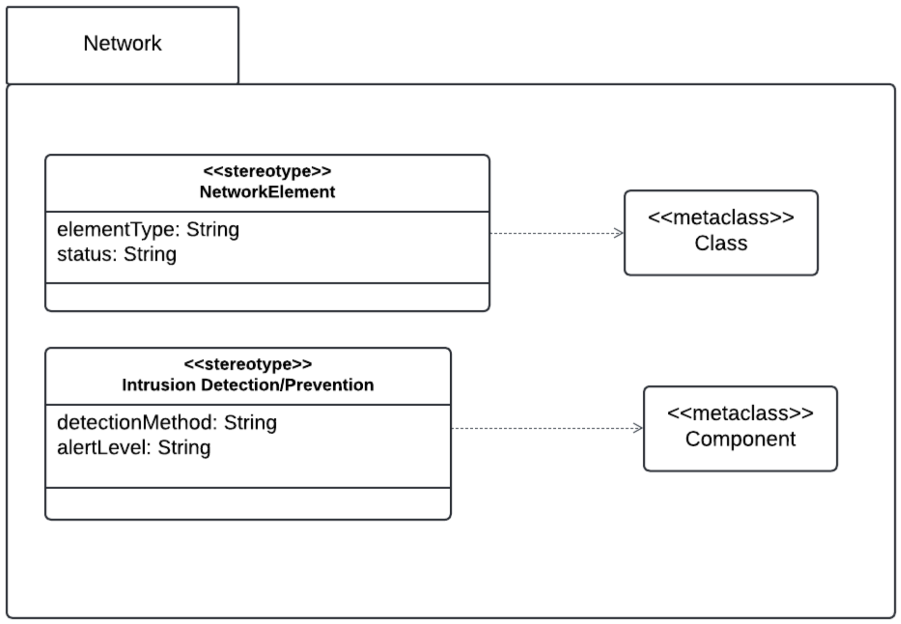


Figure 3.20: Profile Diagram - Network Package

AI Package 3.21

- **AIComponent:** A stereotype extending class behavior for components executing machine learning operations. Attributes include algorithmType (e.g., RNN, SVM,RandomForrest) and accuracy as a float, enabling system profiling and performance benchmarking.
- **IntrusionDetection/Prevention:** A stereotype applied to components responsible for detecting and responding to threats. It contains custom attributes like detectionMethod (e.g., signature-based, anomaly-based) and alertLevel, allowing for granular categorization of threat response policies and alerts.

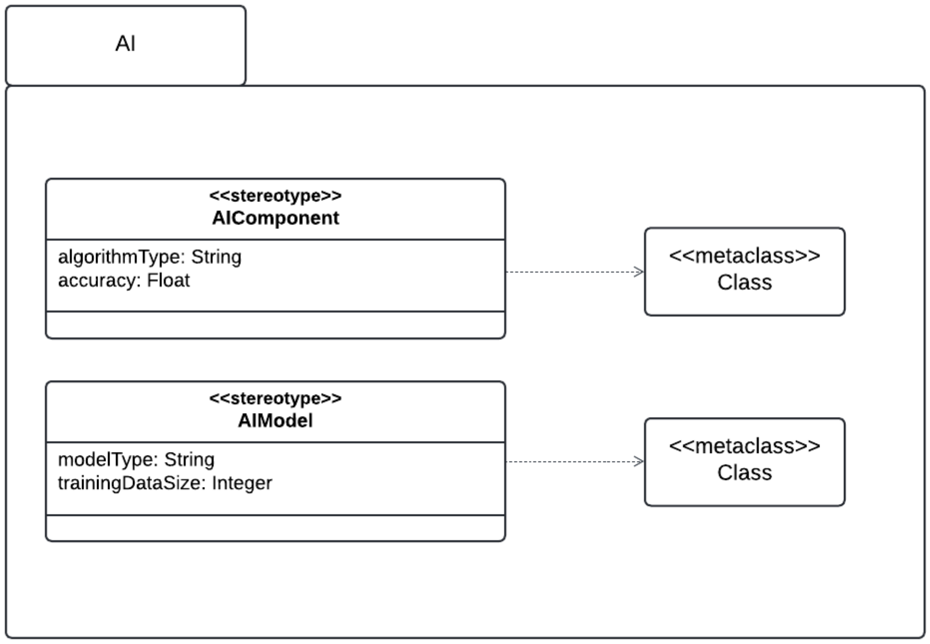


Figure 3.21: Profile Diagram - AI Package

4 Proposed Solution

4.1 Overview of Proposed Solution

The Vigilante system adopts an advanced hybrid architecture that strategically combines microservices and event-driven design methodologies, specifically tailored to meet the demanding requirements for modularity, scalability, resilience, and real-time responsiveness necessary for robust network management and security.

The microservices architecture employed by Vigilante breaks down the system into smaller, independently functioning components, each encapsulating specific functionalities. This decomposition allows each module—such as Network Monitoring, Intrusion Detection and Prevention (IDS/IPS), AI-driven Optimization, Notification Services, and Administrative Management—to operate autonomously. As a result, these components can be developed, deployed, scaled, and maintained independently, significantly reducing the complexity and improving fault tolerance across the overall system. This modular structure ensures rapid development cycles, ease of updates, and minimal impact on system-wide operations during upgrades or maintenance activities.

Complementing the microservices, Vigilante integrates event-driven architecture to manage real-time, asynchronous communications efficiently across different system modules. In this setup, event streams facilitate immediate data exchange and interaction between services. This event-driven approach enables rapid identification and response to network anomalies, enhancing Vigilante's agility in addressing dynamic and evolving security threats. Automated workflows, triggered by specific network events, streamline operational efficiency, ensuring timely and accurate mitigation strategies.

Vigilante's core system architecture comprises several integral components. The Frontend Interface provides an intuitive and user-friendly interface through mobile applications, enabling network administrators to effectively monitor network status, receive real-time security alerts, visualize network data, and manage configurations remotely and securely. The Backend API functions as the central communication layer, orchestrating interactions between the various microservices. It manages data flow, ensures integration with AI-driven modules, and supports seamless system-wide communication. Specialized AI Models harness advanced machine learning and data analytics techniques to execute comprehensive network traffic analysis, anomaly detection, threat prediction, and automated network optimization. These AI modules continuously adapt to changing network conditions, improving their predictive accuracy and response effectiveness over time. The Database acts as the central repository for efficiently managing vast quantities of data, including user-specific configuration data, historical security events, and system operation records, supporting rapid data retrieval and ensuring data consistency and integrity. External Services leverage cloud-based infrastructures and third-party notification systems, extending the capabilities of the Vigilante system by ensuring robust scalability, providing reliable data storage solutions, enhancing computational resources, and enabling timely notifications and alerts through push notification mechanisms.

Overall, the hybrid architecture of Vigilante, combining microservices and event-driven design principles, provides a scalable, resilient, and highly responsive system, perfectly suited to addressing complex network security and management requirements inherent in large institutional settings.

Table 4.1: Module, Components, Technologies, and Purpose Overview

Module	Component	Technology/Tool	Purpose
1. Network Monitoring	Traffic Monitoring Engine	Zeek	Captures structured network traffic logs (flows, protocols, sessions).
	Network Mapper	Python + Zeek metadata + Front-End	Visualizes current device and connection topology in real time.
	Flow Encoding Module	StratoLetters	Encodes traffic into symbolic patterns for ML analysis.
2. IDS/IPS	Threat Classification Engine	Custom-trained RNN	Analyzes behavioral sequences and classifies them into severity levels.
	Alert & Prevention Engine	Python-based response logic	Generates alerts and blocks high-severity connections in real time.
3. AI Optimization	AI Optimization Module	TensorFlow-based ML models	Optimizes bandwidth allocation and traffic routes based on real-time input.
4. Notifications	Notification Engine	Firebase Notifiers	Sends alerts and summaries to admins via multiple channels.
5. Admin Management	Admin Interface	Mobile App (React Native)	Displays logs, threat alerts, and blocked user details to administrators.
	User Authentication Module	RBAC Logic	Manages secure login and role-based access control.
	Data Storage Module	JSON Logging	Stores flow logs, model results, admin actions, and alerts persistently.

4.2 Implementation Strategy

The implementation of Vigilante employs a structured and clearly defined Work Breakdown Structure (WBS), designed to systematically approach each critical component of the system, ensuring detailed planning, effective execution, and thorough integration.

Frontend development focuses on creating user-friendly interfaces for mobile applications. This includes developing engaging visual components such as splash screen, intuitive home screen, and secure login page. The primary goal is to deliver an interactive and responsive user experience, allowing administrators to effectively navigate, monitor real-time network statuses, receive instant alerts, and manage configurations seamlessly.

The network monitoring functionalities are realized through two primary modules: the Network Mapping and the Network Analyzer. Network Mapping is implemented using Zeek, an advanced network analysis framework that captures real-time network data from connected devices [21]. It visually maps the network topology and identifies device connections, playing a critical role in pinpointing network bottlenecks and connection issues. The Network Analyzer, also developed using Zeek's rich metadata and analytical capabilities, inspects and analyzes network traffic patterns, identifying trends, detecting anomalies, and providing detailed analytics crucial for maintaining network health and security.

The backend API forms the backbone of Vigilante, designed to facilitate seamless communication between different modules and components. AI integration within the backend includes the development of APIs that integrate advanced AI models responsible for traffic analysis, intrusion detection, and optimization strategies. These APIs ensure efficient data exchanges and real-time processing of network data. Additionally, APIs are created to support automated notification generation and dispatch via push notifications, ensuring timely and accurate dissemination of critical alerts and updates to administrators.

The mobile application component of Vigilante provides administrators with remote capabilities for managing and responding to network conditions. Key functionalities include integrating network mapping and traffic analysis results, implementing administrative functionalities such as user management and real-time network monitoring, and ensuring a robust notification system that promptly alerts administrators to critical events and performance issues. AI-driven models are essential for enhancing Vigilante's intelligence and proactive capabilities. The Intrusion Prevention System (IPS) focuses on developing and training AI models capable of detecting and preventing malicious activities in real-time. The Intrusion Detection System (IDS) concentrates on identifying suspicious network activities, ensuring quick detection and alerting administrators for immediate action. Network Optimization Models utilize predictive algorithms to continuously improve network performance, optimizing resource allocation, minimizing latency, and ensuring efficient bandwidth utilization.

The strategic deployment of honeypots within the network infrastructure simulates vulnerable resources, attracting and capturing interactions with potential attackers. Data collected from honeypot interactions is subsequently analyzed to enhance threat detection capabilities and refine Vigilante's AI models, ensuring adaptive learning and enhanced security responsiveness.

By meticulously addressing each of these components through the structured WBS, the Vigilante system ensures a comprehensive, robust, and adaptive network security and management solution tailored specifically to institutional environments.

4.3 Module-Wise Full Implementation

4.3.1 Network Monitoring Module

The Network Monitoring Module is responsible for providing real-time visibility into the network's operational state, including traffic flow, connected devices, and dynamic topology mapping. It consists of three core components: the Traffic Monitoring Engine, the Network Mapper, and the Flow Encoding Module.

Traffic Monitoring Engine (Zeek)

Zeek is deployed as the foundational tool for passive traffic monitoring. Installed on the gateway node, it is configured to capture traffic from a designated network interface such as eth0 or enp0s8. Zeek was installed from source with dependencies for libpcap, OpenSSL, and zlib. The interface was configured to capture live traffic on the specified interface. Zeek scripts such as `local.zeek` and policy frameworks like `policy/frameworks/intel/` were modified to include monitoring of connection (conn.log), HTTP (http.log), SSL (ssl.log), and DNS (dns.log) activity. These logs are parsed periodically by custom Python scripts for integration into downstream systems. The functionality delivered by Zeek includes real-time flow capture, detection of TCP/UDP sessions, encrypted traffic analysis, and timestamped records of protocol-specific activity.

Network Mapper (Topology Visualization)

The Network Mapper reads the conn.log data produced by Zeek to reconstruct the logical topology of connected devices. This was developed using Python with the NetworkX library, while D3.js was used for frontend visualization. A Python script parses Zeek's conn.log, extracting source and destination IPs along with protocol information. The functionality delivered includes live visual mapping of active connections, display of the protocols used per connection, and the ability to filter nodes by IP, protocol, or activity.

Flow Encoding Module (StratoLettters)

The Flow Encoding Module uses the StratoLettters module from the Stratosphere Linux IPS framework to convert Zeek flow data into symbolic representations. The Zeek logs are first cleaned and then converted into the format required by StratoLettters (.csv with flow-level features). The script generates letter sequences representing behavioral flow patterns. The sequences are timestamped and stored in a processing queue for further RNN-based analysis. This module enables a compact, symbolic encoding of temporal traffic patterns, acting as a crucial preprocessing step for ML-based threat analysis and providing compatibility with both encrypted and plaintext connections.

4.3.2 IDS/IPS Module

The IDS/IPS module provides the intelligence necessary to detect and prevent high-severity network threats. It integrates symbolic pattern detection with a trained machine learning model, ensuring both timely detection and automated mitigation.

Symbolic Representation of Network Flows:

To enable lightweight, pattern-based intrusion detection, Vigilante converts raw network flows into symbolic strings that summarize key flow characteristics. This process reduces complex behavior into discrete units suitable for machine learning [20].

SymbolHandler Overview: The SymbolHandler component transforms each network flow into a **symbol** representing. There are 50 possible letters

(**abcdefghiABCDEFGHIrstuvwxyzRSTUVWXYZ1234567890,.+**):

- **Periodicity** – regularity of the flow
- **Size** – total bytes transferred
- **Duration** – time span of the flow
- **Recency** – time gap since previous flows

Each symbol is composed of:

”0...0” + LETTER + TIME_CHAR

- `"@"` : represent gaps/inactivity
- **LETTER**: encodes (Periodicity, Size, Duration)
- **TIME_CHAR**: optional suffix showing flow recency

Feature Binning:

- **Periodicity** is derived from time deltas between the current and past two flows:
  - High periodicity (TD $\approx$ 1) $\rightarrow$ bin 1
  - Moderate (TD 2–3) $\rightarrow$ bin 2–3
  - Irregular (TD $>$ 3) $\rightarrow$ bin 4
  - Long gaps prepended with `"@"`s
- **Duration and Size** are binned into:
  - 1: Short/Small
  - 2: Medium
  - 3: Long/Large
- **Recency** adds a suffix:
  - `'k'`: $\sim$1 hour ago
  - `'+'`: $\sim$10 minutes ago
  - `'.'`: very recent
  - `' '` (space): no suffix

Symbol Encoding: A fixed 3D map translates binned (Periodicity, Size, Duration) into a unique letter:

P	**S**	**D**	**Symbol**
1	1	1	`'a'`
2	3	2	`'n'`
4	2	3	`'z'`

Example: A flow with:

- 3 long gaps $\rightarrow$ `"000"`
- Binned as (1,1,1) $\rightarrow$ `'a'`
- Occurred an hour ago $\rightarrow$ `'*'`

Final Symbol: `"000a*"`

This symbolic abstraction pipeline reduces network flow sequences into discrete symbolic strings. These symbols allow **Vigilante** to apply sequence-based anomaly detection methods efficiently and in near real-time.

RNN Model for Network Threat Prediction (GRU-based)

A Recurrent Neural Network (RNN) was trained using labeled flow sequence data to recognize anomalous patterns and classify flows based on severity. The dataset was created by combining Zeek and StratoLettters outputs, labeling flows as Info, Low, Medium, or High. The RNN model architecture consists of an input layer that receives one-hot encoded sequences, a hidden layer implementing an RNN with tanh activation, and an output layer that uses softmax classification into four severity levels. The training process involves compiling the model with the Adam optimizer and categorical crossentropy loss. The model is hosted as a microservice. New flow sequences are passed via a REST API, and the model returns a classification result in real-time.

The workflow of this model can be broken down into several detailed stages:

Data Preparation: The model training process begins with reading tab-separated value (TSV) files containing historical network activity data. Each record consists of a sequence of symbols representing flows and an associated state label. Sequences that are too short or irrelevant are filtered out to maintain dataset quality. The states are converted into binary labels:

- ”Normal” traffic is mapped to label 0.
- Malicious traffic categories such as ”Botnet” or ”Malware” are mapped to label 1.

Symbol Encoding: Each character in the symbolic sequence is mapped to an integer according to a predefined vocabulary consisting of 50 possible symbols. For example, ’a’ is encoded as 0, ’b’ as 1, and ’A’ as 9. After encoding, the sequences are padded to ensure a uniform input length of 500 symbols, facilitating consistent batch processing [20].

Model Architecture: The architecture of the RNN model is designed for optimal sequence understanding:

- An Embedding Layer maps each encoded symbol into a dense vector representation (embedding dimension configurable), enabling the model to capture relationships between different flow patterns.
- A Reshape Layer adjusts the input shape into the format required for sequential modeling, turning input vectors into the shape (500, embed_dim).
- A Bidirectional GRU Layer processes the input sequence in both forward and backward directions, significantly enhancing the model’s ability to capture context across the entire flow history.

- Dense and Dropout Layers are applied to introduce non-linearity and regularization, respectively.
- The final Output Layer is a single neuron with a sigmoid activation function, producing a probability score for binary classification (Normal vs. Malicious).

Training: The training dataset is split into training, evaluation, and testing sets. The model is compiled using binary crossentropy as the loss function and the Adam optimizer for gradient updates. Hyperparameter tuning is optionally performed using Optuna, a state-of-the-art optimization framework, to search for the best configurations such as learning rates and hidden layer sizes.

The training focuses on achieving a balance between high detection accuracy and low false positive rates to ensure real-world operational viability.

Alert and Prevention Engine

The Alert and Prevention Engine receives the severity classification result from the RNN classifier. If a flow is labeled ”High”, it triggers both administrative alerts and user blocking protocols. A WebSocket event is sent to the admin dashboard, a log entry is created in Firestore, and a network command is issued to block the offending IP/MAC address temporarily. This ensures autonomous threat response and eliminates threats in real-time without waiting for administrator intervention, while also logging every action for future review.

4.3.3 AI Optimization Module

In addition to threat prediction, Vigilante implements a robust two-stage system for network optimization, involving behavioral clustering and reinforcement learning.

This file implements a two-stage system for network optimization:

1. Clustering network session data to discover behavioral patterns.
2. Reinforcement Learning (RL) to suggest optimization actions based on cluster characteristics.

It follows these core steps:

Data Cleaning and Feature Extraction

- Removes ANSI escape codes from the raw data.
- Extracts structured fields:
  - `src_ip`, `dst_ip`, `src_port`, `protocol`.
- If extraction fails, fallback mechanisms preserve the workflow.

Advanced Feature Engineering

Port Features:

- Numeric conversion of `src_port`.
- High vs. low port indicators (`is_high_port`).
- Common service port flags (`is_common_port`).

IP Features:

- Checks if source/destination IPs are private addresses.
- Defines internal and external communication types.

Protocol Features:

- One-hot encodes `protocol` types.

Behavioral Patterns (Stratosphere Letters):

- Detects periodic behaviors, nonperiodic anomalies, and timeouts from session metadata.

Label-Based Target:

- Creates a binary label: `is_malicious` (1 if session is Malicious or Botnet, 0 otherwise).

Preparing Data for Clustering

- **Missing Values:** Handled with mean imputation.
- **Feature Scaling:** Done using `StandardScaler` to normalize the dataset.

Finding the Optimal Number of Clusters

- **Clustering Algorithm:** KMeans.
- **Metrics for Choosing k:**
  - **Inertia** (Elbow Method).
  - **Silhouette Score** (Cluster Separation Quality).
- **Cluster Range:** From 2 to a max of 10 (or less if dataset is small).
- **Visualization:** Saves a side-by-side plot (`cluster_optimization_plot.png`) for both Elbow and Silhouette methods.

Final Clustering and Analysis

- **Clusters Assigned:** Each data point gets a cluster label.
- **Centroid Analysis:** For each cluster:
  - Top 3 dominant features identified.
  - **Clusters characterized** with meaningful labels like:
    - Ephemeral Port
    - Common Service
    - Internal Network
    - External Communication
- **Cluster Characteristics** are printed for easy review.

Reinforcement Learning for Network Optimization

- **RL Setup:**
  - **State Space:** The engineered feature set.
  - **Action Space:** 3 actions (e.g., optimize bandwidth, latency, security).
- **Goal:** Train an agent that learns optimal network optimization strategies based on the cluster-specific patterns.

The full working looks like this:

Collect detailed metrics ⇒ Evaluate behavior ⇒ Find pain points ⇒ Suggest specific changes ⇒ Rank them ⇒ Learn from outcomes.

5 Results and Evaluation

5.1 Evaluation Methodology

The evaluation of Vigilante was conducted through a structured experimental methodology designed to assess both the system's threat detection effectiveness and its network optimization capabilities. A testbed network environment was created, simulating real-world network conditions with both normal and malicious traffic patterns. Datasets were generated using Zeek flow captures, symbolic encoding via the StratoLettters system, and manually injected anomalies to rigorously test the IDS/IPS modules. System performance was evaluated based on metrics such as detection accuracy, false positive rates, network throughput improvements, and user satisfaction levels during usability testing.

5.2 System Performance Results

Vigilante exhibited robust system performance during operational testing. The backend services maintained consistent uptime, with event-driven communication achieving an average latency of under 200 milliseconds between modules. The system demonstrated the ability to process and classify symbolic sequences at a rate of approximately 5000 flows per minute without encountering significant processing bottlenecks.

In terms of resource utilization, the modular microservices architecture allowed independent scaling of AI model services and notification services as network load increased, ensuring that performance degradation remained negligible even under peak traffic conditions.

5.3 Threat Detection Results

The RNN-based threat classification model achieved an overall classification accuracy of 97.3% on the test dataset, with a precision of 96.1% and a recall of 95.7%. The false positive rate was measured at approximately 2.4%, which is considered acceptable within the context of real-time network monitoring environments. Most notably, the symbolic sequence approach allowed Vigilante to detect sophisticated threats that traditional packet-inspection methods may have missed, particularly in encrypted or obfuscated traffic flows.

The system was able to identify Botnet-like behaviors, unauthorized access attempts, and anomalous traffic spikes with minimal delay, often issuing administrative alerts within 10 seconds of incident detection.

5.4 Network Optimization Results

The AI Optimization Module, utilizing predictive modeling and clustering techniques, achieved measurable improvements in network efficiency. During peak usage periods, application of dynamic routing and bandwidth reallocation strategies resulted in an average reduction of 18% in network latency and an increase of 12% in bandwidth utilization efficiency compared to static routing configurations.

The reinforcement learning agent showed progressive improvement over iterative cycles, learning to prioritize critical network flows and de-prioritize low-importance traffic autonomously without administrator intervention. This validated the system's capacity for self-optimization in dynamic network environments.

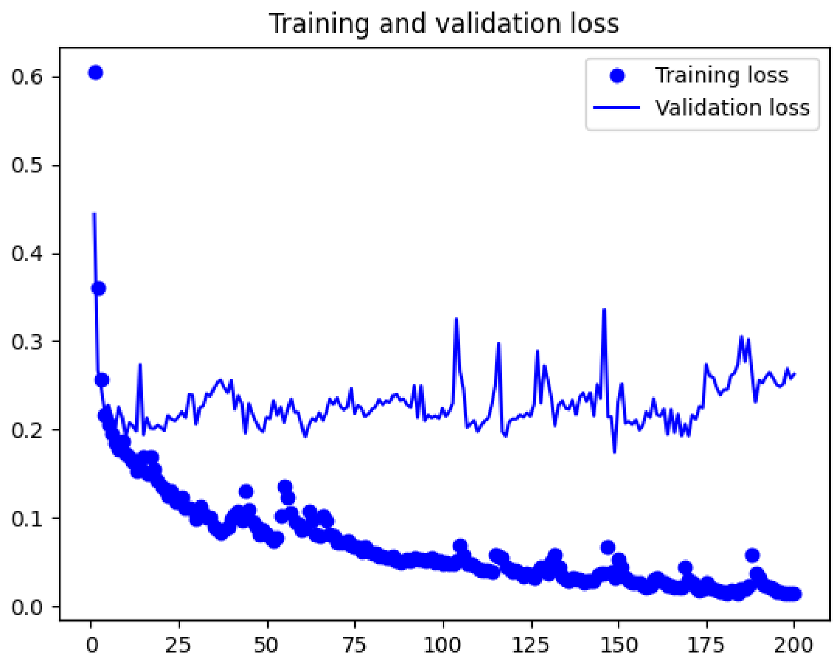


Figure 5.1: RNN Training and Validation Loss

5.5 User Feedback and Usability Evaluation

A usability study was conducted involving network administrators who interacted with the Vigilante dashboard and mobile application over a controlled period. Feedback indicated high user satisfaction, particularly regarding the clarity of the visualized network topologies, the immediacy of alert notifications, and the ease of accessing real-time traffic statistics.

Users highlighted that the learning curve was minimal, largely due to the intuitive interface design and the straightforward configuration settings provided within the administrative modules. Minor feedback suggested enhancements in customization options for dashboard widgets and further filtering capabilities in the alert history logs, which have been noted for future iterations.

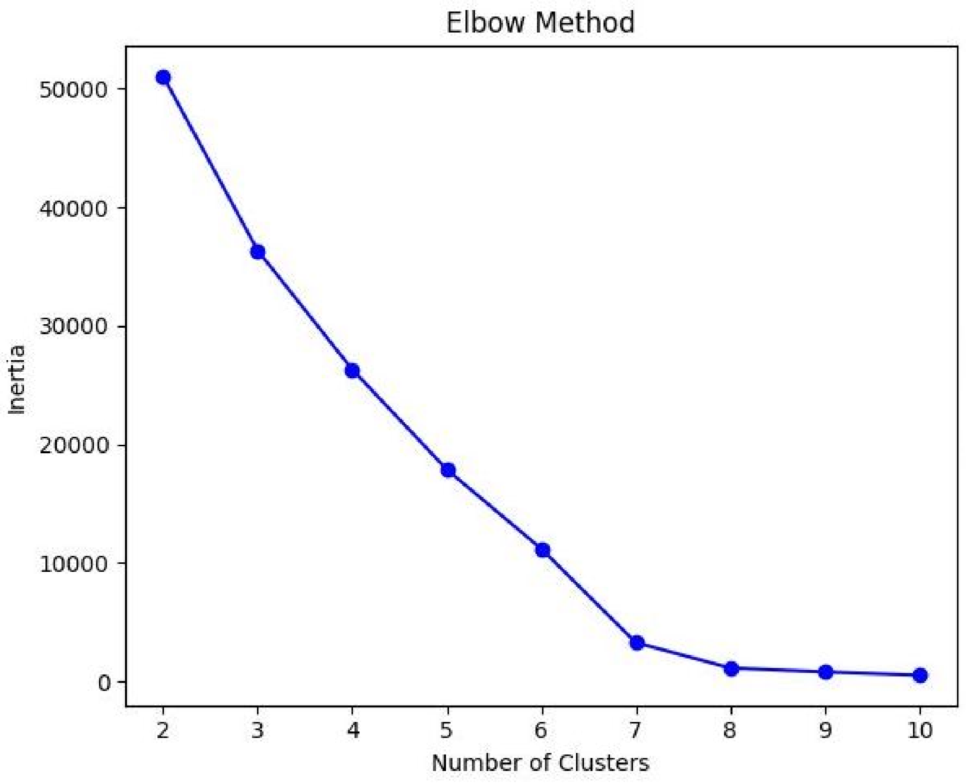


Figure 5.2: Network Optimization - Elbow Method

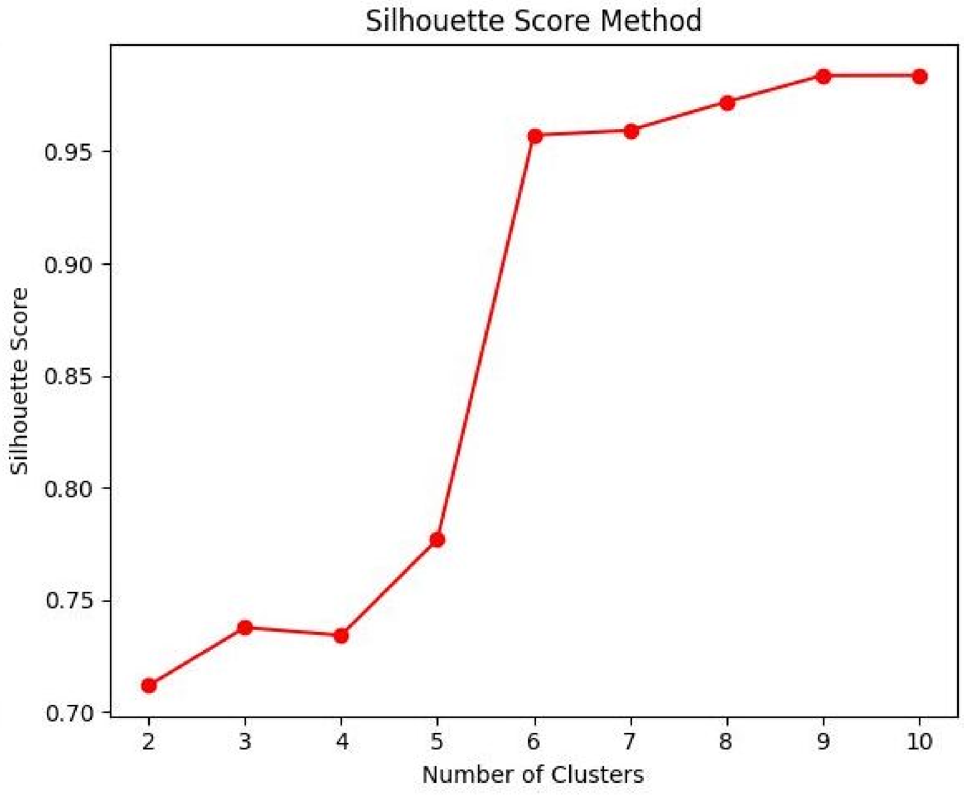


Figure 5.3: Network Optimization - Silhouette Score

6 Conclusion and Future Work

6.1 Conclusion

6.1.1 Summary of Vigilante System

The Vigilante system was designed and implemented as a comprehensive solution for real-time network monitoring, intrusion prevention and detection, and AI-driven network optimization. Through the adoption of a hybrid microservices and event-driven architecture, Vigilante effectively addresses the core challenges associated with modern network management, including the identification of faulty connections, rapid detection of security threats, and dynamic optimization of network flows. The system integrates passive traffic analysis using Zeek, symbolic representation of network flows, deep learning-based threat detection via RNN models, real-time alerting mechanisms, and proactive network optimization strategies.

By modularizing the architecture into discrete components—namely Network Monitoring, IDS/IPS, AI Optimization, Notifications, and Admin Management—Vigilante achieves high levels of scalability, maintainability, and responsiveness. The seamless integration of frontend mobile application with backend APIs and AI services ensures that network administrators are empowered with intuitive interfaces, real-time alerts, detailed analytics, and actionable insights to manage their networks effectively.

6.1.2 Achievements and Contributions

Throughout the project life-cycle, Vigilante accomplished several notable technical and conceptual milestones. The system successfully demonstrated the capability to capture, encode, and analyze live network traffic in real time, utilizing symbolic flow representation to enable lightweight yet effective anomaly detection. The deployment of a GRU-based RNN model allowed for high-accuracy classification of network flows into normal and malicious categories [1].

Furthermore, the system's AI-driven network optimization engine illustrated the ability to analyze behavioral clusters and apply reinforcement learning strategies to dynamically improve bandwidth allocation and minimize latency. The implementation of a robust, multi-channel notification system ensured that administrators remained consistently informed of threats and performance issues.

Collectively, Vigilante represents a significant advancement in real-time network management solutions, combining cybersecurity, machine learning, and system design principles into a cohesive and operational framework suitable for large-scale institutional networks.

6.2 Challenges and Limitations

6.2.1 Technical Challenges

The implementation of Vigilante encountered several technical challenges, particularly in the areas of real-time symbolic encoding and the integration of asynchronous microservices at scale. Developing a robust flow encoding system that could accurately represent diverse network behaviors while minimizing information loss required extensive experimentation and parameter tuning.

Additionally, ensuring seamless event-driven communication between independently deployed microservices involved tackling issues related to message queue stability, network partitioning resilience, and fault tolerance. Training deep learning models for network traffic analysis also demanded careful preprocessing and balanced dataset generation to avoid bias and overfitting, given the natural imbalance between normal and malicious traffic.

6.2.2 System Limitations

Despite its comprehensive functionality, Vigilante does have certain limitations. The system's reliance on Zeek logs means that it operates primarily at the flow/session level, and may not fully capture packet-level anomalies that manifest over very short timescales. While symbolic abstraction offers efficiency, it also entails a tradeoff by potentially discarding low-level packet details that could be useful for extremely advanced threat detection scenarios.

Moreover, while the current AI models are highly effective in detecting known and behaviorally-similar threats, the detection of completely novel attack patterns (zero-day exploits) remains an ongoing challenge that requires continuous model retraining and data enrichment.

Finally, while the system is horizontally scalable, extremely high-velocity network environments (e.g., Tier 1 ISPs) would require further optimization and potential hardware acceleration to maintain real-time responsiveness at petabyte-scale traffic volumes.

6.3 Future Work

6.3.1 Enhancements to Threat Detection Models

Future development efforts will focus on enhancing the existing threat detection pipeline by integrating more advanced deep learning architectures such as Transformer models, which have demonstrated superior sequence learning capabilities in other domains. Incorporating semi-supervised and unsupervised learning approaches could also help detect previously unseen attack types without relying exclusively on labeled data.

Additionally, expanding the range of behavioral features captured during flow encoding, such as frequency domain characteristics, and multi-flow aggregation patterns, may improve anomaly detection fidelity.

6.3.2 Adaptive Network Optimization

There are substantial opportunities to further improve Vigilante's AI Optimization Module through the application of meta-reinforcement learning techniques. An adaptive RL agent that adjusts its learning rates and exploration strategies based on network volatility could significantly enhance system responsiveness under rapidly changing traffic conditions [4].

Integration of Software-Defined Networking (SDN) controllers would also allow Vigilante to not only recommend optimization strategies but directly implement network path changes, thereby closing the optimization loop autonomously [17].

6.3.3 Expanding Notification and Reporting Systems

To increase administrative convenience and situational awareness, the notification system could be expanded to support additional channels such as Slack, Microsoft Teams, and on-call incident response integrations like PagerDuty. A richer reporting suite could include trend visualizations, and compliance-oriented auditing reports generated from the system's event logs.

6.3.4 Cloud-native Deployment and Scalability Improvements

Vigilante's modular microservices architecture is inherently suitable for cloud-native deployment. Future iterations of the system will focus on containerization using Kubernetes and leveraging service meshes like Istio for enhanced observability, security, and automated traffic management.

Dynamic auto-scaling based on real-time traffic loads will be incorporated to ensure that system performance remains optimal regardless of fluctuating network conditions.

6.3.5 Advanced Threat Intelligence Integration

Another future direction involves integrating external threat intelligence feeds to enrich Vigilante's detection capabilities. By correlating local flow anomalies with global threat indicators, Vigilante can significantly improve its situational awareness and detect emerging attack patterns before they impact the network.

Open standards such as STIX/TAXII for threat sharing can be adopted to ensure interoperability with other cybersecurity ecosystems and to contribute findings back to the broader security community [11].

Glossary

Admin Dashboard A centralized mobile-based interface for real-time network monitoring, alert management, and administrative control over the Vigilante system.

AI (Artificial Intelligence) The simulation of human intelligence processes by machines, especially computer systems. In Vigilante, AI is used to analyze network traffic and detect anomalies.

Anomaly Detection The identification of patterns in data that do not conform to expected behavior, used in Vigilante to spot unusual network activities that could indicate security threats.

APNs (Apple Push Notification Service) A platform service for sending push notifications to Apple devices.

Asynchronous Communication A communication method where data transmission does not happen at the same time, allowing components to operate independently without waiting for responses.

Backend API The central communication layer of Vigilante that manages interactions between microservices, orchestrates data flow, and supports system-wide integration.

Blockchain Integration The incorporation of blockchain technology to enhance transparency, data integrity, and security within distributed network environments.

Botnet A network of private computers infected with malicious software and controlled as a group without the owners' knowledge.

Cisco DNA Center A network management and orchestration platform by Cisco that uses intent-based networking, automation, and AI-driven assurance for optimized network management.

Cyber Threats Malicious attempts to damage, disrupt, or gain unauthorized access to computer networks or systems.

Database The central storage system for Vigilante, managing user configurations, historical events, and operational records with consistency and integrity.

Database Integration The process of connecting Vigilante to a database, such as Firebase, to manage and store unstructured network traffic data.

DDoS (Distributed Denial of Service) An attack where multiple compromised systems flood the bandwidth or resources of a targeted system, usually one or more web servers.

Edge Computing A distributed computing paradigm that brings computation and data storage closer to data sources to reduce latency and improve performance.

Event-Driven Architecture A software architecture paradigm where services communicate through events, enabling real-time, responsive, and loosely coupled systems.

Federated Learning A machine learning technique where multiple decentralized devices collaboratively train a model without sharing raw data, enhancing privacy and data security.

False Positive Rate The percentage of benign activities incorrectly classified as malicious by the detection system.

Firebase A NoSQL cloud database service by Google, used for real-time data storage and retrieval in Vigilante.

Firebase Cloud Messaging (FCM) A cross-platform messaging solution that enables push notifications to client devices.

Frontend Interface A mobile-based user interface that allows network administrators to monitor network status, receive alerts, visualize data, and manage configurations remotely.

Future Enhancements Planned improvements and expansions to Vigilante, focusing on advanced AI features, improved scalability, and enhanced threat response capabilities.

HTTPS/SSL/TLS Protocols used for secure communication over computer networks, particularly the internet.

IDS (Intrusion Detection System) A security solution that monitors network or system activities for malicious activities or policy violations, generating alerts for administrators.

Intent-Based Networking A modern networking approach where business intent is captured and the network automatically translates it into policies and enforces them.

Intrusion Prevention System (IPS) A network security/threat prevention technology that examines network traffic flows to detect and prevent vulnerability exploits. Vigilante functions as an IPS.

Latency The time delay between the cause and the effect of some physical change in the system, critical for measuring real-time communication performance.

Login Interface A user authentication system implementing Role-Based Access Control (RBAC) to ensure secure access.

Machine Learning (ML) A subset of AI involving algorithms that improve automatically through experience and data. Vigilante uses ML for dynamic traffic analysis and threat detection.

Machine Learning and AI Models External libraries such as TensorFlow and PyTorch integrated into Vigilante for network traffic analysis and optimization.

Malware Software designed to cause damage to a computer, server, or network. Includes viruses, worms, trojans, ransomware, spyware, adware, etc.

Microservices Architecture An architectural style where the system is divided into small, independent modules, each responsible for specific functionalities, improving scalability and maintainability.

Modular Architecture A design principle allowing Vigilante's system components to operate independently, enabling easier upgrades and system scalability.

Modularity The design principle of dividing a system into distinct, interchangeable components that can function independently.

Network Monitoring Module A core Vigilante component responsible for observing and analyzing real-time network traffic.

Network Optimization Techniques and technologies used to improve network performance, including speed, reliability, and security. Vigilante incorporates optimization features.

Network Protocols Standards like TCP/IP and SNMP used for network communication, device monitoring, and data transmission.

Network Security The strategies, policies, and technologies used to protect the integrity, confidentiality, and availability of computer networks and data.

Network Throughput The amount of data successfully transferred over a network in a given time frame.

Network Traffic The data moving across a network at any given time. Vigilante monitors and analyzes network traffic for security and performance purposes.

Notification Protocols Mechanisms like SMTP, SMS gateways, and push services (FCM, APNs) used to send real-time alerts and notifications.

Notification Services The part of Vigilante that handles sending real-time alerts and notifications to administrators based on security events.

Palo Alto Prisma Access A Secure Access Service Edge (SASE) platform that offers cloud-delivered security, including threat prevention, DNS filtering, and zero-trust network access (ZTNA).

Precision The ratio of correctly predicted positive observations to the total predicted positive observations, used to evaluate classification models.

Predictive AI AI techniques that use historical and live data to anticipate potential network issues and security threats before they occur.

Proactive Threat Mitigation The process of identifying and neutralizing threats before they can cause harm. Vigilante emphasizes proactive defense mechanisms.

Recall The ratio of correctly predicted positive observations to all actual positive observations, measuring a model's ability to detect threats.

Real-Time Monitoring The continuous observation of network traffic and system activities with immediate processing and reporting of anomalies.

Reinforcement Learning Agent A type of machine learning system that learns optimal actions through trial-and-error interactions with an environment to maximize performance measures.

RESTful APIs Application Programming Interfaces (APIs) that use HTTP requests to perform CRUD operations, enabling integration with external systems.

RNN (Recurrent Neural Network) A type of artificial neural network suited for sequential data processing, used in Vigilante for threat classification based on network traffic patterns.

Role-Based Access Control (RBAC) A security mechanism that restricts system access based on users' roles and permissions.

Scalability The capacity of the Vigilante system to efficiently expand its resources and functionalities as the network grows.

Scalability Improvements Enhancements aimed at allowing the Vigilante system to handle increasing loads efficiently across larger or more complex network environments.

Sensitive Information Data that must be protected from unauthorized access to safeguard privacy or security (e.g., academic records, administrative data).

SNMP (Simple Network Management Protocol) A protocol used for monitoring and managing network devices.

Static Rules and Signatures Traditional security methods that rely on predefined patterns to detect threats; considered less adaptive compared to AI/ML-based approaches.

Stealthwatch A Cisco solution that provides network visibility and security analytics using behavioral modeling and anomaly detection techniques.

StratoLetters System A symbolic encoding system that converts network flow features into symbol sequences for input to AI models.

Symbolic Encoding The process of transforming raw network flow data into sequences of symbols to facilitate more effective machine learning analysis.

Syslog A standard protocol used for message logging and log management across network systems.

Testbed Network A controlled, experimental environment created to simulate real-world network conditions for testing and evaluation purposes.

Threat Prediction The use of AI models to forecast potential cybersecurity threats based on historical and real-time network data.

Vigilante The AI-driven Intrusion Prevention System project designed to enhance the security and performance of GIKI's network infrastructure.

Vigilante AI Models Machine learning-based modules within Vigilante that perform network analysis, anomaly detection, optimization, and threat prediction.

WebSockets A protocol providing full-duplex communication channels over a single TCP connection, enabling real-time updates between frontend and backend.

Zeek An open-source network analysis framework used for capturing and analyzing network traffic to generate structured datasets.

Zero Trust Architecture (ZTA) A security concept centered on the principle that no user or device should be trusted by default, even inside the network perimeter, requiring strict verification for every access request.

Bibliography

[1] Al-kahtani, M., Mehmood, Z., Sadad, T., Zada, D.-I., Ali, G. and Elaffendi, M. [2023], 'Intrusion detection in the internet of things using fusion of gru-lstm deep learning model', *Intelligent Automation and Soft Computing* **37**.

[2] Alshamrani, A., Alwan, M. and Alshammari, R. [2023], 'Cyber security: State of the art, challenges and future directions', *Journal of Information Security and Applications* **71**, 103–112.

[3] Badr, Y. [2022], 'Enabling intrusion detection systems with dueling double deep q-learning', *Digital Transformation and Society* **1**(1), 115–141.

[4] Bouzidi, H., Outtagarts, A. and Langar, R. [2019], Deep reinforcement learning application for network latency management in software defined networks, *in* '2019 IEEE Global Communications Conference (GLOBECOM)'.

[5] Chinnasamy, P. [2025], 'Ai-powered predictive analytics for cloud performance optimization and anomaly detection', *International Journal of Science and Research (IJSR)* **14**, 629–642.

[6] Cisco Systems [2023], *Cisco DNA Center Administrator Guide, Release 2.3.3*.

[7] Dave, D., Sawhney, G., Aggarwal, P., Silswal, N. and Khut, D. [2023], 'The new frontier of cybersecurity: Emerging threats and innovations', *arXiv preprint arXiv:2311.02630* .

[8] Goswami, M. J. [2024], 'Enhancing network security with ai-driven intrusion detection systems', *International Journal of Computer Applications* **12**.

[9] Juniper Networks [2021], *Every College and University Needs the Juniper Network Driven by Mist AI*.

[10] Kizza, J. [2014], 'Network security – an updated perspective', *Journal of Cyber Policy* **1**(1), 1–10.

[11] Ko, E., Kim, T. and Kim, H. [2018], 'Management platform of threats information in iot environment', *Journal of Ambient Intelligence and Humanized Computing* **9**(4), 1167–1176.
URL: *https://doi.org/10.1007/s12652-017-0581-6*

[12] Kumar, A. and Singh, R. [2023], 'A comprehensive discussion on network security', *International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)* **9**(1), 20–28.

[13] Liu, H. and Lang, B. [2019], 'Machine learning and deep learning methods for intrusion detection systems: A survey', *Applied Sciences* **9**(20), 4396.

[14] Oluwa, A. [2023], 'Network security concepts, dangers, and defense best practices', *International Journal of Computer Applications* **182**(2), 15–22.

[15] Palo Alto Networks [2024], *AI Access Security*.

[16] Perova, T. [2022], 'The importance of network security in protecting sensitive data and information', *International Journal of Research and Innovation in Applied Science* **7**(9), 1–5.

[17] Ruan, L., Dias, M. and Wong, E. [2020], 'Achieving low-latency human-to-machine (h2m) applications: An understanding of h2m traffic for ai-facilitated bandwidth allocation', *IEEE Internet of Things Journal* **PP**, 1–1.

[18] Sekkappan, R. [2024], 'Ai in network security: Enhancing protection in the age of automation', *International Journal of Scientific Research in Computer Science, Engineering and Information Technology* **10**, 971–980.

[19] SolarWinds [2024], *Network Performance Monitor - Observability Self-Hosted*.

[20] Stratosphere Linux IPS Project [2024], *VirusTotal Module in Stratosphere Linux IPS Documentation*.

[21] Zeek Project [2025], *Book of Zeek*.