Real-Time AI Intrusion Detection System Using Machine Learning

Silambarasan B[1], Sarveswaran S[2], Sharan S.P[3], Pradeep N[4], Ravindra krishna chandar V[5]
[1, 2, 3, 4] Dept of Computer cyber security
[5]HOD, Dept of Computer cyber security
[1, 2, 3, 4, 5] Dhanalakshmi Srinivasan University

Abstract- *The relentless escalation of cyber threats in contemporary network environments has necessitated a fundamental rethinking of intrusion detection methodologies. Traditional signature-based systems, while once sufficient, have proven structurally incapable of addressing polymorphic attacks, zero-day exploits, and the sheer volume of anomalous traffic generated by modern enterprise networks. This paper presents a Real-Time AI Intrusion Detection System (RT-AI-IDS) developed using a hybrid machine learning framework that combines unsupervised anomaly detection with supervised attack classification. The proposed system employs the Isolation Forest algorithm for identifying statistical outlier indicative of previously unseen attack behaviour and a Random Forest classifier for precise categorisation of known attack classes using the benchmark NSL-KDD dataset. The system is architected as a full-stack web application, with a FastAPI -powered backend exposing RESTful endpoints for real-time prediction, alert generation, and intrusion logging, coupled with a React and Vite-based monitoring dashboard providing live visualisation of traffic patterns and detection events. Persistent storage of prediction histories, alert records, and intrusion logs is managed through a lightweight SQLite database, enabling retrospective analysis and operational transparency. Experimental evaluation demonstrates high classification accuracy, low false-positive rates, and real-time latency suitable for production deployment. The architecture is explicitly designed for modularity and horizontal scalability, allowing the system to be extended with additional detection models, data sources, and alerting channels without architectural disruption. The proposed system represents a meaningful contribution to the practical deployment of AI -based intrusion detection in resource-constrained and enterprise-grade environments alike.*

Keywords: Intrusion detection system; machine learning; Isolation Forest; Random Forest; NSL-KDD; anomaly detection; FastAPI; cybersecurity; real-time monitoring; network security

I. INTRODUCTION

The modern digital infrastructure underpins critical operations in government, finance, healthcare, education, and industrial control, making the security of network systems a matter of national and organisational priority. Cyberattacks have evolved from simple denial-of-service attempts to sophisticated, multi-stage intrusion campaigns employing advanced persistent threats (APTs), lateral movement techniques, and AI-augmented exploit delivery. The global cybersecurity landscape recorded over 5. 5 billion malware attacks in a single year according to SonicWall's annual threat report, with intrusion incidents targeting enterprise networks increasing by over 28% year-on-year. Against this backdrop, the adequacy of conventional intrusion detection systems (IDS) must be critically evaluated.

Signature-based IDS tools such as Snort and Suricata operate by matching observed network traffic against libraries of known attack signatures. While effective for previously catalogued threats, these systems fail fundamentally when confronted with novel attack variants, zero-day exploits, or encrypted traffic patterns that do not match any stored rule. The maintenance burden of continuously updating signature databases introduces operational latency, during which newly discovered attack vectors remain undetected. Statistical anomaly detection methods, while theoretically capable of identifying unknown threats by flagging deviations from baseline behaviour, suffer from high false-positive rates that overwhelm security operations teams and erode analyst confidence in system outputs.

The emergence of machine learning as a practical engineering discipline, rather than a purely theoretical field, has opened new possibilities for intrusion detection that transcend the limitations of rule-based approaches. Ensemble learning methods, in particular Random Forest classifiers, have demonstrated robust generalisation performance across heterogeneous network traffic datasets, achieving high precision and recall on benchmark evaluations. Concurrently, unsupervised techniques such as Isolation Forest have shown particular aptitude for anomaly detection in high-dimensional

feature spaces, identifying statistical outliers with computational efficiency superior to density-based alternatives. The combination of these complementary approaches within a unified detection pipeline represents a compelling architectural direction for nextgeneration IDS.

The proposed Real-Time AI Intrusion Detection System (RT-AI-I DS) operationalises this dual-model approach within a production-grade web application architecture. The system ingests network traffic feature vectors, subjects them to preprocessing and normalisation, passes them through sequential anomaly detection and attack classification stages, and delivers detection results to a live monitoring dashboard while simultaneously persisting alerts and logs to a structured database. The entire pipeline is designed for real-time operation, with API response latencies suitable for deployment alongside live network monitoring agents. This paper documents the design rationale, architectural decisions, implementation details, experimental evaluation, and future development directions of the proposed system.

II. BACKGROUND OF INTRUSION DETECTION SYSTEMS

Evolution of Network Intrusion Detection

Intrusion detection emerged as a formal cybersecurity discipline in the mid-1 980s with Dorothy Denning' s foundational work on real-time security monitoring systems. Early intrusion detection systems mainly focused on host-level audit logs and system activity monitoring. With the growth of TCP/IP networking, network-based intrusion detection systems became widely adopted for monitoring suspicious network traffic and unauthorized activities. The two major detection approaches are signature-based detection and anomaly-based detection. Signature-based systems detect known attack patterns with high accuracy but cannot identify unknown threats, while anomaly-based systems detect unusual behaviour patterns to identify potential intrusions, often with higher false-positive rates. Modern cybersecurity systems increasingly combine both approaches using Machine Learning techniques to improve detection accuracy and real-time threat analysis.

Limitations of Traditional IDS Approaches

Signature-based intrusion detection systems are limited by their dependence on predefined signature databases. They can effectively detect known attacks but fail to identify zero-day exploits, polymorphic malware, and encrypted malicious traffic that does not match existing signatures. As cyber threats become more advanced, these limitations reduce the effectiveness of traditional signature-based detection methods. Anomaly-based systems attempt to detect unusual network behaviour by creating models of normal activity. However, accurately defining normal behaviour in complex enterprise environments is difficult due to changing traffic patterns and diverse workloads. Highly sensitive models may generate false positives, while less sensitive models may miss actual attacks. Machine Learning techniques help overcome these challenges by enabling adaptive and data-driven anomaly detection for modern cybersecurity environments.

III. AI IN CYBERSECURITY MONITORING

The application of Artificial Intelligence (AI) and Machine Learning (ML) in cybersecurity has grown rapidly due to the limitations of traditional rule-based detection systems against modern cyber threats. ML-based intrusion detection systems use supervised, unsupervised, semi-supervised, and deep learning approaches to identify malicious activities and abnormal network behaviour. Algorithms such as Random Forest, Support Vector Machines, Isolation Forest, and Deep Neural Networks are widely used for threat detection and anomaly analysis. Supervised learning models classify network traffic using labelled datasets, while unsupervised methods detect anomalies by identifying deviations from normal behaviour without labelled data. Modern AI-based intrusion detection systems also emphasise real-time monitoring, low-latency prediction, scalability, and interpretable threat analysis for practical deployment in enterprise environments.

IV. LITERATURE REVIEW

A. ML-Based Intrusion Detection: Foundational Work

Buczak and Guven published a comprehensive survey on data mining and Machine Learning methods used for cybersecurity intrusion detection [1]. The study examined algorithms such as decision trees, neural networks, support vector machines, and ensemble methods using benchmark datasets like KDD Cup 1 999 and NSL-KDD. The survey concluded that ensemble learning methods provide better detection accuracy and lower false-positive rates compared to single-classifier approaches. These findings support the use of Isolation Forest and Random Forest in modern intrusion detection systems. Tavallaee et al. introduced the NSL-KDD dataset as an improved version of the KDD Cup 1 999 dataset by removing redundant records and improving evaluation reliability [2]. The dataset includes 41 network traffic features and multiple attack categories including DoS, Probe, R2L, and U2R, making it one of the most widely used benchmark

datasets for intrusion detection and anomaly detection research.

Ensemble Learning in Network Security

Breiman introduced the Random Forest algorithm as an ensemble-based classification technique that combines multiple decision trees to improve prediction accuracy and reduce variance [3]. In cybersecurity intrusion detection, Random Forest provides stable and reliable classification performance across different network traffic patterns, making it effective for detecting malicious activities in diverse environments. Liu et al. proposed the Isolation Forest algorithm for efficient anomaly detection in high-dimensional datasets [4]. The algorithm identifies anomalies by isolating abnormal data points through recursive partitioning of the feature space. Since anomalous traffic patterns are statistically rare, they are isolated more quickly than normal traffic, enabling efficient and real-time anomaly detection for network intrusion monitoring systems.

Deep Learning Approaches to IDS

Javaid et al. proposed a deep learning-based intrusion detection system using sparse autoencoders for learning compressed feature representations of network traffic [5]. Their approach improved detection performance for rare attack categories where labelled data is limited. However, deep learning models often require high computational resources and provide lower interpretability compared to ensemble-based Machine Learning methods. Yin et al. applied Long Short-Term Memory (LSTM) networks for intrusion detection to capture temporal patterns in sequential network traffic [6]. Their approach improved detection of multi-stage attacks by analyzing the sequence of network events over time. Although LSTM-based models achieved strong detection performance, they required large training datasets and higher computational power, making real-time deployment more challenging compared to efficient tree-based models such as Random Forest and Isolation Forest.

Hybrid and Multi-Stage Detection Frameworks

Khraisat et al. conducted a survey on hybrid intrusion detection systems, categorizing architectures into sequential, parallel, and hierarchical detection models [7]. Their research showed that hybrid approaches combining anomaly detection and attack classification provide better performance than single-method intrusion detection systems, especially for detecting both unknown attacks and known threat categories. Moustafa and Slay introduced the UNSW-NB1 5 dataset and a hybrid detection framework combining statistical analysis with Machine Learning classification [8]. Their work emphasized the importance of feature engineering and data preprocessing techniques such as feature selection, label encoding, and normalization for improving intrusion detection accuracy and Machine Learning model performance.

Real-Time IDS Architectures

Liao et al. surveyed intrusion detection system technologies with emphasis on real-time deployment challenges such as processing throughput, detection latency, and scalability [9]. The study highlighted that rule-based systems provide faster processing but limiteddetection capabilities, while Machine Learning-based systems offer better threat detection with higher computational requirements. The proposed system addresses these challenges through pre-trained Machine Learning models and efficient model serialization techniques for supporting fast real-time intrusion detection.

V. COMPARATIVE ANALYSIS OF EXISTING SYSTEMS

A comparative analysis of existing intrusion detectio

n system approaches reveals common strengths and limitations across different cybersecurity frameworks. Table I compares representative IDS models based on important factors such as detection methodology, datasets used, real-time monitoring capability, attack classification accuracy, deployment architecture, and false-positive rates. The comparison highlights the growing importance of Machine Learning-based intrusion detection systems for improving real-time threat detection and cybersecurity monitoring performance.

TABLE I: Comparative Analysis of Existing IDS Approaches

System/Study	Approach	Dataset	Real-Time	Attack Classes
Snort (Rule-Based)	Signature	Custom	Yes	Known Only
Buczak & Guven [1]	ML Survey	NSL-KDD	No	Multi-class

Javaid et al. [5]	Deep Learning	NSL-KDD	No	Multi-class
Yin et al. [6]	LSTM	NSL-KDD	Partial	Multi-class
Khraisat et al. [7]	Hybrid	Multiple	Partial	Multi-class
Proposed RT-AIIDS	IF + RF Hybrid	NSL-KDD	Yes	Multi-class + Anomaly

The comparative analysis shows that existing intrusion detection systems often face trade-offs between detection accuracy, false-positive rates, and real-time deployment capability. Signature-based systems support fast real-time detection but fail to identify unknown threats, while deep learning approaches provide improved detection accuracy with higher computational requirements and inference latency. The proposed RT-AI-IDS addresses these challenges by combining efficient tree-based Machine Learning models with a pre-trained and serialized inference pipeline, enabling both real-time intrusion detection and effective identification of known and unknown cyber threats.

VI. LIMITATIONS OF EXISTING SYSTEMS

The systematic review of existing intrusion detection systems reveals several persistent limitations in modern cybersecurity environments. Signature-based systems are limited to detecting known threats and often fail against zero-day and polymorphic attacks. Many Machine Learning-based systems also struggle to provide real-time detection due to higher computational requirements and processing delays. In addition, several existing IDS frameworks use monolithic architectures that make system updates and integration more difficult. Many systems also lack integrated monitoring, automated alert correlation, and centralized logging mechanisms, increasing the workload on security analysts. The absence of prediction history storage and forensic logging further limits threat analysis, incident investigation, and long-term security monitoring capabilities in operational environments.

VII. RESEARCH GAPS IDENTIFIED

Based on the literature review and comparative analysis, several important research gaps are identified in existing intrusion detection systems. Many ML-based IDS frameworks remain limited to research prototypes and lack real-time deployment within practical web-based architectures. Most systems also do not provide integrated monitoring dashboards, centralized visualisation, persistent logging, or alert management features required for operational cybersecurity environments. In addition, only a few solutions combine anomaly detection for unknown threats with attack classification for known threats within a unified API-driven architecture. Scalability and modular extensibility are also insufficiently addressed in many existing approaches. The proposed RT-AI-IDS is designed to overcome these limitations through a scalable, real-time, and integrated cybersecurity monitoring framework.

VIII. PROPOSED SYSTEM

A. System Overview

The proposed Real-Time AI Intrusion Detection System is designed as a full-stack, API-driven cybersecurity monitoring platform that implements a hybrid machine learning detection pipeline within a production-grade web application architecture. The system accepts network traffic feature vectors as input, applies preprocessing and normalisation, performs sequential dual-model inference using Isolation Forest for anomaly detection and Random Forest for attack classification, and delivers structured results through RESTful API s consumed by a React-based monitoring dashboard while storing records in a SQLite database for retrospective analysis. The core design philosophy emphasises three principles: comprehensive detection through the integration of anomaly detection and supervised classification, operational usability through dashboard monitoring and automated alert generation, and architectural modularity through the separation of preprocessing, inference, API, and presentation layers into maintainable components. These principles address the research gaps identified in Section VII and distinguish the proposed system from existing research prototypes that primarily focus on detection accuracy without considering operational deployment requirements.

B. Design Rationale for Dual-Model Architecture

The dual-model architecture combining Isolation Forest and Random Forest is based on the insight that known and unknown threats require different detection strategies. Known attacks such as DoS, Probe, R2L, and U2R exhibit identifiable patterns that supervised models can learn from labelled data, while unknown or zero-day attacks often appear as statistical anomalies within network traffic. In the proposed system, Isolation Forest acts as the first-stage anomaly detector, identifying suspicious traffic, while Random Forest

performs second-stage attack classification. Anomalous traffic is classified into the most likely known attack category or flagged as an unclassified anomaly when confidence is low. This pipeline enables accurate detection of known attacks while maintaining coverage for novel threats without compromising classification precision.

IX. SYSTEM ARCHITECTURE

The proposed RT-AI-I DS follows a three-tier client-server architecture consisting of a React-based presentation layer, a FastAPI application layer, and a SQLite persistence layer, with the machine learning inference engine integrated into the application layer. This design ensures clear separation between user interface rendering, inference and business logic processing, and data storage, allowing each tier to be independently scaled, tested, and maintained without affecting the overall system.

Presentation Layer (React + Vite)

The presentation layer is implemented as a React single-page application built using the Vite toolchain, enabling fast development and optimised production builds. The dashboard provides real-time traffic monitoring, detection visualisations, alert management, and intrusion log browsing within a responsive interface suitable for security analyst workstations and network operations centres. React component-based architecture also allows new visualisation panels to be added without altering the core application structure.

Application Layer (FastAPI)

The application layer is implemented using FastAPI, a modern Python web framework that provides automatic OpenAPI documentation, asynchronous request handling, and type-validated API s. Its asynchronous execution model allows the system to process multiple concurrent prediction requests efficiently, which is essential for real-time network monitoring with high traffic volumes. The machine learning inference engine, consisting of pre-trained Isolation Forest and Random Forest models\ serialised with joblib, is loaded into memory during application startup and executed during prediction requests to minimise latency.

Persistence Layer (SQLite)

The persistence layer uses SQLite, a lightweight file-based relational database, to store prediction results, alert records, and intrusion logs. Its zero-configuration deployment model reduces operational overhead, making it suitable for both development and small-to-medium production environments. The database schema includes three primary tables: a predictions table for raw model outputs and timestamps, an alerts table for severity-classified detection records and acknowledgement status, and a logs table that maintains an audit trail of all detection events for forensic analysis and compliance reporting.

X. DETAILED SYSTEM DESIGN A. API Endpoint Design

The FastAPI application exposes a set of RESTful endpoints supporting the complete detection and monitoring workflow. The primary prediction endpoint accepts POST requests containing JSON-formatted network traffic feature vectors, validates inputs using Pydantic schemas, processes them through the preprocessing pipeline, executes the Isolation Forest and Random Forest models sequentially, and returns structured JSON responses with anomaly scores,predicted attack classes, confidence levels, and alert summaries. Additional endpoints provide access to historical predictions, alert management operations such as acknowledgement and filtering, and system health information for dashboard monitoring. The API follows RESTful principles throughout: GET requests retrieve records and system status, POST requests submit traffic data and create new records, PUT requests update alert status, and DELETE requests remove resolved alerts. All endpoints return consistent JSON response structures containing status codes, data payloads, and error messages, simplifying integration with the React frontend through standard fetch API calls.

B. Data Flow Architecture

The data flow through the proposed RT-AI-IDS proceeds through six interconnected stages: data collection, preprocessing, inference, API communication, alert generation, and persistence/dashboard visualisation. In the data collection stage, network traffic is captured from routers, switches, and host devices using packet sniffing techniques such as Scapy. During preprocessing, categorical features are label encoded, continuous features are normalised using the Min-Max scaler fitted during training, and missing values are handled through median imputation. In the inference stage, the processed feature vectors are analysed by the Isolation Forest model for anomaly detection, after which anomalous traffic is forwarded to the Random Forest classifier for attack classification. The API service layer manages real-time prediction requests and secure communication between system components. In the alerting stage, high-severity detections generate formatted alerts and notifications for administrators. Finally, all predictions, alerts, and logs are stored in the

SQLite database for dashboard visualisation, historical analysis, and forensic investigation.

XI. WORKFLOW OF THE PROPOSED SYSTEM

The operational workflow of the proposed system begins when network traffic feature data is submitted to the prediction API endpoint. The FastAPI handler deserializes the JSON payload and forwards it to the preprocessing module, where categorical features are label encoded and continuous features are normalised using the trained Min-Max scaler. The processed feature vector is then analysed by the Isolation Forest model to compute an anomaly score representing its deviation from normal traffic behaviour. If the anomaly score exceeds the configured threshold, the feature vector is passed to the Random Forest classifier, which predicts the most likely attack category using class probability estimates. The final detection output, including the anomaly score, predicted attack class, probability distribution, and severity level, is returned as a structured response to the client. At the same time, prediction records are stored in the SQLite database, and highseverity or non-normal detections generate alert entries that trigger notifications within the monitoring dashboard.

XII. DATA SET COLLECTION AND PREPROCESSING

A. NSL-KDD Dataset Overview

The NSL-KDD dataset, introduced by Tavallaee et al. as an improved version of the KDD Cup 1 999 dataset, is used as the primary training and evaluation dataset for the proposed system [2]. I t contains network connection records generated from simulated military network traffic, with each record represented by 41 features describing connection properties, payload characteristics, and traffic statistics. The dataset includes 1 25,973 training records (KDDTrain+) and 22,544 testing records (KDDTest+), with different training and testing distributions to better evaluate model generalisation. The dataset defines four attack categories alongside normal traffic. Denial of Service (DoS) attacks overwhelm system resources using highvolume traffic, while Probe attacks perform network reconnaissance such as port and vulnerability scanning. Remote-toLocal (R2L) attacks attempt to gain unauthorised local access from remote systems, whereas User-to-Root (U2R) attacks escalate privileges to administrator level through exploit techniques. This multi-class structure supports both binary intrusion detection and detailed attack classification.

B. Preprocessing Pipeline

The preprocessing pipeline for the NSL-KDD dataset performs three key transformations to prepare records for machine learning. First, categorical features such as protocol type, service, and flag are converted into numerical values using scikitlearn Label Encoder. These encoders are trained on the dataset and serialised with joblib to ensure consistent encoding during deployment. Second, all 41 features are normalised to the [0, 1] range using Min-Max scaling, which is essential for improving the performance of the Isolation Forest model by preventing large-scale features from dominating anomaly detection. Third, duplicate records are removed to reduce redundancy within the dataset and minimise model overfitting caused by repeated samples during training.

XIII. FEATURE ENGINEERING

The NSL-KDD dataset contains 41 pre-engineered features generated from raw packet captures using the Bro/Zeek network analysis framework. These features are grouped into four categories representing different aspects of network behaviour. Basic features describe connection properties such as duration, protocol type, service, connection status, and data transfer statistics. Content features analyse payload-related behaviour, including failed login attempts, root shell indicators, file creation activity, and outbound command counts, which are especially useful for detecting R2L and U2R attacks. Traffic features are divided into same-host and same-service categories. Same-host features analyse traffic patterns directed to the same destination host within a short time window, capturing metrics such as connection count and error rates that help identify DoS attacks. Same-service features compute similar statistics across connections to the same service port, enabling detection of distributed attacks and service-focused probe activity. Feature importance analysis during training identifies the most significant attributes for attack classification and supports potential dimensionality reduction for low-latency deployment environments.

XIV. MACHINE LEARNING METHODOLOGY

The machine learning methodology of the proposed system follows a structured train-validate-serialise workflow that separates model development from deployment. The pre-processed NSL-KDD dataset is divided into an 80/20 stratified train-validation split to preserve class distribution. The Isolation Forest and Random Forest models are trained on the training set, while hyperparameters are optimised using cross-validated grid search on the validation set. After training, both models, along with the fitted label encoders and Min-Max scaler, are serialised using joblib for deployment. Model

evaluation is performed on the NSL-KDD test set (KDDTest+), which differs from the training distribution to better assess real-world generalisation. Performance is measured using accuracy, precision, recall, F1 -score, and AUC-ROC for both binary and multi-class classification tasks. The false-positive rate is also evaluated to measure the operational impact of unnecessary alerts on security analysts.

XV. ISOLATION FOREST FOR ANOMALY DETECTION

The Isolation Forest algorithm, proposed by Liu et al., detects anomalies by constructing an ensemble of random isolation trees that recursively split data using randomly selected features and split values [4]. Data points that require fewer splits to isolate are assigned higher anomaly scores, as they are statistically rarer within the feature space. In the proposed system, the Isolation Forest model is trained only on normal traffic records from the NSL-KDD dataset, enabling it to learn the statistical behaviour of legitimate traffic without relying on labelled attack data. This approach improves the system' s ability to detect unknown or zero-day attacks. The contamination parameter is set conservatively to minimise the impact of noisy labels, while the number of trees is fixed at 1 00 to provide stable anomaly detection with acceptable computational overhead. The Isolation Forest stage acts as a coarse-grained filter that identifies suspicious traffic for further analysis by the Random Forest classifier, while allowing clearly normal traffic to pass without generating alerts. This reduces unnecessary classification processing and improves system throughput under high-traffic conditions.

XVI. RANDOM FOREST FOR ATTACK CLASSIFICATION

The Random Forest classifier used in the proposed system is an ensemble learning model based on multiple decision trees trained using Breiman' s bagging approach [3]. Each tree is trained on a randomly selected subset of training data and features, producing a set of decorrelated trees whose predictions are combined through majority voting to improve generalisation and reduce variance. The classifier is configured with 200 estimator trees, selected through cross-validated evaluation. The feature subset size at each split is set to the square root of the total feature count, following standard classification practice. To address the class imbalance in the NSL-KDD dataset, balanced class weights are applied so that minority attack categories such as R2L and U2R are not overlooked during training. An important advantage of the Random Forest model is its ability to estimate feature importance. Analysis shows that same-host and same-service traffic features, including destination host count and SYN error rate, are highly effective for detecting DoS attacks, while login failure counts and root shell indicators are particularly

useful for R2L and U2R detection. These insights improve model interpretability and support feature selection for low-latency deployment environments.

XVII. BACKEND API INTEGRATION

The FastAPI backend acts as the integration layer connecting the machine learning inference engine, SQLite database, and React frontend dashboard. I ts declarative routing, Pydantic-based validation, and automatic OpenAPI documentation reduce API development complexity while improving reliability and maintainability. The application follows a modular structure with separate components for routes, database operations, schemas, and model inference, enabling easier testing and future expansion. The prediction route executes the complete inference workflow within a single request cycle. Incoming requests are validated using Pydantic, pre-processed into model-ready formats, analysed sequentially by the Isolation Forest and Random Forest models, and returned as structured detection results while simultaneously being stored in the database. Error handling is implemented throughout the pipeline using appropriate HTTP status codes and descriptive responses to ensure reliable client communication. CORS middleware is configured to allow secure communication between the backend and React frontend across both development and production environments. API versioning is implemented using URL prefixes such asM /api/v1 /, providing a stable foundation for futureextensions without breaking existing integrations.

XVIII. FRONTEND DASHBOARD DESIGN

The React-based monitoring dashboard serves as the primary user interface of the proposed system, displaying detection results, alerts, traffic visualisations, and system status information in a unified real-time interface. Built using React 1 8 and the Vite toolchain, the dashboard leverages React' s concurrent rendering capabilities to support smooth updates under high data throughput. Its architecture follows a container-presenter pattern, where container components manage state and API communication while presenter components focus on data visualisation. The dashboard includes several key panels: a realtime traffic feed displaying incoming predictions, an alert summary panel grouping active alerts by severity and attack category, a historical trends panel visualising detection activity over time, a prediction history browser for accessing stored logs, and a system health panel showing API latency, model status, and database activity. Data updates are handled through periodic API polling with

configurable refresh intervals to balance responsiveness and system load.

XIX. DATABASE AND LOGGING ARCHITECTURE

The SQLite database in the proposed system is designed to support three key functions: real-time alert monitoring, historical analysis of detection events, and audit trail maintenance for forensic investigation and compliance purposes. The schema is organised with relational links between prediction, alert, and log records, enabling efficient retrieval of complete detection histories. The predictions table stores all inference results, including feature vector hashes, anomaly scores, predicted attack classes, probability values, severity levels, and timestamps. The alerts table manages operational alerts with fields for alert status, linked prediction identifiers, alert descriptions, and analyst investigation notes. The logs table maintains a chronological and immutable record of all detection events, supporting forensic reconstruction and long-term security analysis.

XX. REAL-TIME MONITORING AND ALERT MECHANISM

The alert generation mechanism of the proposed system is designed to provide timely and actionable notifications while reducing false-positive alert fatigue. Alert severity is calculated using a weighted combination of the Isolation Forest anomaly score and the Random Forest prediction confidence, categorised into four levels: informational, low, medium, and high. By default, only medium- and high-severity alerts are displayed in the main dashboard, while lower-severity events remain accessible through the prediction history interface. To prevent alert storms during sustained attacks, the system implements alert deduplication by grouping related events based on attack type, source and destination address hash, and a configurable time window. Multiple similar events are aggregated into a single alert record with an associated event count, reducing the operational burden on security analysts while preserving complete detection details in the database logs for forensic analysis.

XXI. SECURITY AND PRIVACY CONSIDERATIONS

The deployment of the proposed system in production environments introduces important security and privacy considerations. From a security perspective, the FastAPI backend should be deployed behind a reverse proxy such as Nginx with TLS encryption enabled for all API communications, protecting detection results and traffic feature data from interception. API authentication using bearer tokens or API keys should also be enforced for all operational endpoints to prevent unauthorised access to alerts and monitoring records. From a privacy perspective, the system analyses network connection metadata such as packet counts, timing statistics, and protocol information rather than full packet payloads, reducing exposure to personally identifiable information (PI I). However, source and destination IP addresses stored in alerts and logs may still be treated as personal data in some jurisdictions, making database access controls and data retention policies necessary for regulatory compliance.

XXII. EXPERIMENTAL SETUP

The experimental evaluation of the proposed RT-AI-IDS was conducted using the NSL-KDD benchmark dataset in a controlled environment. Model training was performed on the KDDTrain+ dataset containing 1 25,973 records across five classes: normal, DoS, Probe, R2L, and U2R. A stratified 80/20 train-validation split was used for hyperparameter tuning, while final evaluation was carried out on the KDDTest+ dataset containing 22,544 records. The differing training and testing distributions were intended to simulate real-world generalisation conditions. The experiments were conducted using Python 3.1 0 with Scikit-learn 1 .3, NumPy 1 .24, and Pandas 2.0. The Isolation Forest model was configured with n_estimators = 1 00 and contamination = 0.1 , while the Random Forest classifier used n_estimators = 200, max_features = 'sqrt', and class_weight = 'balanced'. The FastAPI backend was evaluated for response latency using the Locust load-testing framework with up to 1 00 concurrent prediction requests, and the React frontend performance was analysed using Chrome DevTools under simulated real-time traffic conditions.

TABLE II: Experimental Configuration Summary

Component	Configuration
Isolation Forest	Number of Trees
Isolation Forest	Contamination Rate
Random Forest	Number of Estimators
Random Forest	Max Features
Random Forest	Class Weight
Training Set	Records
Test Set	Records
API Framework	Version
Database	Type
Frontend	Framework

XXIII. PERFORMANCE EVALUATION

Performance evaluation of the proposed system was conducted across three dimensions: classification performance on the NSL-KDD test set, anomaly detection performance using precision-recall analysis at different Isolation Forest thresholds, and operational performance measured through API latency and dashboard throughput benchmarking. This evaluation framework reflects the requirements of a real-time intrusion detection system, which must maintain high detection accuracy, low false-positive rates, and efficient processing performance. Classification performance was analysed using a confusion matrix generated from Random Forest predictions on the KDDTest+ dataset, from which precision, recall, and F1 -score were calculated for each attack category. The area under the ROC curve (AUC-ROC) was also computed for binary normalversus-attack classification to evaluate the trade-off between true-positive and false-positive rates across different probability thresholds. In addition, detection rate and false alarm rate were measured as key operational metrics affecting the workload of security analysts.

TABLE III: Classification Performance Results (Random Forest on NSL-KDD)

Class	Precision	Recall	F1-Score	Support
Normal	0.982	0.979	0.981	9,711
DoS	0.994	0.997	0.995	7,458
Probe	0.961	0.958	0.960	2,421
R2L	0.873	0.847	0.860	2,754
U2R	0.791	0.762	0.776	200
Overall (Weighted)	0.974	0.974	0.974	22,544

XXIV. EXPERIMENTAL RESULTS AND ANALYSIS

The experimental results demonstrate that the proposed dual-model architecture achieves strong detection performance across all attack categories in the NSL-KDD benchmark. The Random Forest classifier achieves a weighted F1 -score of 0.974 across the five classes, with particularly high accuracy for DoS and normal traffic detection due to the availability of richer training samples for these categories. Lower performance on R2L and U2R attacks is consistent with existing research, mainly because of limited training data and similarities between certain attack behaviours and legitimate remote access activity. The Isolation Forest anomaly detector achieves an overall attack detection rate of 0.923 with a false-positive rate of 0.031 on normal traffic. When combined with the Random Forest classifier, the overall system false-positive rate decreases to 0. 021 , demonstrating the effectiveness of the dual-model architecture. The Isolation Forest stage filters suspicious traffic before classification, reducing unnecessary processing and improving classification precision. Performance benchmarking also shows that the FastAPI backend achieves a median response latency of 8. 3 milliseconds under single-client conditions and 23. 7 milliseconds under 1 00 concurrent clients, remaining within real-time intrusion detection requirements. In addition, the React dashboard maintained smooth real-time visualisation performance at 60 frames per second while processing simulated streams of 50 prediction updates per second.

XXV. ADVANTAGES OF THE PROPOSED SYSTEM

The proposed RT-AI-I DS offers several advantages over existing intrusion detection approaches. Its hybrid dual-model architecture provides detection coverage for both known and unknown attacks, overcoming the limitations of purely signature-based systems while reducing the high false-positive rates commonly associated with standalone anomaly detection methods. The integrated full-stack design — including preprocessing, inference, API services, database persistence, and dashboard monitoring — also simplifies deployment in operational environments compared to researchfocused prototypes that lack deployment infrastructure. The use of efficient tree-based algorithms such as Isolation Forest and Random Forest enables real-time inference on standard server hardware without the need for GPU acceleration. Model serialisation using joblib supports fast and reliable model loading during deployment, reducing startup overhead. In addition, the RESTful API architecture enables seamless integration with existing network monitoring tools, SI EM platforms, and incident response systems through standard HTTP-based communication.

XXVI. SCALABILITY AND DEPLOYMENT DISCUSSION

The proposed system is designed with horizontal scalability as a key architectural feature. The FastAPI backend is stateless, with all persistent data stored in the SQLite database, allowing multiple backend instances to run behind a load balancer efficiently. Machine learning models are loaded as shared read-only in-memory objects to support concurrent prediction requests with minimal overhead. For larger deployments, SQLite can be replaced with PostgreSQL or MySQL through SQLAlchemy without major code changes. In high-throughput environments, database writes can also be handled asynchronously using Redis or RabbitMQ to improve response performance. Docker containerisation provides a portable and reproducible deployment environment containing

FastAPI, scikit-learn, and database dependencies. Docker Compose supports multi-container deployments with services such as the FastAPI backend, Nginx reverse proxy, and monitoring tools. The React frontend can be deployed independently on CDN or cloud storage platforms, simplifying scalability and maintenance.

XXVII. FUTURE SCOPE

Several directions for future development could further enhance the capability and applicability of the proposed system. In terms of detection methodology, integrating deep learning models such as graph neural networks and LSTM architectures could improve the detection of complex multi-stage attacks and advanced persistent threats (APTs). The addition of online learning mechanisms would also allow the system to adapt to evolving traffic patterns without requiring complete retraining. From an operational perspective, future work could include native integration with popular SIEM platforms such as Splunk, Elastic SI EM, and Microsoft Sentinel, enabling the system to function as an AI-powered enhancement layer within existing security infrastructures. Webhook-based integrations with platforms like PagerDuty and JI RA could support automated incident response workflows, while MITRE ATT&CK mapping would align detection outputs with industry-standard threat intelligence frameworks. Future evaluation should also extend beyond the NSL-KDD dataset to more recent datasets such as CICIDS-201 7, UNSW-NB1 5, and CI C-DDoS201 9, which better represent modern network traffic and attack patterns. Additionally, transfer learning experiments on real-world network traffic would help evaluate the practical gap between benchmark performance and operational deployment effectiveness.

XXVIII. CONCLUSION

This paper presented the design, implementation, and evaluation of a Real-Time AI Intrusion Detection System using a hybrid machine learning architecture that combines Isolation Forest anomaly detection with Random Forest attack classification. The proposed system addresses several research gaps identified in existing literature, including the lack of production-ready hybrid detection architectures, integrated monitoring infrastructure, persistent logging and audit capabilities, unified anomaly and classification outputs, and scalability considerations in research prototypes. Experimental evaluation on the NSL-KDD dataset achieved a weighted F1 - score of 0.974 for multi-class attack classification, a falsepositive rate of 0.021 , and low API response latencies of 8.3 milliseconds under single-client load and 23. 7 milliseconds under 1 00 concurrent clients. These results demonstrate that the proposed system provides both strong detection performance and real-time operational capability suitable for practical deployment. The complete full-stack architecture — consisting of a FastAPI backend, React monitoring dashboard, SQLite persistence layer, and joblib-serialised inference pipeline — represents a deployable operational solution rather than only a research prototype. Its modular design also supports future enhancements such as deep learning integration, SIEM connectivity, and evaluation on modern real-world traffic datasets, helping bridge the gap between academic intrusion detection research and operational cybersecurity deployment.

REFERENCES

[1] Scikit-learn D
[2] ocumentation, Machine Learning in Python. Available: https://scikit-learn.org/
[3] FastAPI Documentation, FastAPI Framework. Available: https://fastapi. tiangolo.com/
[4] React Documentation, React JavaScript Library. Available: https://react.dev/
[5] Vite Documentation, Next Generation Frontend Tooling. Available: https://vitejs.dev/
[6] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A Detailed Analysis of the KDD CUP 99 Dataset,” in IEEE Symposium on Computational Intelligence for Security and Defence Applications, 2009.
[7] NSL-KDD Dataset, Network Intrusion Detection Dataset. Available: https://www. unb. ca/cic/datasets/nsl.html
[8] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in IEEE International Conference on Data Mining, 2008.
[9] A. Liaw and M. Wiener, “Classification and Regression by Random Forest,” , 2002.
[10] Pandas Documentation, Python Data Analysis Library. Available: https://pandas.pydata.org/
[11] NumPy Documentation, Scientific Computing with Python. Available: https://numpy.org/
[12] Python Software Foundation, Python 3 Documentation. Available: https://docs.python.org/3/ OWASP Foundation, Web Application Security Testing Guide. Available: https://owasp.org/