Contents

1 Introduction

1.1 System Introduction

The AI & RL-Driven Distributed Intrusion Detection System (DIDS) is a modern cybersecurity framework developed to protect the banking sector's distributed infrastructure—including ATMs, branch networks, online banking portals, and cloud services—against advanced and evolving cyber threats. Unlike traditional Intrusion Detection Systems that rely solely on static signatures, DIDS combines Artificial Intelligence (AI), Machine Learning (ML), and Reinforcement Learning (RL) to intelligently monitor, detect, and respond to both known and novel (zero-day) cyberattacks in real-time.

DIDS is architected as a scalable, cloud-native solution using microservices deployed via Docker containers and managed by Kubernetes on Azure. The system operates by continuously capturing and analyzing network traffic and transaction patterns from multiple sources. It detects anomalies through deep learning-based models trained on recent and comprehensive cybersecurity datasets. Simultaneously, it uses signature-based engines like Snort and Suricata to identify known attack patterns.

With integrated threat intelligence feeds and an adaptive RL module, DIDS enhances real-time threat response by learning from past attack patterns and automatically mitigating risks. Its modular architecture ensures high availability, fault tolerance, and ease of deployment across diverse financial environments.

The system also complies with regulatory frameworks such as PCI-DSS, ISO 27001, and GDPR, ensuring secure handling of sensitive financial data. Through an intuitive dashboard, DIDS provides real-time alerts, visualizations, and audit-ready reports, making it a robust and intelligent defense platform for modern banking networks.

1.2 Background of the System

In an increasingly digitized and interconnected financial world, the banking sector has become a primary target for sophisticated cyberattacks, including phishing, data breaches, insider threats, malware infections, and advanced persistent threats (APTs). Traditional Intrusion

Detection Systems (IDS) rely heavily on static rule sets and known attack signatures, which often prove ineffective against novel, evolving, or zero-day attacks. Moreover, with the rise of distributed banking infrastructures—spanning branches, ATMs, cloud services, and online platforms—the need for a dynamic, intelligent, and distributed security solution has become critical.

AI-driven cybersecurity solutions have shown significant promise in identifying hidden patterns and detecting anomalies that traditional systems might overlook. However, many existing AI-powered IDS solutions are limited in adaptability, lack real-time response capabilities, and are not designed to operate efficiently in distributed, resource-diverse environments like those found in large-scale banking networks.

The proposed Distributed Intrusion Detection System (DIDS) addresses these challenges by integrating Artificial Intelligence (AI), Machine Learning (ML), and Reinforcement Learning (RL) in a cloud-native microservices architecture. Unlike conventional tools, DIDS learns from historical and real-time data, adapts to emerging threats using reinforcement learning, and offers both anomaly-based and signature-based threat detection.

By leveraging modern datasets (e.g., CICIDS2018, UNSW-NB15, TON_IoT 2024), edge-compatible deployment via Docker/Kubernetes, and real-time threat intelligence feeds, DIDS is designed to provide a scalable, resilient, and intelligent security layer for the modern banking ecosystem. It aims to bridge the gap between outdated detection tools and the real-time, adaptive defense needed to protect critical financial systems from increasingly sophisticated cyber threats.

1.3 Objectives of the System

The primary objectives of the AI & RL-Driven Distributed Intrusion Detection System (DIDS) are:

- Detect known and unknown cyber threats using a hybrid approach combining Machine Learning (ML), Reinforcement Learning (RL), and signature-based detection techniques.
- Monitor real-time network traffic and financial transactions across distributed banking infrastructures, including ATMs, branch networks, and cloud-based banking systems.

- Prevent financial fraud, insider threats, and zero-day attacks by continuously learning from new data patterns and adapting detection strategies through RL.
- Ensure regulatory compliance with banking cybersecurity standards such as PCI-DSS, ISO 27001, and GDPR by enforcing secure data handling and automated security auditing.
- Provide an interactive dashboard for real-time threat visualization, alerts, and incident reporting accessible to SOC teams and bank administrators.
- Enable scalable deployment using Docker containers and Kubernetes on cloud platforms (e.g., Azure) to support high-traffic, multi-branch environments.
- Integrate live threat intelligence feeds from sources like IBM X-Force and OTX to proactively identify emerging threats and blacklisted entities.
- Maintain user accessibility and security by implementing role-based access control, encrypted communication, and multi-user management for administrators and analysts.
- Support offline threat detection and logging at edge nodes (e.g., ATM servers) with synchronization to the central system when reconnected.
- Provide modular and fault-tolerant architecture for ease of maintenance, upgrades, and integration with existing banking systems.

1.4 Significance of the System

The AI & RL-Driven Distributed Intrusion Detection System (DIDS) is highly significant in the evolving cybersecurity landscape of the banking sector, where real-time protection against complex threats is crucial. Its importance spans multiple areas:

- **Real-Time Cyber Threat Mitigation:** DIDS offers continuous, real-time monitoring and detection of cyber threats across distributed banking environments, minimizing the risk of financial fraud, data breaches, and unauthorized access.

- **Zero-Day Attack Detection:** By leveraging AI and Reinforcement Learning, DIDS can adaptively detect and respond to new and previously unseen attack patterns, providing an edge over traditional signature-based systems.
- **Protection of Sensitive Financial Data:** The system ensures secure handling of customer data in compliance with global standards like PCI-DSS, GDPR, and ISO 27001, reducing legal and reputational risks for financial institutions.
- **Scalability Across Distributed Infrastructure:** Its cloud-native, microservices-based architecture allows deployment across multiple banking branches, ATMs, and online systems, ensuring consistent protection at every endpoint.
- **Support for Security Operations Centers (SOCs):** DIDS provides actionable insights, real-time dashboards, and automated alerts, enabling faster incident response and reducing the workload of cybersecurity teams.
- **Cost-Effective Banking Security-as-a-Service (B-SaaS):** Financial institutions can access enterprise-grade threat detection and compliance tools without heavy infrastructure investments.
- **Educational and Research Relevance:** The system contributes to cybersecurity research by integrating modern datasets and testing adaptive AI models, providing a foundation for further academic and industrial development.

Overall, DIDS is a forward-looking solution that addresses the limitations of legacy intrusion detection tools and brings intelligent, distributed, and responsive security into critical financial ecosystems.

2 Overall Description

2.1 Product Perspective

The AI & RL-Driven Distributed Intrusion Detection System (DIDS) is a cloud-native, modular security solution built specifically for the banking sector. It is designed to operate as an independent yet integrable platform that can be seamlessly deployed across a distributed financial infrastructure, including on-premise banking networks, ATM systems, and cloud-based banking services.

The system is built on a **microservices architecture**, using containerized components deployed via **Docker** and orchestrated through **Kubernetes** within the **Azure Kubernetes Service (AKS)** environment. This allows for highly scalable and resilient deployment across multiple geographic locations and network environments.

DIDS comprises the following key components:

- **Traffic Capture and Preprocessing Module:** Deployed at distributed endpoints (e.g., branch offices or ATMs), this module collects real-time network data and securely forwards it for analysis.
- **AI & RL Detection Engine:** Hosted on Azure, this core component uses ML algorithms and reinforcement learning policies to identify both known and zero-day attacks in real time.
- **Signature-Based Detection Engine:** Integrates with open-source tools like **Snort** and **Suricata** to detect predefined attack patterns for fast and reliable identification of well-known threats.
- **Threat Intelligence and Response Module:** Fetches threat indicators from sources like IBM X-Force and OTX and helps the system automatically respond to high-confidence threats using trained RL agents.
- **Dashboard & API Layer:** A centralized web-based interface and RESTful API system allow security teams to monitor alerts, visualize analytics, manage compliance reports, and interact with system components securely.

- **Cloud Backend (Azure + MongoDB):** All logs, anomalies, alerts, and user activity are stored securely and managed using cloud-based data infrastructure to ensure high availability, disaster recovery, and scalability.

In essence, DIDS is designed to function both as a standalone platform and as an integrable component of a broader enterprise security infrastructure, supporting **Banking Security-as-a-Service (B-SaaS)** deployments for financial institutions of varying sizes.

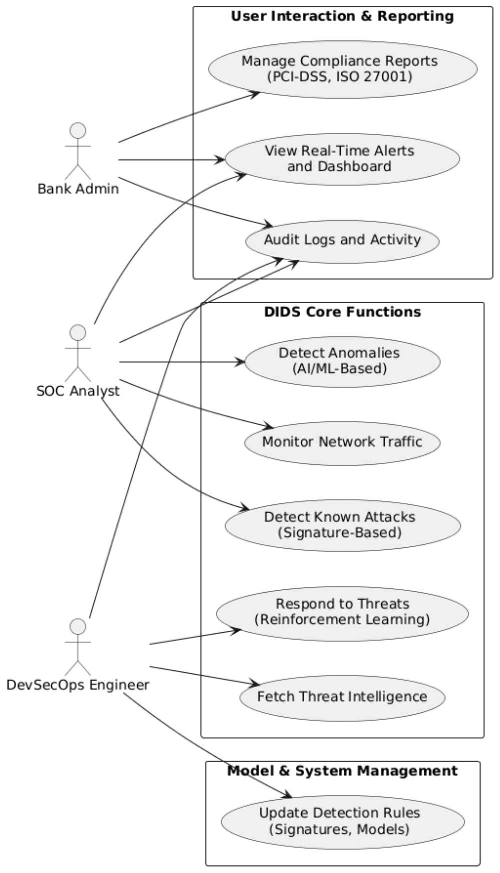
User Interaction & Reporting
Manage Compliance Reports (PCI-DSS, ISO 27001)
View Real-Time Alerts and Dashboard
Audit Logs and Activity
Bank Admin
SOC Analyst
DevSecOps Engineer
DIDS Core Functions
Detect Anomalies (AI/ML-Based)
Monitor Network Traffic
Detect Known Attacks (Signature-Based)
Respond to Threats (Reinforcement Learning)
Fetch Threat Intelligence
Model & System Management
Update Detection Rules (Signatures, Models)

Figure 2.1 Product Perspective of DIDS

2.2 Product Scope

In Scope:

The following features and capabilities are included in the scope of the Distributed Intrusion Detection System (DIDS):

- **Real-Time Intrusion Detection:** Continuous monitoring of network traffic and banking transaction logs across distributed systems (branches, ATMs, cloud services) to detect both known and unknown threats.
- **AI/ML & RL-Based Anomaly Detection:** Implementation of machine learning models to identify suspicious activities and zero-day attacks. Reinforcement learning enables adaptive threat mitigation strategies.
- **Signature-Base Threat Detection:** Use of intrusion detection tools such as Snort and Suricata to detect well-known attack patterns like brute force, malware, and phishing.
- **Live Threat Intelligence Integration:** Integration with external threat intelligence platforms (e.g., IBM X-Force, OTX) for real-time updates on emerging attack vectors and blacklisted sources.
- **Automated Threat Response System:** Execution of predefined responses (alert, block, isolate) using trained RL models based on threat severity and past experiences.
- **Compliance Management:** Support for security compliance with industry standards like PCI-DSS, GDPR, and ISO 27001 through automated auditing and report generation.
- **Monitoring Dashboard:** A centralized web-based dashboard for SOC analysts and administrators to visualize threats, access logs, and review system performance.
- **Cloud-Native Deployment:** Deployment using containerized microservices via Docker and Kubernetes on Azure for scalability, fault tolerance, and easy maintenance.

Out of Scope:

The following items are not part of this FYP's current development scope:

- **Active Countermeasures:** The system will not actively disrupt external attacker infrastructure (e.g., DDoS retaliation).
- **Internal HR/Employee Behavior Monitoring Tools:**DIDS does not include modules for physical access control, HR monitoring, or non-network-based behavioral analysis.

- **Mobile Banking App Security Integration:** While DIDS secures infrastructure traffic, it does not directly embed into mobile banking apps.
- **Legacy Hardware Support:** The system will be optimized for modern infrastructure and may not fully support outdated networking devices or unsupported operating systems.
- **Third-Party Legal Enforcement Tools:** DIDS does not communicate with law enforcement or initiate legal procedures based on detected threats.

2.3 Product Functionality

The AI & RL-Driven Distributed Intrusion Detection System (DIDS) is designed with modular and integrated features to provide financial institutions with real-time threat detection, monitoring, and response capabilities. The core functionalities of the product include:

Network Traffic Monitoring

- Continuously monitors network traffic across distributed banking environments (e.g., ATMs, branches, cloud-based services).
- Captures and processes packet-level data for further analysis.

AI/ML-Based Anomaly Detection

- Identifies unusual activity, fraud, or abnormal transaction patterns using trained ML models (e.g., Autoencoders, Isolation Forests).
- Detects zero-day attacks using adaptive learning models.

Reinforcement Learning-Based Response

- Uses trained RL agents to automatically take actions based on threat severity (e.g., block IP, isolate node, trigger alert).
- Continuously learns and adapts from past incident outcomes to improve future defense strategies.

Signature-Based Detection

- Detects known attacks (e.g., phishing, SQL injection, brute force) using open-source IDS tools like Snort and Suricata.
- Maintains an updated rule base for efficient signature matching and response.

Threat Intelligence Integration

- Fetches real-time threat feeds and blacklist data from platforms like IBM X-Force and Open Threat Exchange (OTX).
- Enhances proactive defense through early detection of emerging threats.

Dashboard & Reporting

- Provides a centralized web-based dashboard for monitoring real-time alerts, threat maps, and security trends.
- Allows users to generate detailed incident and compliance reports for analysis and regulatory purposes.

User Authentication & Access Control

- Implements secure login and access features for analysts and administrators using encrypted credentials.
- Role-based access controls ensure limited access to sensitive operations.

Compliance Management

- Ensures compliance with banking security standards like PCI-DSS, ISO 27001, and GDPR.
- Automates compliance checks, maintains audit trails, and generates regulatory reports.

Cloud Deployment & Scalability

- Deploys as containerized services using Docker and Kubernetes in Azure cloud.
- Supports auto-scaling, high availability, and fault tolerance for large-scale banking environments.

2.4 Users And Characters:

The AI & RL-Driven Distributed Intrusion Detection System (DIDS) is designed to serve multiple user types within a banking and cybersecurity context. Each user group interacts with the system at a different level of complexity and responsibility. Understanding these roles helps guide the system's interface design, permission structures, and user experience flow.

2.5 User Types

1. SOC Analysts (Security Operations Center Analysts)

- Primary users responsible for real-time monitoring and threat analysis.
- Moderate to advanced technical expertise in cybersecurity tools and procedures.
- Require access to detailed alerts, traffic logs, and incident dashboards.
- Expect rapid alerting, intuitive filtering options, and visualization of threat trends.

2. Bank IT Administrators / CISOs (Chief Information Security Officers)

- Use the system to review compliance reports, evaluate system security posture, and manage user roles.
- Moderate technical knowledge with a focus on governance and compliance.
- Require high-level overviews, reporting features, and regulatory alignment (e.g., PCI-DSS, GDPR).
- Need clear summaries of security incidents and response effectiveness.

3. DevSecOps Engineers / System Integrators

- Responsible for system deployment, model updates, and integration with the bank's infrastructure.
- High technical proficiency in cloud platforms, Docker, Kubernetes, and security tools like Snort/Suricata.
- Require CLI/API access, logs, performance metrics, and configuration tools.

- Expect automated deployments, robust logging, and modular architecture.

4. Risk & Compliance Officers

- Monitor system reports for regulatory audits and fraud investigations.
- Limited technical background; require readable summaries and compliance dashboards.
- Use exportable audit logs, incident summaries, and regulatory checklists.
- Expect user-friendly interfaces tailored to policy management and legal standards.

5. Auditors / External Inspectors (Occasional Users)

- Access the system temporarily during audits or investigations.
- Minimal technical interaction required; need read-only access to logs and reports.
- Use system to validate data integrity and regulatory alignment.
- Expect secure, role-restricted access and downloadable summaries.

2.6 Operating Environment

The DIDS system will operate in a distributed, cloud-native environment with components deployed across containerized services and accessed via web-based dashboards or secure APIs. The infrastructure is designed for scalability, fault tolerance, and performance in financial institution networks.

Platform Requirements:

1. **Cloud Platform:** Microsoft Azure (Azure Kubernetes Service, Azure Blob Storage)
2. **Containerization:** Docker (Microservices)
3. **Orchestration:** Kubernetes (AKS)

2.7 Operating Systems:

- Ubuntu 20.04+ (for backend servers and container hosts)
- Windows 10/11 (for dashboards, analyst use)

- Kali Linux (for testing and synthetic attack generation)
- Hardware Requirements

1. For Server Nodes (Kubernetes Cluster):

- CPU: 4 vCPUs or more (per node)
- RAM: 8–16 GB (per node)
- Storage: 100 GB+ SSD (per node)
- Network: High availability with low-latency access (10 Mbps minimum)

2. For Analyst Workstations (Dashboard Access):

- Minimum RAM: 4 GB
- Browser: Chrome/Firefox with WebSocket and REST API support
- Internet: Required for cloud access and real-time dashboard communication

2.8 Software Dependencies

1. Languages & Frameworks:

- Python 3.10+ (for AI modules, backend services)
- TensorFlow / PyTorch (for anomaly detection, RL models)
- Scikit-learn, Pandas, NumPy (for data preprocessing and analytics)

2. Deployment & Infrastructure:

- Terraform (Infrastructure-as-Code)
- Docker (Containerization)
- Kubernetes (Cluster orchestration)
- Azure CLI & kubectl (Cluster management)

3. Security & Monitoring Tools:

- Snort / Suricata IDS (for signature-based detection)
- Zeek (traffic analysis)
- Kibana & Grafana (for real-time dashboards)
- OpenAI Gym (for RL simulation environments)

4. Database & Storage:

- MongoDB / PostgreSQL (for event and alert storage)
- Azure Blob Storage (for logs and backup)

5. Authentication & Access Control:

- OAuth2.0 / JWT tokens for secure API access
- Azure Active Directory (role-based access for admin and analysts)

2.9 Network & Integration

1. APIs: RESTful APIs for system modules and dashboard communication

2. External Integrations:

- IBM X-Force Exchange
- AlienVault OTX
- FraudNet Threat Feeds

3 Specific Requirements

3.1 Functional Requirements

This section outlines the detailed functional requirements for the AI & RL-Driven Distributed Intrusion Detection System (DIDS). These requirements describe the expected system behavior from the perspective of its users and ensure that all functional aspects are addressed.

3.1.1 User Authentication and Account Management

- **FR-3.1.1.1** The system shall allow new users (bank analysts, admins) to register securely with email and password.
- **FR-3.1.1.2** The system shall allow registered users to log in using their credentials.
- **FR-3.1.1.3** The system shall validate user credentials against a secure database before granting access.
- **FR-3.1.1.4** The system shall provide a password reset mechanism via verified email.
- **FR-3.1.1.5** The system shall support social authentication options, e.g., Google OAuth.
- **FR-3.1.1.6** The system shall maintain active user sessions with timeout and manual logout options.
- **FR-3.1.1.7** The system shall allow users to log out at any time securely.

3.1.2 Network Traffic Capture and Preprocessing

- **FR-3.1.2.1** The system shall continuously capture network packets from distributed banking environments including branches and ATMs.
- **FR-3.1.2.2** The system shall preprocess captured data by filtering, normalizing, and structuring it for AI analysis.
- **FR-3.1.2.3** The system shall securely store raw and preprocessed network data for audit and analysis.
- **FR-3.1.2.4** The system shall support real-time streaming of network data to detection modules.

3.1.3 Anomaly Detection and Signature-Based Detection

- **FR-3.1.3.1** The system shall analyze preprocessed network data using AI/ML models to detect anomalies in real time.
- **FR-3.1.3.2** The system shall identify known attack signatures using Snort and Suricata IDS engines.
- **FR-3.1.3.3** The system shall generate alerts for suspicious activities with threat classification and severity levels.

3.1.4 Automated Threat Response

- **FR-3.1.4.1** The system shall use Reinforcement Learning to decide on threat response actions such as alerting, blocking, or isolating affected nodes.
- **FR-3.1.4.2** The system shall log all automated responses for audit and review.

3.1.5 Threat Intelligence Integration

- **FR-3.1.5.1** The system shall fetch live threat intelligence updates from trusted external sources (e.g., IBM X-Force, OTX).
- **FR-3.1.5.2** The system shall incorporate threat intelligence data to update detection rules and enhance AI model predictions.

3.1.6 Dashboard and Reporting

- **FR-3.1.6.1** The system shall provide a real-time dashboard for security analysts showing alerts, network status, and incident trends.
- **FR-3.1.6.2** The system shall allow users to generate detailed reports for incidents, fraud detection, and compliance audits.
- **FR-3.1.6.3** The system shall support exporting reports in formats suitable for regulatory submission.

3.2 Behaviour Requirements

Use Case Diagram

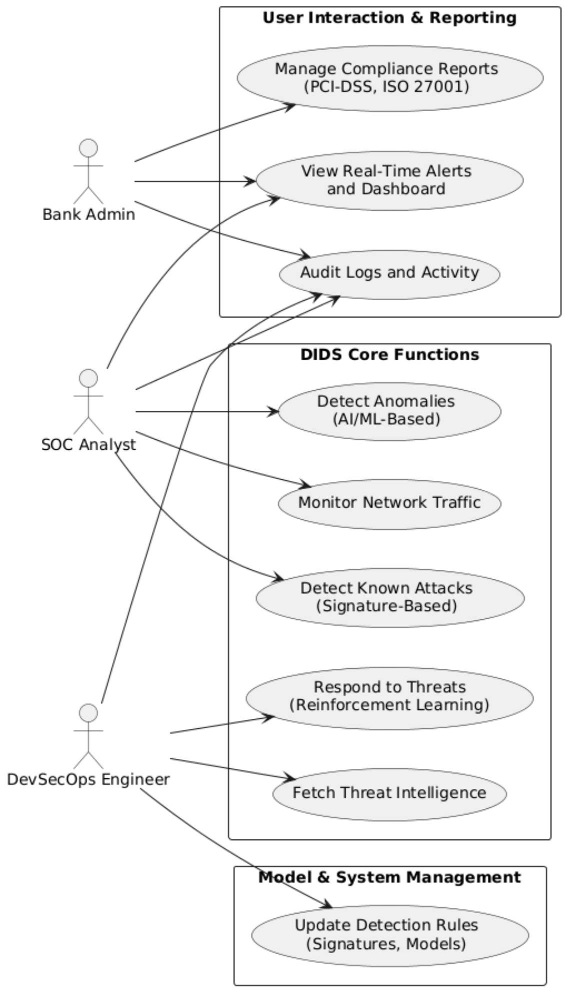
User Interaction & Reporting
Manage Compliance Reports (PCI-DSS, ISO 27001)
View Real-Time Alerts and Dashboard
Audit Logs and Activity
Bank Admin
SOC Analyst
DevSecOps Engineer
DIDS Core Functions
Detect Anomalies (AI/ML-Based)
Monitor Network Traffic
Detect Known Attacks (Signature-Based)
Respond to Threats (Reinforcement Learning)
Fetch Threat Intelligence
Model & System Management
Update Detection Rules (Signatures, Models)

Figure 3.2.1 Use Case Diagram of DIDS

Fully dressed use case

3.2.1 Upload Data

Field	**Description**
Use Case Name	Upload Data
Scenario	Security analysts or system components upload network traffic logs or transaction data
Triggering Event	User clicks “Upload” or system automatically captures network data

Field	Description
Brief Description	The system accepts uploaded network/transaction data, validates the format, and queues it for analysis
Primary Actor(s)	Security Analyst, Automated System
Supporting Actor(s)	System
Related Use Case(s)	Analyze Data, Manage User Accounts
Stakeholder(s)	Analysts: Need accurate, timely data for threat detection
Precondition(s)	User is authenticated and data is in an accepted format

3.2.2 *Analyze Data*

Field	Description
Use Case Name	Analyze Data
Scenario	System processes uploaded network and transaction data to detect anomalies and threats
Triggering Event	Valid data file is uploaded or data stream is captured
Brief Description	The system runs AI/ML and signature-based detection models on input data, identifies threats, and stores results
Primary Actor(s)	System
Supporting Actor(s)	None
Offstage Actor(s)	User
Related Use Case(s)	Upload Data, View Alerts
Stakeholder(s)	Security Team: Needs reliable, actionable threat analysis
Precondition(s)	Data is available and validated

3.2.3 View Alerts and Reports

Field	**Description**
Use Case Name	View Alerts and Reports
Scenario	User reviews detection results and threat alerts
Triggering Event	Analysis completes and results are ready for viewing
Brief Description	The system displays threat alerts, anomaly scores, and detailed logs through a user dashboard
Primary Actor(s)	Security Analyst
Supporting Actor(s)	System
Related Use Case(s)	Analyze Data, Download Reports
Stakeholder(s)	Analysts: Require comprehensive and clear information for response planning
Precondition(s)	Analysis is completed and results stored

3.2.4 Download Reports

Field	**Description**
Use Case Name	Download Reports
Scenario	User downloads detailed security incident or anomaly reports
Triggering Event	User selects “Download Report” option
Brief Description	The system generates downloadable reports (PDF, CSV, JSON) summarizing threat detections
Primary Actor(s)	Security Analyst
Supporting	System

Field	Description
Actor(s)	
Related Use Case(s)	View Alerts and Reports
Stakeholder(s)	Analysts: Need to keep records or share with other stakeholders
Precondition(s)	Reports are generated and user is authenticated

3.2.5 Manage User Accounts

Field	Description
Use Case Name	Manage User Accounts
Scenario	Users register, log in, reset passwords, and manage profiles
Triggering Event	User navigates to authentication or profile management interface
Brief Description	The system facilitates secure registration, login, password recovery, and profile updates
Primary Actor(s)	User
Supporting Actor(s)	System
Related Use Case(s)	Upload Data, View Alerts
Stakeholder(s)	Users: Require secure and personalized access
Precondition(s)	User is on authentication or profile management pages

3.3 External Interface Requirements

3.3.1 User Interfaces

The DIDS system provides a user-friendly web-based dashboard and interfaces designed for security analysts, administrators, and system operators. The UI is designed for clarity and efficiency, enabling users to monitor, analyze, and respond to cybersecurity threats within banking networks.

3.3.2 Key UI Screens and Components:

1. **Login/Registration Screen**

- Fields for username/email and password input.
- Support for multi-factor authentication (MFA).
- Forgot password and account recovery options.
- Feedback on login success, failure, or invalid input via messages.

2. **Dashboard / Home Screen**

- Overview of system status with summary alerts and metrics.
- Real-time threat feed showing current alerts and incident severity.
- Navigation menus for modules like Traffic Logs, Anomaly Detection, Reports, and Settings.

3. **Traffic Capture & Analysis Screen**

- Interface to view captured network traffic summaries and logs.
- Filtering options by date, source, severity, and type of threat.
- Real-time data stream display and search functionality.

4. **Anomaly Detection & Alerts Screen**

- Visual list of detected anomalies with details such as timestamp, affected system, and risk score.
- Ability to drill down into incident details and historical trends.
- Buttons to acknowledge, escalate, or dismiss alerts.

5. **Automated Response & Incident Management Screen**

- View actions taken by the RL-based automated system.

- Manual override controls for administrators to intervene or tune response rules.

6. **Reports and Compliance Screen**

- Generate and download reports related to security incidents, compliance audits, and system performance.
- Visualization charts for trends in attacks, detection rates, and system uptime.

7. **User Profile & Settings Screen**

- Manage user credentials, roles, and permissions.
- Configure system preferences such as notification settings and API integrations.
- View system version and support contact info.

3.3.3 Hardware Interfaces

1. **Network Sensors and Capture Devices:**

- Packet sniffers and traffic sensors deployed at bank branches, ATMs, and data centers.
- These devices capture network traffic for real-time analysis.

2. **Server and Cloud Infrastructure:**

- Virtual machines and Kubernetes nodes running AI models, signature detection, and dashboards.
- Storage systems for logs and event data.

3. **User Workstations:**

- Analyst computers accessing the dashboard via browsers.
- Standard desktop/laptop hardware with internet connectivity.

3.4 Software Interfaces

Component	Details
Operating System	Linux (Ubuntu 20.04+ for backend), Windows 10/11 (for dashboards)
Backend Framework	Python (Flask/FastAPI) for API services
AI Libraries	TensorFlow, PyTorch, Scikit-learn, OpenAI Gym
Intrusion Detection Systems	Snort, Suricata, Zeek
Database	MongoDB / PostgreSQL for event and user data
Cloud Platform	Microsoft Azure (AKS, Blob Storage, Azure Active Directory)
Deployment Tools	Docker, Kubernetes, Terraform
APIs	RESTful APIs for communication between components and external threat intelligence feeds
Dashboard	React.js / Angular based frontend

3.5 Communications Interfaces

- **Communication Standards:** All data exchanges between clients, servers, and cloud services use HTTPS with TLS encryption to ensure confidentiality and integrity.
- **Protocols:** REST APIs serve as the primary interface for inter-module communication and external integrations such as threat intelligence feeds (IBM X-Force, OTX).
- **Data Transfer Optimization:** The system leverages data compression and incremental updates to support low latency and efficient real-time analytics.
- **Alerting:** Notifications to security teams are transmitted securely via email or integrated messaging platforms (e.g., Microsoft Teams, Slack).
- **Authentication:** OAuth 2.0 and JWT tokens secure API access and user sessions

4 Other Non-functional Requirements

4.1 Performance Requirements

- NFR-4.1.1 The system shall return intrusion detection results within 10 seconds under normal network conditions.

- NFR-4.1.2 Real-time network traffic monitoring shall process data at a rate sufficient to analyze at least 1000 packets per second.
- NFR-4.1.3 User authentication (login, signup) shall complete within 3 seconds.
- NFR-4.1.4 The system shall support a minimum of 100 concurrent users (security analysts and admins) accessing the dashboard simultaneously.
- NFR-4.1.5 AI and RL models shall load and initialize in under 3 seconds on cloud infrastructure.
- NFR-4.1.6 The microservices shall auto-scale to handle variable network traffic volumes without performance degradation.

4.2 Safety and Security Requirements

- NFR-4.2.1 All communications between client dashboards, backend services, and cloud APIs shall be encrypted using TLS (HTTPS).
- NFR-4.2.2 User authentication and role-based access control shall be enforced using OAuth 2.0 and Azure Active Directory integration.
- NFR-4.2.3 Unauthorized access to network data, logs, or control modules shall be prevented by strict access controls and audit logging.
- NFR-4.2.4 Sensitive log data and detection results shall be stored encrypted at rest in cloud storage solutions (e.g., Azure Blob Storage).
- NFR-4.2.5 Detection results and network logs shall only be accessible to authenticated users with appropriate permissions.
- NFR-4.2.6 The system shall comply with banking data security regulations including PCI-DSS, GDPR, and ISO 27001.
- NFR-4.2.7 All inputs and external data sources shall be validated to prevent injection attacks, data corruption, and tampering.

- NFR-4.2.8 Security incident alerts and notifications shall be transmitted securely, minimizing risk of interception or spoofing.

4.3 Software Quality Attributes

4.3.1 Reliability

- NFR-4.3.1.1 AI and RL detection models shall be trained on up-to-date datasets (e.g., CICIDS2018, UNSW-NB15 2024) to minimize false positives/negatives.
- NFR-4.3.1.2 The system shall implement redundancy and failover mechanisms to ensure continuous monitoring and alerting.

4.3.2 Portability

- NFR-4.3.2.1 The system components shall run on cloud platforms supporting Linux containers (e.g., Azure Kubernetes Service).
- NFR-4.3.2.2 Dashboard clients shall support common browsers on Windows, Linux, and macOS.

4.3.3 Maintainability

- NFR-4.3.3.1 Modular architecture with clear separation of AI, data collection, and dashboard components.
- NFR-4.3.3.2 Source code will be version-controlled in GitHub with detailed documentation.
- NFR-4.3.3.3 Updates shall follow semantic versioning with CI/CD pipelines for automated testing and deployment.

4.3.4 Usability

- NFR-4.3.4.1 Dashboard UI shall be designed with intuitive workflows for security analysts and administrators.

- NFR-4.3.4.2 Alerts and logs shall be clearly categorized by severity, with easy navigation and filtering options.

4.3.5 Testability

- NFR-4.3.5.1 Unit and integration tests will cover AI inference pipelines, data ingestion, and API endpoints.
- NFR-4.3.5.2 System performance and load tests will validate scalability under realistic banking network traffic volumes.

4.3.6 Availability

- NFR-4.3.6.1 The cloud infrastructure shall provide 99.9% uptime for all critical services including detection engines, data storage, and dashboards.

4.3.7 Adaptability

- NFR-4.3.7.1 The system architecture will allow integration of new detection models and threat intelligence sources without major rede

5 Design Description

5.1 Composite Viewpoint

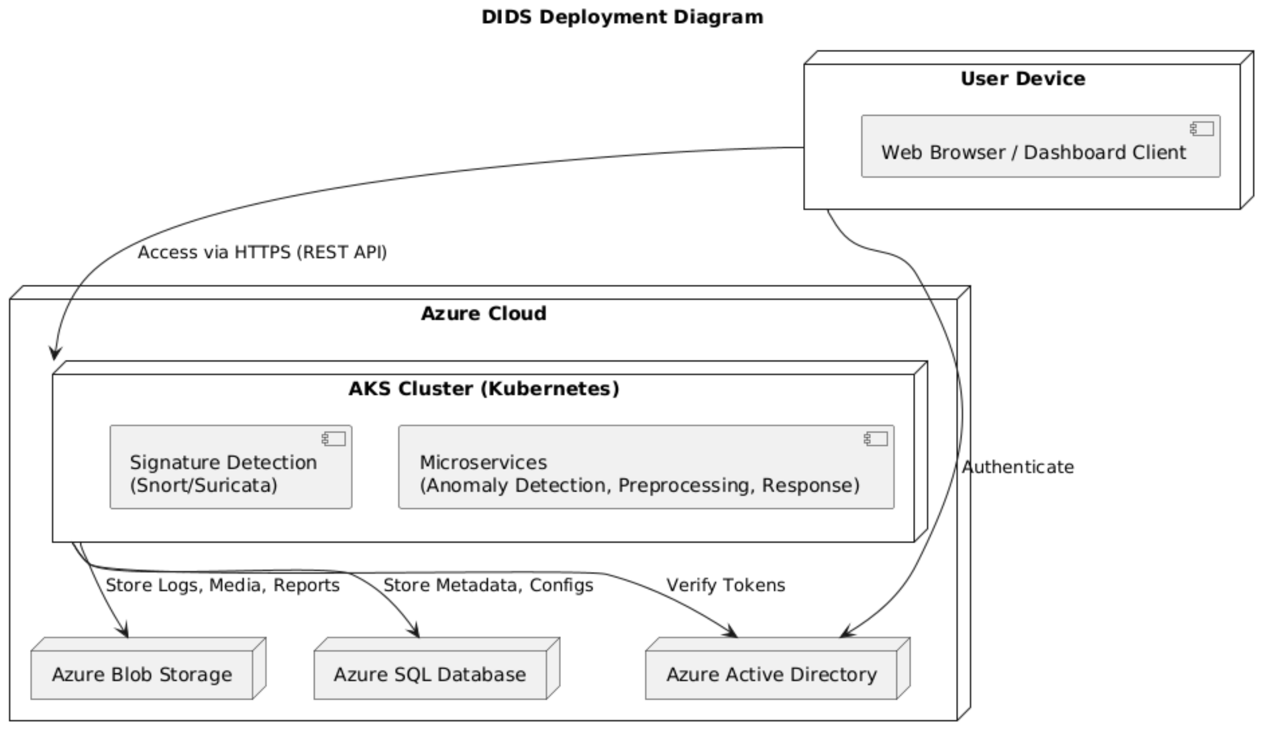
DIDS Deployment Diagram
User Device
Web Browser / Dashboard Client
Access via HTTPS (REST API)
Authenticate
Azure Cloud
AKS Cluster (Kubernetes)
Signature Detection (Snort/Suricata)
Microservices (Anomaly Detection, Preprocessing, Response)
Store Logs, Media, Reports
Store Metadata, Configs
Verify Tokens
Azure Blob Storage
Azure SQL Database
Azure Active Directory

Figure 5.1 deployment diagram of DIDS

5.2 Logical Viewpoint

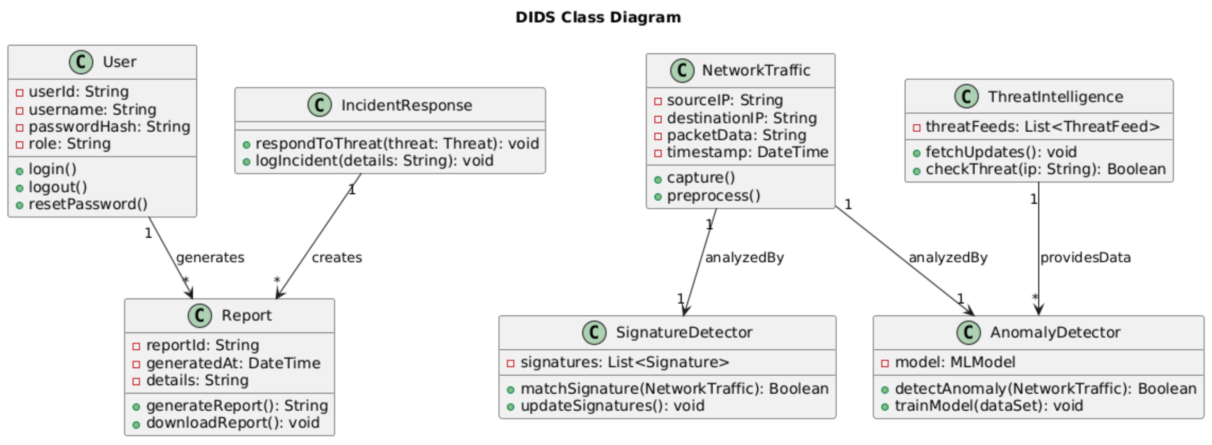

Figure 5.2 Class diagram of DIDS

5.3 Information Viewpoint

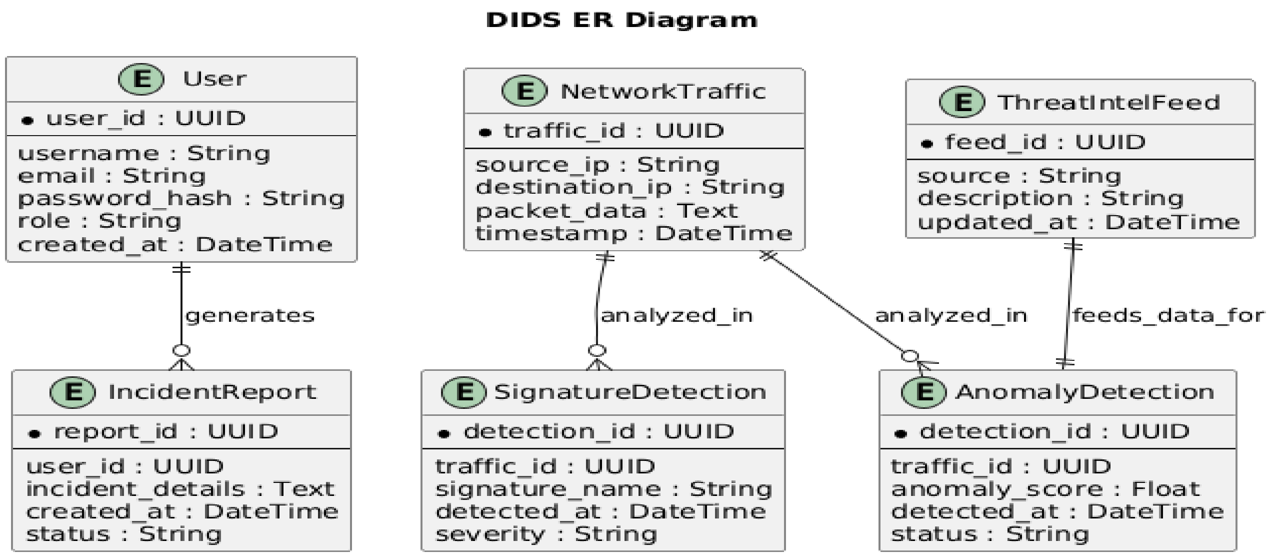

Figure 5.3 Entity-Relationship Diagram for DIDS

5.4 Interaction Viewpoint

5.4.1. User registration and Login Logout:

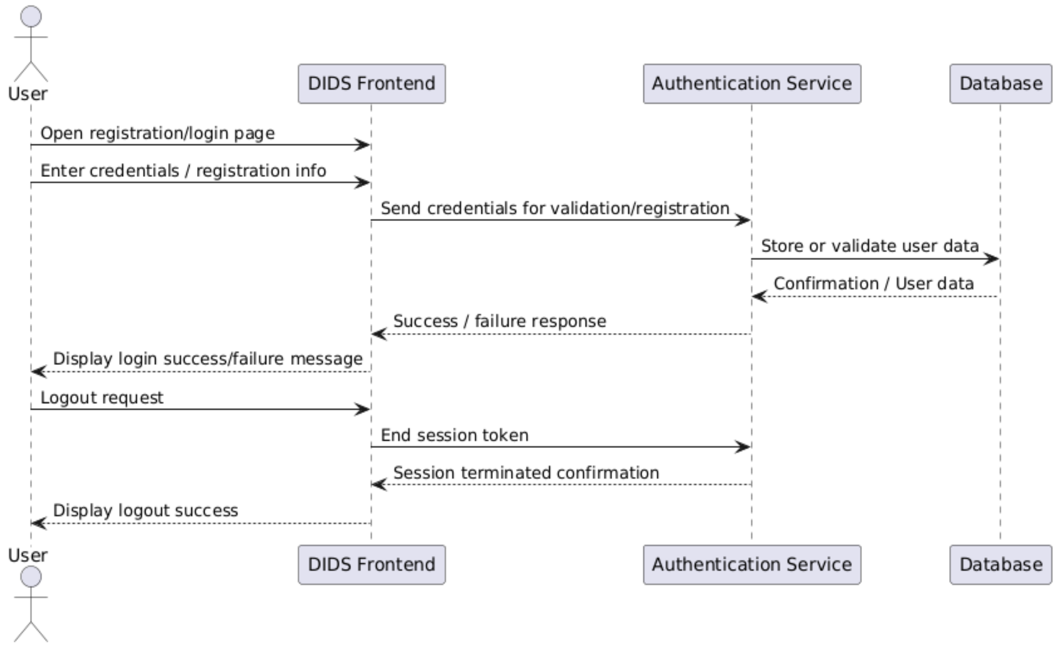
User
DIDS Frontend
Authentication Service
Database
Open registration/login page
Enter credentials / registration info
Send credentials for validation/registration
Store or validate user data
Confirmation / User data
Success / failure response
Display login success/failure message
Logout request
End session token
Session terminated confirmation
Display logout success
User
DIDS Frontend
Authentication Service
Database

Figure 5.4.1 User registration, login and logout

5.4.2. Traffic Capure & Preprocessing:

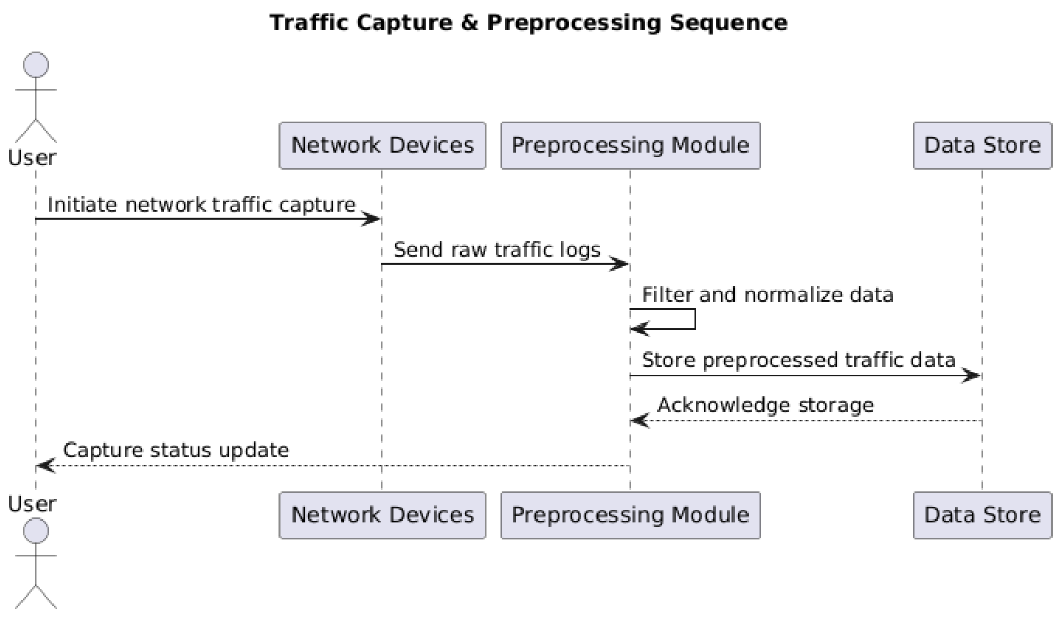
Traffic Capture & Preprocessing Sequence
User
Network Devices
Preprocessing Module
Data Store
Initiate network traffic capture
Send raw traffic logs
Filter and normalize data
Store preprocessed traffic data
Acknowledge storage
Capture status update

Figure 5.4.2 Traffic Capture & Preprocessing

5.4.3. AI & RL-Based Anomaly Detection

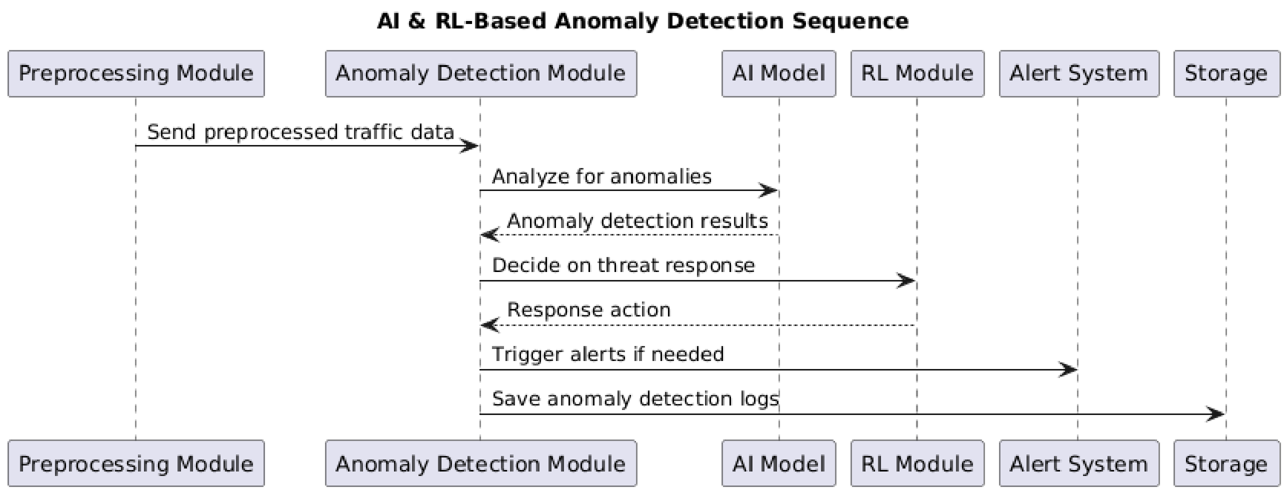
AI & RL-Based Anomaly Detection Sequence
Preprocessing Module
Anomaly Detection Module
AI Model
RL Module
Alert System
Storage
Send preprocessed traffic data
Analyze for anomalies
Anomaly detection results
Decide on threat response
Response action
Trigger alerts if needed
Save anomaly detection logs

Figure 5.7 AI & RL-Based Anomaly Detection

5.4.4. Signature-Based Detection:

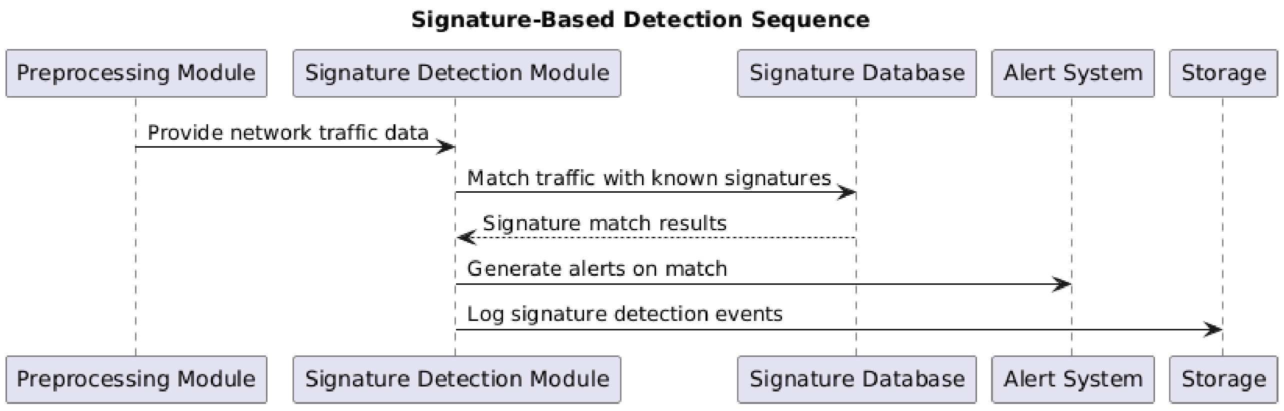
Signature-Based Detection Sequence
Preprocessing Module
Signature Detection Module
Signature Database
Alert System
Storage
Provide network traffic data
Match traffic with known signatures
Signature match results
Generate alerts on match
Log signature detection events
Preprocessing Module
Signature Detection Module
Signature Database
Alert System
Storage

Figure 5.4.4 Signature-Based Detection

5.4.5 Threat Intelligence Integration

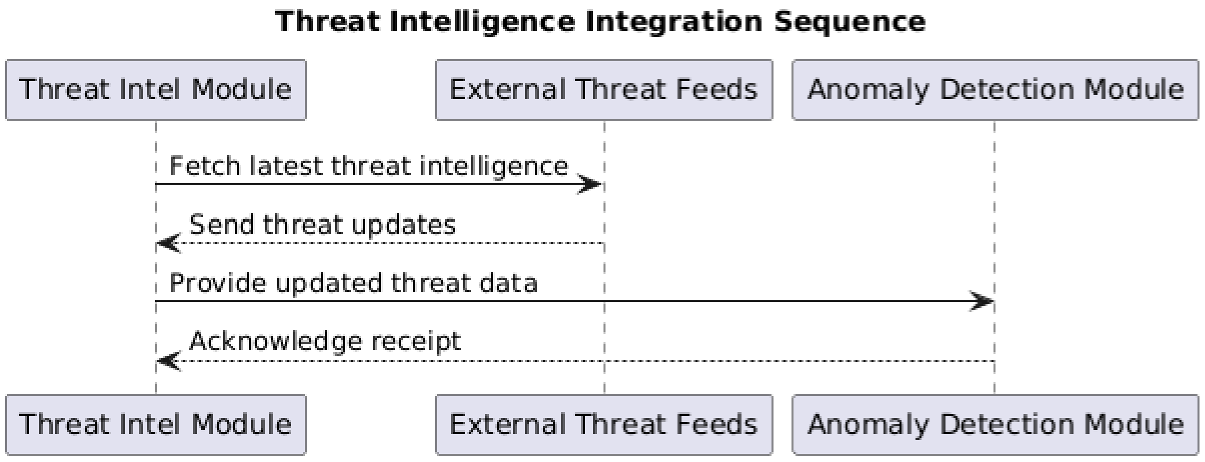

Figure 5.4.5 Threat Intelligence Integration

5.4.6. Automated Threat Response & Incident Management

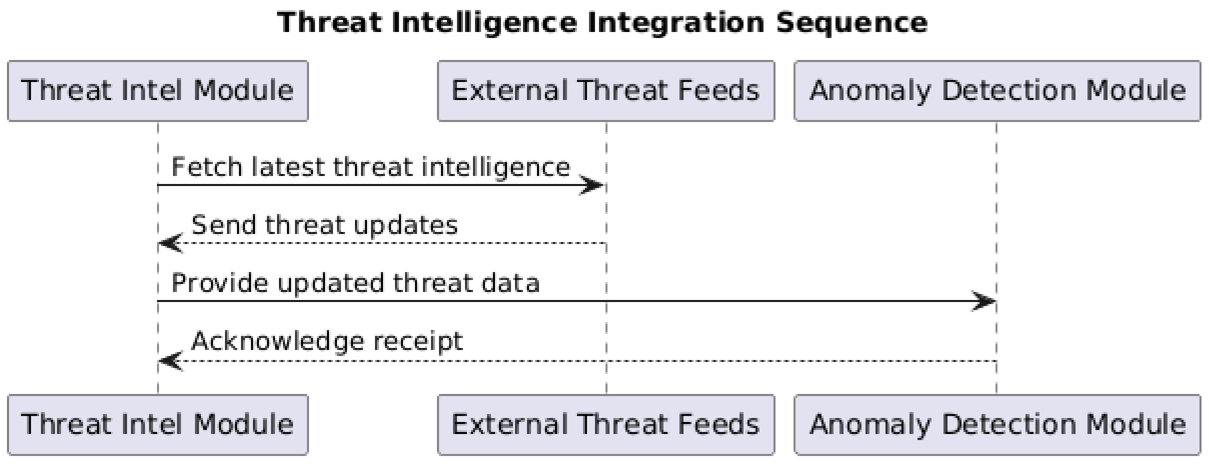

Figure 5.4.6 Threat Intelligence Integration

5.4.7. Regulatory Compliance & Security Auditing

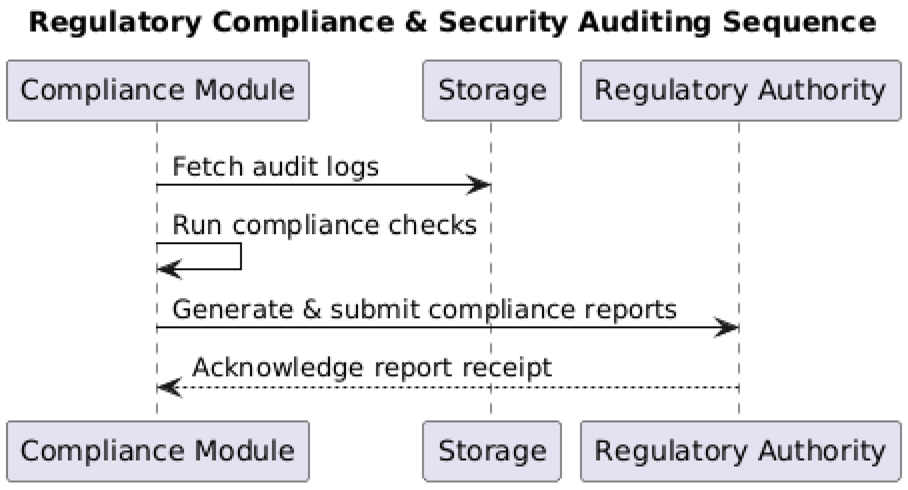

Figure 5.4.7 Regulatory Compliance & Security Auditing

5.4.8. Dashboard & Reporting Module

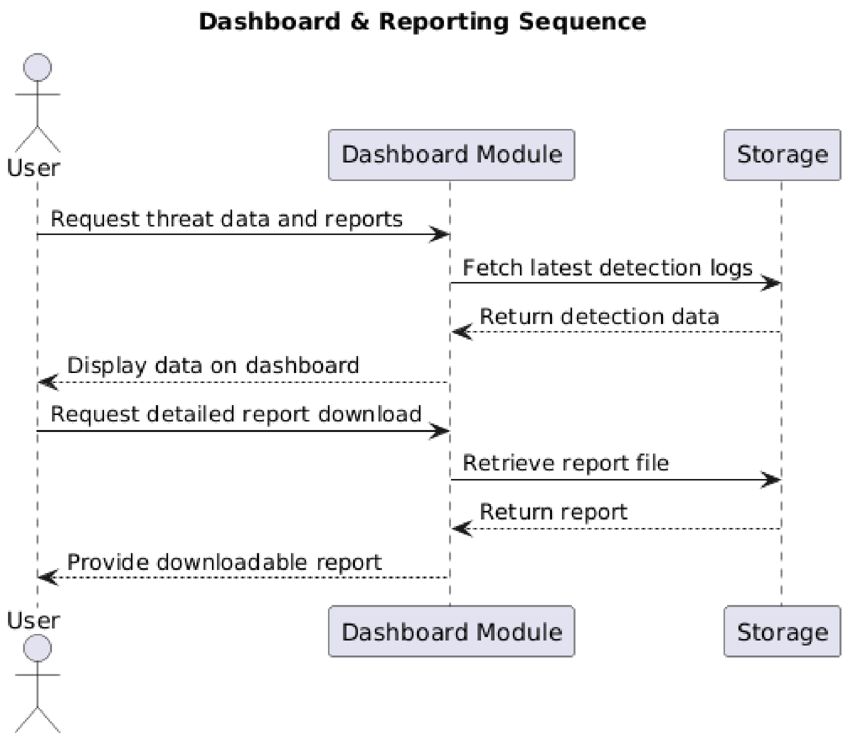
Dashboard & Reporting Sequence
User
Dashboard Module
Storage
Request threat data and reports
Fetch latest detection logs
Return detection data
Display data on dashboard
Request detailed report download
Retrieve report file
Return report
Provide downloadable report
User
Dashboard Module
Storage

Figure 5.4.8 Dashboard & Reporting Module

5.4.9. Cloud-Based Microservices Deployment (Conceptual)

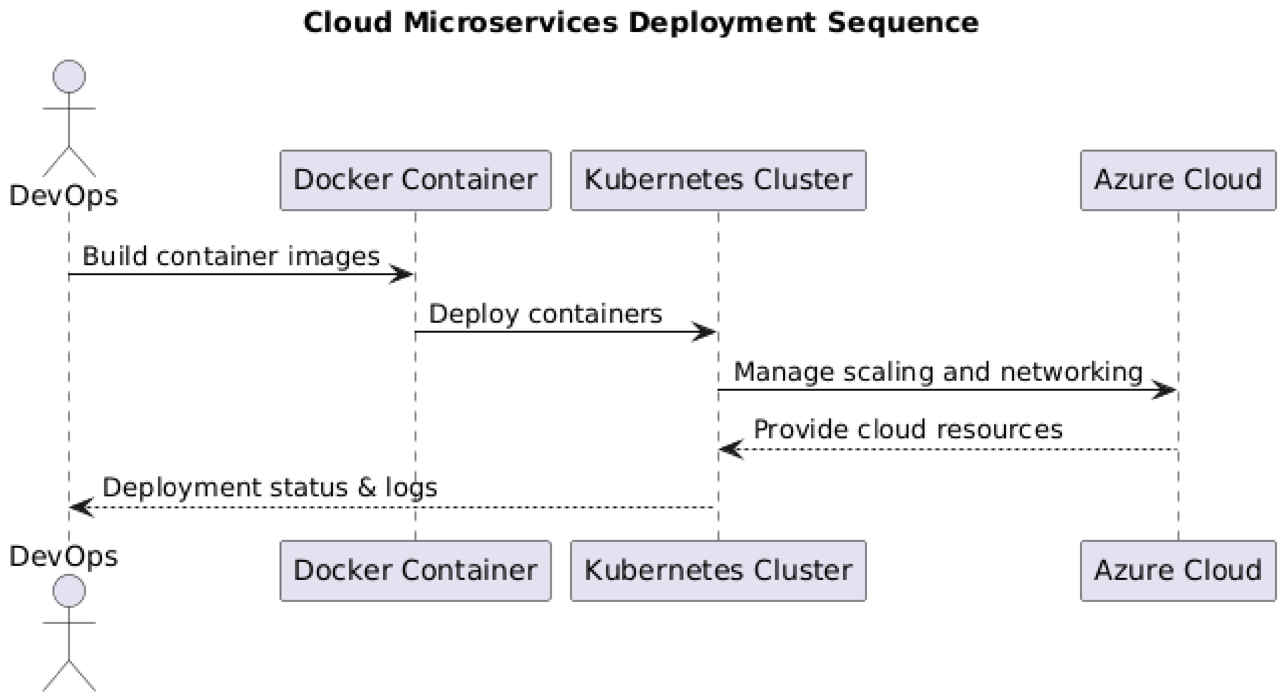

Figure 5.4.9 Cloud-Based Microservices Deployment (Conceptual)

5.3 State Dynamics Viewpoint

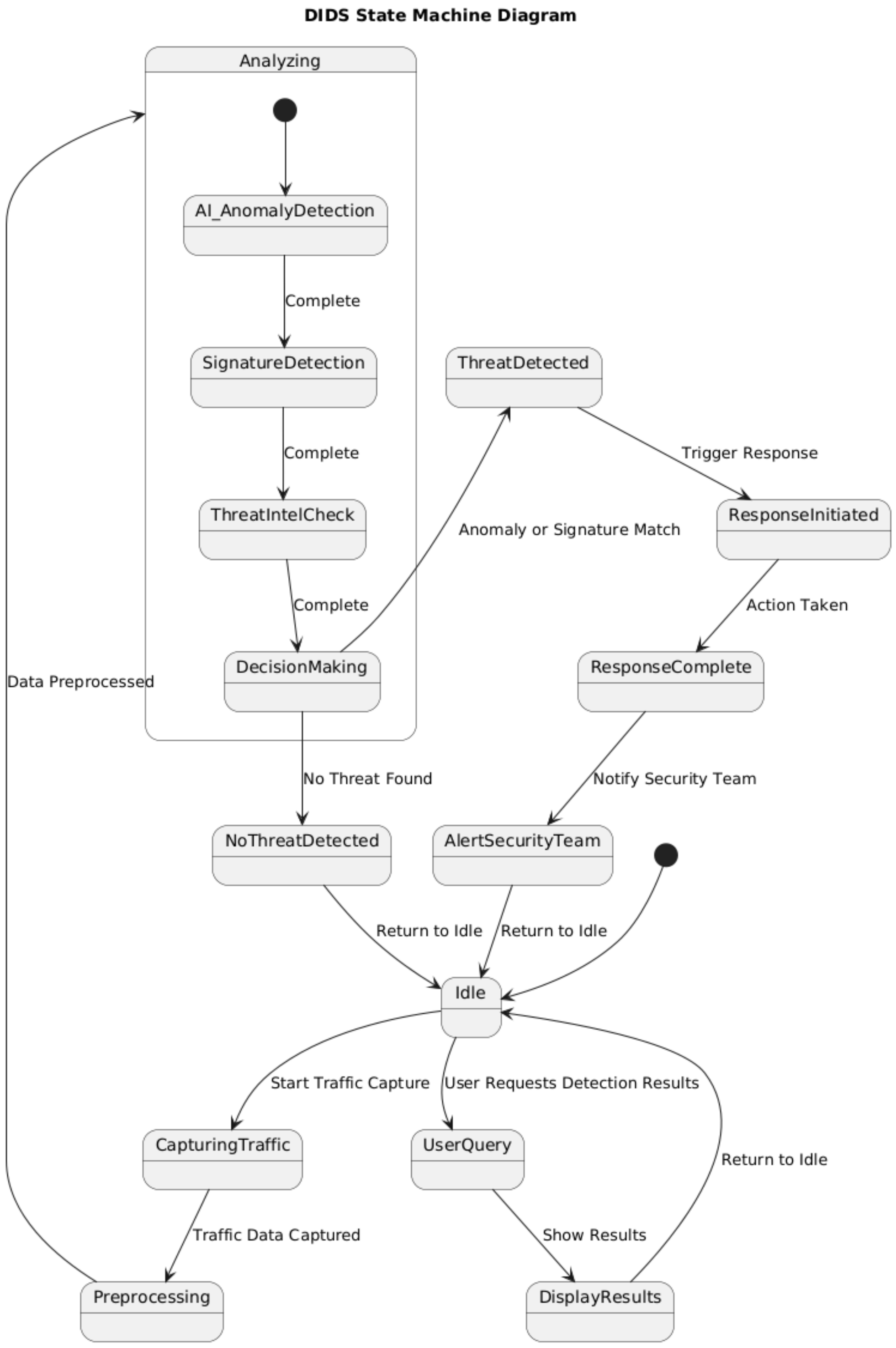

Figure 5.5 State machine diagram of DIDS

5.5. Algorithm Viewpoint

1. **Pseudocode Authentication:**

Authenticate (email,password)

```
function authenticateUser(email, password):
    if not validateEmailFormat(email):
        return "Invalid Email"

    user = Database.getUserByEmail(email)
    if user is None:
        return "User does not exist"

    if not verifyPassword(password, user.hashedPassword):
        return "Incorrect Password"

    session.start(user.id)
    return "Login Successful"
```

2. **Pseudocode Traffic Capture & Preprocessing:**

```
function captureNetworkTraffic(deviceSource):
    if not session.isLoggedIn():
        return "Unauthorized Access"

    rawPackets = NetworkSniffer.capture(deviceSource)
    filteredPackets = removeNoise(rawPackets)
    formattedPackets = normalizePackets(filteredPackets)

    Database.storePreprocessedTraffic(formattedPackets)
    return formattedPackets
```

3. Pseudocode AI & RL-Based Anomaly Detection:

```
function detectAnomaly(trafficData):
    model = MLModel.load("XGBoost_Model")
    rlAgent = RLModel.load("DQN_Model")

    prediction = model.predict(trafficData)
    confidenceScore = prediction.confidence
    isThreat = prediction.label

    decision = rlAgent.evaluate(prediction)

    detection = new Detection()
    detection.traffic = trafficData.id
    detection.isThreat = isThreat
    detection.confidence = confidenceScore
    detection.action = decision.action

    Database.saveDetection(detection)
    return detection
```

4. Pseudocode for Signature-Based Detection (Snort/Suricata Integration):

```
function runSignatureBasedScan(preprocessedData):
    snortResult = Snort.scan(preprocessedData)
    suricataResult = Suricata.scan(preprocessedData)

    threats = mergeResults(snortResult, suricataResult)

    if threats.isEmpty():
        return "No known threat detected"

    Database.logThreats(threats)
    return threats
```

5. **Pseudocode View Detection Results:**

```
function viewDetections(userId):
    if not session.isLoggedIn():
        return "Access Denied"

    records = Database.getDetectionsByUser(userId)

    for detection in records:
        print("Traffic ID: " + detection.traffic)
        print("Threat: " + detection.isThreat)
        print("Confidence: " + detection.confidence)
        print("Action Taken: " + detection.action)
        print("Timestamp: " + detection.timestamp)

    return "All Detection Logs Shown"
```

6. **Pseudocode View Detection Results:**

```
function downloadDetectionReport(userId, detectionId):
    if not session.isLoggedIn():
        return "Login Required"
```

```
report = Database.getDetectionById(detectionId)
if report is None or report.userId != userId:
    return "Report not found"

fileContent = formatReport(report)
filePath = FileSystem.generateFile("report_" + report.traffic + ".txt", fileContent)
return filePath
```