RHINO - INTRUSION DETECTION SYSTEM USING AI

Software Requirements Specification (SRS)

Field	**Details**
Version	1.0
Prepared By	1. Sakib Dalal 2. Aishwarya Patil
Department	AI & ML
Internship At	intiGrow
Internship Guide	[Guide Name]
Date	January 23, 2026

Contents

1 Introduction

- **Intrusion Detection System (IDS):** An intrusion detection system is a device or software application that monitors a network or systems for malicious activity or policy violations. Any intrusion activity or violation is typically either reported to an administrator or collected centrally using a security information and event management system. A SIEM system combines outputs from multiple sources and uses alarm filtering techniques to distinguish malicious activity from false alarms. IDS types range in scope from single computers to large networks. The most common classifications are network intrusion detection systems and host-based intrusion detection systems. A system that monitors important operating system files is an example of an HIDS, while a system that analyzes incoming network traffic is an example of an NIDS [1].

- **AI based IDS:** An AI-based intrusion detection system (IDS) uses artificial intelligence technologies to monitor network traffic and identify unauthorized access or malicious activities. These systems leverage machine learning to analyze data in real-time, allowing them to detect novel threats and adapt to emerging cyber risks more effectively than traditional methods [2].

1.1 Purpose

The purpose of this document is to clearly define and detail the software and system requirements for the AI-Based Intrusion Detection System (IDS). This system aims to enhance cybersecurity by leveraging artificial intelligence and machine learning techniques to identify, analyze, and respond to both known and unknown security threats in real time.

The proposed IDS is designed to operate effectively across heterogeneous and distributed computing environments, including cloud-based virtual machines (VMs), containerized infrastructures, edge computing devices, and Internet of Things (IoT) nodes. By continuously monitoring network traffic, system logs, and behavioral patterns, the system seeks to detect anomalies, malicious activities, and policy violations with high accuracy and minimal false positives.

This Software Requirements Specification (SRS) serves as a comprehensive reference document throughout the system's lifecycle—from design and development to testing, deployment, and maintenance. It establishes a common understanding of system capabilities, constraints, interfaces, and performance expectations among all stakeholders.

This document is intended for the following audiences:

- **Developers** – to understand functional and non-functional requirements for implementation
- **Mentors/Supervisors** – to guide and evaluate technical design decisions
- **Project Evaluators** – to assess compliance with defined objectives and requirements
- **Security Engineers** – to validate threat detection mechanisms and security controls
- **QA Testers** – to design test cases and verify system reliability, accuracy, and robustness

1.2 Scope

The AI-Based Intrusion Detection System (RHINO) is designed as a unified security monitoring platform to protect modern hybrid digital infrastructures through real-time visibility and intelligent threat detection. At the edge, lightweight monitoring agents developed in Go and C++ are deployed across diverse endpoints, including cloud virtual machines, containers, local servers, and IoT nodes such as Raspberry Pi and ESP32 [3], [4], [5]. These agents continuously collect critical host-based metrics—such as CPU memory, RAM usage, and temperature—alongside network-based data like traffic IP and upload/download speeds while maintaining minimal resource overhead.

All telemetry is securely transmitted via TCP (Port 9092) to a centralized Apache Kafka cluster hosted on AWS EC2, which serves as a high-throughput, fault-tolerant data ingestion and streaming layer [6]. Following ingestion, the data enters a specialized processing pipeline where Python-based consumers utilize Pandas and Scikit-learn for feature engineering, data normalization, and scaling. A dedicated Data Processing Engine, powered by Python3 and Flask, then manages the refined data flow, ensuring it is properly structured for both immediate analysis and long-term storage.

A critical component of this architecture is the integration of InfluxDB, a high-performance time-series database used to store the processed telemetry. By utilizing InfluxDB, the system can efficiently handle the high-volume metrics generated by distributed agents, enabling rapid querying and historical trend analysis [7]. This stored data feeds into the core intelligence layer, where Scikit-learn and PyTorch models identify suspicious patterns and anomalies to detect both known signatures and zero-day threats in real time [8], [9].

To ensure operational efficiency, the system provides a web-based dashboard built with React.js, Tailwind CSS, and Three.js for asset onboarding, device management, and alert visualization. The backend, powered by Node.js and Express, integrates with AWS Cognito for secure authentication and AWS DynamoDB for persistent configuration storage. This modular, cloud-ready design allows RHINO to operate as a Software-as-a-Service (SaaS) solution, providing comprehensive security visibility across an entire organization with minimal deployment complexity [10].

1.3 Document Conventions

- **FR:** Functional Requirement
- **NFR:** Non-Functional Requirement
- Bold items represent critical modules or shortlisted technologies.

1.4 References

References

[1] Wikipedia contributors. "Intrusion Detection System — Wikipedia, The Free Encyclopedia." Accessed: 2026-01-19. [Online]. Available: `https://en.wikipedia.org/wiki/Intrusion_detection_system`.

[2] Faddom, *Ai intrusion detection: Components, challenges and best practices*, `https://faddom.com/ai-intrusion-detection-components-challenges-and-best-practices/`, Accessed: 2025-01-19, Jul. 2025.

[3] The Go Authors, *Documentation — the go programming language*, `https://go.dev/doc/`, Accessed: 2026-01-19, 2026.

[4] Microsoft, *C++ documentation — microsoft learn*, `https://learn.microsoft.com/en-us/cpp/cpp/?view=msvc-170`, Accessed: 2026-01-19, 2026.

[5] Wikipedia contributors, *Raspberry pi foundation — wikipedia, the free encyclopedia*, `https://en.wikipedia.org/wiki/Raspberry_Pi_Foundation`, Accessed: 2026-01-19, 2025.

[6] Apache Software Foundation, *Kafka – getting started: Introduction*, `https://kafka.apache.org/41/getting-started/introduction/`, Accessed: 2026-01-19, 2026.

[7] InfluxData, Inc., *Influxdata: Real-world ai starts with time series*, `https://www.influxdata.com/`, Accessed: 2026-01-23, 2026.

[8] scikit-learn developers, *Scikit-learn: Machine learning in python*, `https://scikit-learn.org/stable/index.html`, Accessed: 2026-01-19, 2026.

[9] PyTorch Contributors, *Pytorch documentation*, `https://docs.pytorch.org/docs/stable/index.html`, Accessed: 2026-01-19, 2026.

[10] Wikipedia contributors, *Software as a service — wikipedia, the free encyclopedia*, https://en.wikipedia.org/wiki/Software_as_a_service, Accessed: 2026-01-19, 2025.

1.5 Intended Audience

The intended audience includes:

- Project Mentors and Evaluators – to review the technical scope, assess requirement completeness, and evaluate the system against academic and project objectives.
- Internship Company Engineers – to understand system architecture, integration points, and deployment considerations within real-world production or enterprise environments.
- Security Analysts – to examine threat detection mechanisms, alerting workflows, and the overall security posture provided by the IDS.
- Developers and DevOps Engineers – to implement, deploy, monitor, and maintain the system, including agent development, backend services, CI/CD pipelines, and cloud infrastructure.

2 Overall Description

This section provides a high-level overview of the RHINO Intrusion Detection System, including its system context, major functions, user categories, operating environment, and underlying assumptions and dependencies.

2.1 Product Perspective

RHINO is an AI-based Intrusion Detection System (IDS) designed as a modular, cloud-native security monitoring platform. The system follows a distributed architecture where lightweight host and network monitoring agents are deployed across heterogeneous environments such as cloud virtual machines, containerized workloads, local servers, and edge or IoT devices.

These agents continuously collect host-level metrics (CPU usage, memory utilization, temperature, and process information) and network-level metrics (traffic flow, IP addresses, upload/download speed, and port activity). The collected data is streamed securely to a centralized backend using Apache Kafka as the data ingestion and messaging layer.

The backend system processes incoming telemetry using Python-based consumers, performs feature engineering and normalization, and stores the refined data in a time-series database (InfluxDB). Machine learning models built using Scikit-learn and PyTorch analyze this data to detect anomalies and potential intrusions. A web-based dashboard and REST APIs allow users to visualize alerts, manage devices, and respond to threats. RHINO is designed to operate as a Software-as-a-Service (SaaS) solution while remaining extensible for on-premise deployments.

2.2 Product Functions

The primary functions of the RHINO system include:

- Continuous monitoring of host and network activities across distributed systems.
- Secure collection and streaming of telemetry data using Kafka.
- Real-time data preprocessing, feature extraction, and normalization.
- AI-driven detection of known attacks and unknown (zero-day) anomalies.
- Rule-based and machine learning–based decision making for threat classification.
- Storage of time-series security data for historical analysis and auditing.
- Alert generation and notification for detected intrusions.

- Web-based visualization of system status, alerts, and device health.
- REST API support for integration with external security tools and services.

2.3 User Classes

The RHINO system is intended for the following user classes:

- **System Administrators**: Responsible for deploying agents, configuring monitoring policies, and managing infrastructure security.
- **Security Analysts**: Monitor alerts, analyze detected threats, and investigate anomalous behavior using dashboards and logs.
- **DevOps Engineers**: Integrate RHINO into CI/CD pipelines, cloud environments, and container orchestration platforms.
- **Organization Management**: View high-level security insights and compliance-related reports.
- **API Consumers**: External systems or applications that consume RHINO APIs for automation or extended analytics.

2.4 Operating Environment

RHINO operates in a hybrid and distributed computing environment that includes:

- Cloud platforms such as AWS EC2 instances.
- Linux-based operating systems for agents and backend services.
- Apache Kafka for message streaming over TCP (Port 9092).
- Python runtime (Python 3.x) for data processing and AI components.
- InfluxDB for time-series data storage.
- Web browsers supporting modern JavaScript frameworks for dashboard access.
- Internet connectivity for SaaS-based deployment and remote monitoring.

2.5 Assumptions & Dependencies

The development and operation of RHINO are based on the following assumptions and dependencies:

- Target systems allow installation of lightweight monitoring agents.
- Continuous network connectivity is available between agents and the Kafka server.
- Cloud services such as AWS EC2, DynamoDB, and Cognito remain available and stable.
- Machine learning models are trained on representative datasets to ensure accuracy.
- Users possess basic knowledge of cybersecurity concepts and system administration.
- Third-party libraries and frameworks (Kafka, Scikit-learn, PyTorch, InfluxDB) are maintained and supported.

3 System Features

This section describes the major functional features of the RHINO AI-Based Intrusion Detection System. Each feature represents a core subsystem responsible for data collection, processing, threat detection, and user interaction.

3.1 Agent Deployment & Data Collection

RHINO deploys lightweight monitoring agents across heterogeneous environments including cloud virtual machines, containerized systems, local servers, and edge or IoT devices. These agents are designed to operate with minimal system overhead while providing continuous visibility into host and network activity.

The agents collect host-based metrics such as CPU usage, RAM utilization, disk activity, and system temperature, along with network-based metrics including traffic flow, IP addresses, port numbers, and upload/download speeds. The collected telemetry is packaged in structured messages and securely transmitted to the backend using a producer-based communication model. This ensures real-time data availability while maintaining scalability and fault tolerance.

3.2 Data Ingestion & Processing Layer

The data ingestion layer is responsible for receiving, buffering, and processing telemetry data generated by distributed agents. Apache Kafka serves as the central streaming platform, enabling high-throughput and reliable data ingestion over TCP (Port 9092).

Kafka consumers implemented in Python subscribe to relevant topics and process incoming streams. The processing pipeline performs feature engineering, data normalization, scaling, and labeling using libraries such as Pandas and Scikit-learn. A dedicated data processing engine built with Python and Flask manages data flow between components. Processed data is stored in InfluxDB, a time-series database optimized for high-frequency security telemetry and historical analysis.

3.3 AI-powered Threat Detection Engine

The AI-powered threat detection engine forms the core intelligence layer of RHINO. It utilizes both supervised and unsupervised machine learning techniques to identify malicious behavior and anomalous system activity.

Models developed using Scikit-learn and PyTorch analyze processed telemetry data to detect known attack patterns as well as zero-day threats. A rule-based decision engine complements the machine learning models to enhance detection accuracy and reduce false positives. Based on prediction outcomes, the system classifies events as normal activity or potential intrusions and triggers alerts when suspicious behavior is detected.

3.4 User Dashboard & APIs

RHINO provides a web-based dashboard that enables users to visualize system status, monitored devices, detected threats, and historical trends. The frontend is built using modern web technologies, allowing interactive charts, real-time alert updates, and device management features.

The backend exposes RESTful APIs that support secure communication between the dashboard, agents, and external systems. Authentication and authorization mechanisms ensure that only authorized users and devices can access system resources. These APIs also enable integration with third-party security tools, automation workflows, and incident response systems.

4 External Interface Requirements

This section describes the interfaces between the RHINO Intrusion Detection System and external hardware, software components, and communication mechanisms. These interfaces define how RHINO interacts with monitored systems, backend services, and user-facing applications.

4.1 Hardware Interfaces

RHINO interacts with various hardware platforms through lightweight monitoring agents. These agents are deployed on the following hardware environments:

- Cloud-based virtual machines hosted on platforms such as AWS EC2.
- Physical servers running Linux-based operating systems.
- Containerized environments managed through container runtimes.
- Edge and IoT devices such as Raspberry Pi and ESP32.

The system does not require specialized hardware peripherals. All interactions with hardware components are performed through the host operating system interfaces for accessing system metrics, network statistics, and process information.

4.2 Software Interfaces

RHINO integrates with multiple software components and third-party services to deliver its functionality. The key software interfaces include:

- **Operating Systems**: Linux-based operating systems for agent deployment and backend services.
- **Apache Kafka**: Used as the primary message broker for streaming telemetry data between agents and backend consumers.
- **InfluxDB**: Serves as the time-series database for storing processed security metrics and historical data.
- **Machine Learning Libraries**: Scikit-learn and PyTorch are used for model training and inference.
- **Web Backend**: RESTful services implemented using Node.js and Express for API handling.
- **Authentication Services**: AWS Cognito is used for user authentication and authorization.

- **Frontend Frameworks**: Web-based dashboard developed using React.js, Tailwind CSS, and Three.js.

All software interfaces follow standard data formats such as JSON for request and response payloads.

4.3 Communication Interfaces

RHINO relies on secure and reliable communication interfaces to transmit telemetry data and control messages. The primary communication interfaces include:

- **TCP/IP**: Used for data transmission between agents and the Kafka server.
- **Kafka Protocol**: Enables publish-subscribe messaging over TCP on Port 9092.
- **HTTP/HTTPS**: Used for REST API communication between the backend services and the web dashboard.
- **Encryption**: Secure communication channels are established using encryption mechanisms to protect data in transit.

These communication interfaces ensure real-time data flow, secure access, and reliable integration between distributed system components.

5 Non-Functional Requirements

This section defines the non-functional requirements (NFRs) for the RHINO Intrusion Detection System. These requirements specify quality attributes such as performance, scalability, security, reliability, and usability that the system must satisfy to ensure effective and dependable operation.

5.1 Performance Requirements

- The system shall process incoming telemetry data in near real-time with minimal latency.
- Kafka consumers shall handle high-throughput data streams without message loss.
- The AI inference engine shall generate intrusion detection results within an acceptable time window suitable for real-time alerting.
- Dashboard updates shall reflect new alerts and system metrics with minimal delay.

5.2 Scalability Requirements

- The system shall support horizontal scaling to accommodate an increasing number of monitored devices and agents.
- Kafka topics and partitions shall be scalable to handle growing data volumes.
- The backend services shall support scaling in cloud environments without service interruption.

5.3 Reliability and Availability

- The system shall ensure high availability of data ingestion and processing components.
- Temporary network failures shall not result in permanent data loss.
- The system shall recover gracefully from component failures.

5.4 Security Requirements

- All communication between agents and backend services shall be encrypted.
- Authentication and authorization mechanisms shall prevent unauthorized access to system resources.

- Sensitive security data shall be protected against unauthorized disclosure and tampering.
- Role-based access control shall be enforced for dashboard and API usage.

5.5 Usability Requirements

- The web dashboard shall provide an intuitive and user-friendly interface.
- Alerts and notifications shall be clearly presented and easy to interpret.
- The system shall require minimal training for security analysts and administrators.

5.6 Maintainability Requirements

- The system shall follow a modular design to facilitate maintenance and upgrades.
- Logging and monitoring mechanisms shall support debugging and performance analysis.
- Machine learning models shall be updatable without requiring system downtime.

5.7 Portability Requirements

- Monitoring agents shall be portable across different Linux-based environments.
- Backend services shall be deployable on both cloud and on-premise infrastructures.

5.8 Compliance Requirements

- The system shall comply with applicable data protection and cybersecurity best practices.
- Logging and audit trails shall support organizational security policies and compliance needs.

6 Other Requirements

This section describes additional requirements that do not fall strictly under functional or non-functional categories. These include ethical, legal, and practical considerations relevant to the deployment and operation of the RHINO Intrusion Detection System.

6.1 Ethical & Legal Concerns

The RHINO system is designed with careful consideration of ethical responsibilities and legal obligations related to cybersecurity monitoring and data handling.

- The system shall collect only security-relevant telemetry required for intrusion detection and system monitoring.
- User privacy shall be respected by avoiding unnecessary collection of personally identifiable information (PII).
- All collected data shall be handled in accordance with applicable data protection and privacy regulations.
- Access to monitoring data and alerts shall be restricted to authorized users only.
- The system shall ensure transparency in alert generation to avoid unfair or biased decision-making by AI models.
- Logs and audit trails shall be maintained to support accountability and forensic analysis.

6.2 Limitations

Despite its capabilities, the RHINO Intrusion Detection System has certain limitations that users and administrators must consider:

- The accuracy of threat detection depends on the quality and representativeness of training data used for machine learning models.
- False positives or false negatives may occur, particularly for novel or highly sophisticated attacks.
- Real-time detection performance may be affected under extremely high data ingestion loads.
- The system requires stable network connectivity between agents and backend services.

- RHINO provides detection and alerting capabilities but does not automatically remediate or block attacks.
- The system's effectiveness may vary across different deployment environments and configurations.

7 Appendix

This appendix provides supplementary information to support the understanding of the RHINO Intrusion Detection System, including acronyms, tools and technologies used, project milestones, and an example dashboard wire-frame.

7.1 Acronyms

- **AI** – Artificial Intelligence
- **API** – Application Programming Interface
- **AWS** – Amazon Web Services
- **CI/CD** – Continuous Integration / Continuous Deployment
- **CPU** – Central Processing Unit
- **HIDS** – Host-based Intrusion Detection System
- **IDS** – Intrusion Detection System
- **IoT** – Internet of Things
- **ML** – Machine Learning
- **NFR** – Non-Functional Requirement
- **NIDS** – Network Intrusion Detection System
- **PII** – Personally Identifiable Information
- **REST** – Representational State Transfer
- **SaaS** – Software as a Service
- **SRS** – Software Requirements Specification
- **TCP/IP** – Transmission Control Protocol / Internet Protocol
- **VM** – Virtual Machine

7.2 Tools & Technologies Used

The RHINO Intrusion Detection System is developed using the following tools and technologies:

- **Programming Languages**: Go, C++, Python, JavaScript
- **Streaming Platform**: Apache Kafka

- **Databases**: InfluxDB, AWS DynamoDB
- **Machine Learning Libraries**: Scikit-learn, PyTorch
- **Backend Frameworks**: Flask, Node.js, Express
- **Frontend Frameworks**: React.js, Tailwind CSS, Three.js
- **Cloud Platform**: Amazon Web Services (AWS)
- **Authentication**: AWS Cognito
- **Version Control**: Git and GitHub

7.3 Project Milestones

The key milestones of the RHINO project are outlined below:

- Requirement analysis and SRS documentation
- System architecture and design finalization
- Agent development for host and network monitoring
- Kafka-based data ingestion pipeline implementation
- Data processing and feature engineering module development
- AI model training and integration
- Web dashboard and API development
- System testing, validation, and performance evaluation

7.4 Example Dashboard Wire-frame

This subsection provides an example wire-frame of the RHINO web dashboard. The dashboard presents real-time alerts, system metrics, monitored devices, and historical trends to assist security analysts and administrators in effective threat monitoring and analysis.

8 References

References

[1] Wikipedia contributors, "Intrusion Detection System – Wikipedia, The Free Encyclopedia," [Online]. Available: `https://en.wikipedia.org/wiki/Intrusion_detection_system` (Accessed: Jan. 19, 2026).

[2] Faddom, "AI Intrusion Detection: Components, Challenges and Best Practices," [Online]. Available: `https://faddom.com/ai-intrusion-detection-components-challenges-and-best-practices/` (Accessed: Jan. 19, 2026).

[3] The Go Authors, "Documentation – The Go Programming Language," [Online]. Available: `https://go.dev/doc/` (Accessed: Jan. 19, 2026).

[4] Microsoft, "C++ Documentation – Microsoft Learn," [Online]. Available: `https://learn.microsoft.com/en-us/cpp/cpp/?view=msvc-170` (Accessed: Jan. 19, 2026).

[5] Wikipedia contributors, "Raspberry Pi Foundation – Wikipedia, The Free Encyclopedia," [Online]. Available: `https://en.wikipedia.org/wiki/Raspberry_Pi_Foundation` (Accessed: Jan. 19, 2026).

[6] Apache Software Foundation, "Kafka – Getting Started: Introduction," [Online]. Available: `https://kafka.apache.org/41/getting-started/introduction/` (Accessed: Jan. 19, 2026).

[7] InfluxData Inc., "InfluxData: Real-World AI Starts with Time Series," [Online]. Available: `https://www.influxdata.com/` (Accessed: Jan. 23, 2026).

[8] Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," [Online]. Available: `https://scikit-learn.org/stable/index.html` (Accessed: Jan. 19, 2026).

[9] PyTorch Contributors, "PyTorch Documentation," [Online]. Available: `https://docs.pytorch.org/docs/stable/index.html` (Accessed: Jan. 19, 2026).

[10] Wikipedia contributors, "Software as a Service – Wikipedia, The Free Encyclopedia," [Online]. Available: `https://en.wikipedia.org/wiki/Software_as_a_service` (Accessed: Jan. 19, 2026).