International Journal of Linguistics and computational Applications (IJLCA)
Volume 13, Issue 2, April – June 2026
ISSN 2394-6385(Print)
ISSN 2394-6393(Online)

An AI-Driven Cybersecurity Threat Detection and Incident Response Platform for Real-time Network Protection

Kamaluri Mubarak[*1], V. Vijayalakshmi[2], Dr. S. Usharani[3]

[1]*Student, Department of Computer Applications, Viswam Engineering College, Andhra Pradesh, India*
[2]*Associate Professor, Department of Computer Applications, Viswam Engineering College, Andhra Pradesh, India*
[3]*Professor & Head, Department of MCA, Viswam Engineering College, Andhra Pradesh, India*
Email: vijji.varapana@gmail.com, anjanasundar80@gmail.com

Abstract— The rapid expansion of digital infrastructure across industries has significantly increased the vulnerability of systems to cyber threats such as Denial-of-Service attacks, unauthorized access, probing activities, and privilege escalation attempts. Traditional security mechanisms, which rely on signature-based detection, are increasingly ineffective against evolving and unknown attack patterns. This paper presents a Smart Cybersecurity Threat Detection Platform that leverages machine learning techniques for real-time identification and classification of network threats. The proposed system is developed using the Django web framework and Python, integrating a Random Forest classifier trained on a dataset modelled after the NSL-KDD benchmark. The platform processes network traffic parameters such as protocol type, connection duration, source and destination byte counts, connection flags, and connection frequency to classify activities into Normal, DoS, Probe, R2L, and U2R categories. In addition to detection, the system incorporates an Automated Threat Isolation System (ATIS) that identifies high-confidence threats and initiates isolation workflows. A real-time dashboard provides visualization of threat distribution, system performance metrics, and alert management. The system also maintains an audit trail of all activities and supports alert tracking and resolution workflows. The Smart Cybersecurity Threat Detection Platform offers a scalable and intelligent approach to network security by combining anomaly detection, real-time monitoring, and automated response mechanisms. It demonstrates an effective transition from reactive to proactive cybersecurity defense strategies.

Keywords— *Cybersecurity; Machine Learning; Intrusion Detection; Random Forest; Django; Threat Detection; Network Security.*

1. Introduction

In today's highly interconnected digital world, organizations depend heavily on networked systems for communication, data exchange, and operational efficiency. However, this interconnectedness also exposes systems to a wide range of cyber threats, including Distributed Denial-of-Service (DoS) attacks, unauthorized access attempts, network probing, and advanced persistent threats. Traditional cybersecurity solutions such as firewalls and signature-based intrusion detection systems operate on predefined rules and known attack patterns. While effective against known threats, these systems fail to detect novel or evolving attacks, often referred to as zero-day attacks. Additionally, the massive volume of network traffic generated in modern environments makes manual monitoring impractical and inefficient.

Machine learning has emerged as a powerful tool in cybersecurity, enabling systems to automatically learn patterns from historical data and detect anomalies without explicit rule definitions. By analyzing network traffic characteristics, machine learning models can classify activities as normal or malicious with high accuracy. The Smart Cybersecurity Threat Detection Platform is designed to address these challenges by integrating machine learning-based threat detection with real-time monitoring and automated response mechanisms. The system uses a Random Forest classifier to analyze network activity and identify potential threats in real time. Furthermore, the platform introduces an Automated Threat Isolation System (ATIS), which enhances security by initiating isolation procedures for high-confidence threats. The system also provides a comprehensive dashboard for monitoring alerts, analyzing trends, and evaluating model performance. This project demonstrates how modern web technologies and machine learning techniques can be combined to build an intelligent, scalable, and efficient cybersecurity solution.

2. Literature Survey

Cybersecurity threat detection has evolved from static rule-based mechanisms to advanced intelligent systems that use statistical learning and artificial intelligence. Traditional Intrusion Detection Systems (IDS) such as signature-based tools are effective only against previously identified threats. These systems compare incoming traffic against known attack patterns and trigger alerts when matches are found. While precise for known threats, they are unable to detect zero-day exploits or modified attack strategies.

Anomaly-based intrusion detection systems were introduced to overcome this limitation by identifying deviations from normal network behavior. These systems

DOI: 10.30726/ijlca/v13.i2.2026.132018

establish a baseline of expected activity and flag anomalies that differ significantly. However, pure anomaly-based systems often suffer from high false positive rates because legitimate but unusual activity may also be classified as malicious. Security Information and Event Management (SIEM) platforms extended threat detection by aggregating logs from multiple sources and correlating events. Tools such as Splunk and IBM QRadar provide centralized monitoring, search, and reporting capabilities.

Although powerful, they require extensive configuration, continuous rule updates, and skilled analysts to remain effective. With the rise of machine learning, cybersecurity systems began leveraging classification algorithms such as Decision Trees, Support Vector Machines, Neural Networks, and Random Forest classifiers. These approaches use historical labeled datasets to learn the statistical characteristics of benign and malicious traffic. Random Forest has become especially popular because of its ability to handle heterogeneous features, reduce overfitting, and provide feature importance scores that improve model interpretability.

Modern research also explores automated threat response systems that not only detect attacks but also initiate containment measures. Automated isolation, blocking, and alert escalation mechanisms reduce incident response time and improve operational efficiency. The proposed Smart Cybersecurity Threat Detection Platform builds upon these advancements by integrating machine learning, real-time logging, alert management, and automated isolation into a unified web-based system.

3. System Proposal

3.1 Existing System

Existing cyber security threat detection approaches primarily fall into four categories:

- *Signature-based Intrusion Detection Systems* - These systems detect only known threats using predefined signatures. They are accurate for familiar attacks but ineffective against unknown or evolving threats.
- *Anomaly-based Detection Systems* - These systems identify deviations from normal traffic behavior. While useful for detecting unknown patterns, they often generate many false alarms.
- *SIEM Platforms* - SIEM systems collect and correlate logs from multiple devices and applications. They provide centralized visibility but require continuous tuning and skilled management.
- *Commercial Machine Learning Security Platforms* - Advanced commercial tools apply machine learning to detect threats but are often expensive, vendor-dependent, and less transparent to users.

3.2 Proposed System

The Smart Cybersecurity Threat Detection Platform proposes an integrated solution that combines intelligent threat classification with operational response capabilities. The system uses a Random Forest classifier trained on traffic data modeled after the NSL-KDD benchmark to classify activities into:

- Normal
- DoS
- Probe
- R2L
- U2R

The platform is implemented using Django for web application management and scikit-learn for machine learning. The system includes the following major capabilities:

- *Threat Alert Generation* -Suspicious activity results in the creation of structured alerts with severity levels.
- *Automated Threat Isolation System (ATIS)* - High-confidence threats trigger an automatic isolation workflow.
- *Interactive Security Dashboard* - The dashboard provides logs, recent threats, charts, and model performance insights.
- *Persistent Audit Trail* - All analyzed activities and system actions are stored in the database for review and reporting.

This proposed system moves beyond passive detection by combining analysis, visualization, and containment within a single framework.

4. System Architecture

4.1 Overall System Architecture

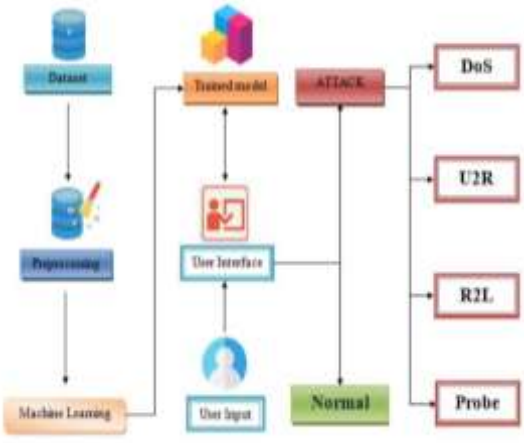

Fig.1: Overall System Architecture

DOI: 10.30726/ijlca/v13.i2.2026.132018

The Smart Cyber security Threat Detection Platform follows a three-tier layered architecture that separates presentation, application logic, and data persistence. This architectural design improves maintainability, scalability, and modular development. The uploaded project content describes the system as a layered web application built on Django with a machine learning engine, database layer, and dashboard-based presentation interface. The architecture is divided into the following tiers:

4.1.1 Presentation Tier

The Presentation Tier consists of the browser-based user interface developed using Django templates, Bootstrap 5, and Chart.js. This layer allows security analysts and administrators to:

- View the dashboard
- Monitor activity logs
- Review threat alerts
- Analyze model metrics
- Trigger simulation and response actions

The visual dashboard provides charts, tables, and alert cards, making security information easy to understand and act upon.

4.1.2 Application Tier

The Application Tier is the processing core of the platform. It is implemented using Django and contains:

- URL routing
- View/controller logic
- Authentication and authorization
- Machine learning inference logic
- Alert generation workflow
- Automated Threat Isolation System (ATIS)

This layer receives requests from the interface, processes network activity data, interacts with the machine learning model, and updates the database with predictions, alerts, and isolation records.

4.1.3 Data Tier

The Data Tier includes the SQLite database and model artifact storage. In addition to relational data storage, the trained Random Forest model and label encoders are stored as serialized files (rf_model.pkl and encoders.pkl) and are loaded when required for prediction. This tier ensures persistence, auditability, and performance tracking.

4.2 System Flow Diagram

The operational workflow of the Smart Cybersecurity Threat Detection Platform consists of two main processes:

Fig.2: System Flow Diagram

- Model Training Workflow
- Real-Time Threat Detection Workflow

4.2.1 Model Training Workflow

The training process begins when the administrator runs the model training command. The system generates a synthetic dataset based on NSL-KDD-like statistical properties, preprocesses the data, trains a Random Forest classifier, evaluates model performance, and saves the trained model and encoders to disk. This process ensures that the detection engine is prepared before live prediction begins.

4.2.2 Real-Time Detection Workflow

The real-time workflow starts when a new network connection event is received, either through simulation or future external integration. The event parameters are passed to the prediction engine, if the prediction is Normal, the activity is simply stored in the database. If it is a threat, the system creates a ThreatAlert. If the threat has high confidence, the Automated Threat Isolation System records an IsolatedEntity entry for follow-up and containment. The dashboard is then updated for analyst review.

- Loads the trained model if needed
- Preprocesses the input
- Predicts the traffic category
- Computes a confidence score

4.3 UML Diagrams

4.3.1 Use Case Diagram

The Use Case Diagram of the Smart Cybersecurity Threat Detection Platform identifies three primary actors: the Security Analyst, the Security Administrator, and the Machine Learning Engineer. The Security Analyst interacts with the operational dashboard and performs activities such as viewing the dashboard, browsing activity logs, reviewing threat alerts, resolving alerts, and simulating traffic for testing. The Security Administrator performs all analyst-

DOI: 10.30726/ijlca/v13.i2.2026.132018

level operations and additionally manages user access, reviews isolation actions, and lifts isolated entities when required. The Machine Learning Engineer is responsible for training and evaluating the machine learning model through the management command workflow. This use case structure shows that the platform supports both day-to-day security monitoring and model lifecycle administration within a unified system.

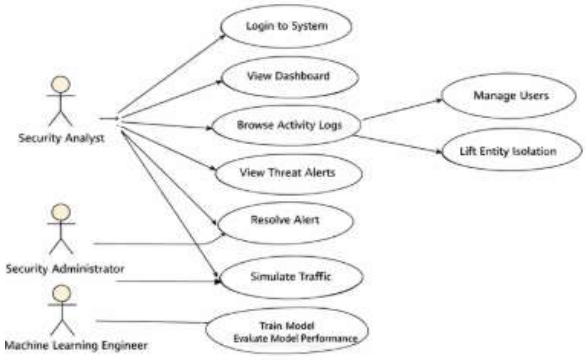

Fig.3: Use Case Digram of Smart Cyber Security Threat Detection Platform

4.3.2 Class Diagram

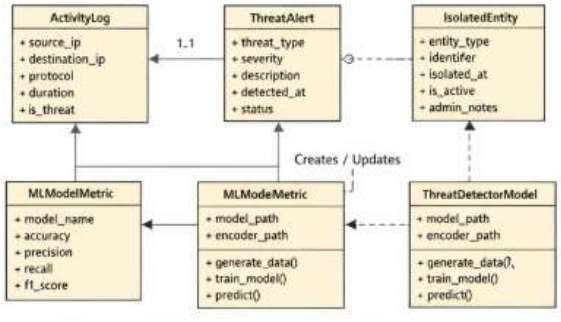

Fig.4: Class Diagram of Smart Cyber Security Threat Detection Platform

The Class Diagram reflects the structural design of the platform across its major domain entities and machine learning component. The ActivityLog class stores network connection parameters and prediction results, including source IP, destination IP, protocol, byte counts, confidence score, and threat status. The ThreatAlert class maintains a one-to-one relationship with ActivityLog and stores alert-specific details such as threat type, severity, description, and status. The IsolatedEntity class records all isolated network identifiers and their administrative state. The MLModelMetric class stores model performance measurements such as accuracy, precision, recall, F1-score, and feature importance. The ThreatDetectorModel class handles synthetic data generation, preprocessing, model training, model loading, and prediction. Together, these classes form the logical and operational backbone of the cybersecurity platform.

Key Attributes:

- ActivityLog → source_ip, destination_ip, protocol, duration, src_bytes, dst_bytes, flag, count, prediction, confidence, is_threat
- ThreatAlert → threat_type, severity, description, detected_at, status
- IsolatedEntity → entity_type, identifier, reason, isolated_at, is_active, admin_notes
- MLModelMetric → model_name, accuracy, precision, recall, f1_score, trained_at, feature_importance
- ThreatDetectorModel → model_path, encoder_path, model, encoders, feature_cols

Relationships:

- One ActivityLog → One ThreatAlert
- ThreatAlert depends on ActivityLog
- ThreatDetectorModel creates predictions used by ActivityLog
- High-severity alerts may create or update IsolatedEntity
- Model training results are stored in MLModelMetric

5. Implementation

5.1 Modules

The Smart Cyber security Threat Detection Platform is designed using a modular architecture, where each module is responsible for a specific functionality within the system. This modular approach improves maintainability, scalability, and ease of development. According to the project structure, the system consists of the following major modules:

- Machine Learning Engine Module
- Database Model Module
- View and Controller Module
- URL Routing Module
- Configuration Module
- Template and Presentation Module
- Management Command Module

Each module interacts with others to provide a complete threat detection and response system.

5.2 Module Description

The system consists of several integrated modules working together to ensure efficient operation. The Machine Learning Engine Module, implemented in *ml_engine.py*, handles data generation, preprocessing, model training, and prediction using a Random Forest classifier with encoded inputs and stored models for reuse, enabling fast and accurate classification of network activities. The Database Model Module, defined in *models.py* using Django ORM,

DOI: 10.30726/ijlca/v13.i2.2026.132018

manages entities like activity logs, threat alerts, isolated IPs, and model metrics, ensuring data consistency and traceability. The View and Controller Module in *views.py* processes user requests, connects with the ML model, and manages dashboards, alerts, logs, and simulations, acting as the core logic layer.

The URL Routing Module in *urls.py* maps requests to appropriate views, ensuring smooth navigation across features like dashboard, alerts, and authentication. The Configuration Module in *settings.py* controls system settings including database, middleware, and security, ensuring coordinated functioning. The Template Module uses HTML, Bootstrap, and Chart.js to provide an interactive UI with dashboards, logs, and visualizations. Finally, the Management Command Module enables model training and traffic simulation through command-line tools for efficient administration.

6. Testing of the System

Testing is a crucial phase in the development of the Smart Cybersecurity Threat Detection Platform to ensure that all system components function accurately, efficiently, and securely. Since the system integrates machine learning, web application logic, and database operations, a comprehensive testing strategy is adopted. The testing process includes unit testing, integration testing, functional testing, and performance testing to validate both system correctness and real-time responsiveness.

6.1 Unit Testing

Unit testing focuses on validating individual modules and components of the system independently. Each module is tested using predefined inputs and expected outputs to ensure correctness. The Machine Learning Engine Module is tested by providing sample input data and verifying whether the model produces correct predictions along with confidence scores. The preprocessing functions are tested to ensure categorical encoding and feature alignment are accurate.The Database Model Module is tested by verifying CRUD operations on models such as ActivityLog, ThreatAlert, IsolatedEntity, and MLModelMetric. Relationships between these models are also validated to ensure data integrity.The View Module is tested to confirm that requests are processed correctly and responses are returned with the expected structure. This includes testing dashboard rendering, alert generation, and activity logging. Unit testing ensures that each system component behaves as expected in isolation.

6.2 Integration Testing

Integration testing verifies that different modules of the system interact seamlessly. It ensures that the complete workflow—from receiving input to generating alerts—is functioning correctly. Key integration scenarios include:

- Passing simulated network data to the machine learning engine
- Storing prediction results in the ActivityLog
- Generating ThreatAlert entries for malicious activity
- Triggering isolation workflows for high-confidence threats
- Updating dashboard data dynamically

The Django test client is used to simulate HTTP requests and validate end-to-end functionality. Integration testing ensures that data flows correctly across modules and that the system operates as a unified platform.

6.3 Functional Testing

Functional testing evaluates whether the system meets all specified requirements and performs expected operations. The system is tested for:

- Correct classification of network activity
- Accurate alert generation for threats
- Proper storage of logs and metrics
- Functionality of isolation mechanisms
- Dashboard updates and visualization

Various test cases are executed using both normal and malicious input data to ensure reliable classification and response.

6.4 Performance Testing

Performance testing assesses the responsiveness and scalability of the system under different workloads. The response time for prediction and alert generation is measured to ensure real-time capability. The system is tested with multiple simulated traffic inputs to evaluate its ability to handle concurrent requests. The machine learning model is evaluated for prediction speed and efficiency. The Random Forest model demonstrates fast inference time, making it suitable for real-time threat detection.

7. Conclusion and Future Enhancement

7.1 Conclusion

The Smart Cybersecurity Threat Detection Platform successfully demonstrates the application of machine learning techniques in identifying and mitigating network-based cyber threats. By integrating a Random Forest classifier with a Django-based web application, the system provides an intelligent, real-time threat detection mechanism capable of classifying network activities into multiple categories such as Normal, DoS, Probe, R2L, and U2R.The platform not only detects threats but also enhances cybersecurity operations through automated alert generation

DOI: 10.30726/ijlca/v13.i2.2026.132018

and the implementation of the Automated Threat Isolation System (ATIS). This enables faster response to high-confidence threats, reducing potential damage and improving system resilience. The inclusion of a centralized dashboard further strengthens the system by offering clear visualization of alerts, logs, and performance metrics.The modular architecture, use of open-source technologies, and lightweight database make the system scalable, cost-effective, and suitable for deployment in academic as well as small to medium enterprise environments. Overall, the project demonstrates a shift from traditional reactive security mechanisms to a proactive, intelligent cybersecurity solution.

7.2 Future Enhancement

Although the current system provides a strong foundation, several enhancements can further improve its effectiveness and scalability. One major enhancement is the integration of real-time network traffic capture using packet analysis tools such as Wireshark or network sniffing libraries. This would eliminate reliance on simulated data and enable deployment in live environments. Another improvement is the adoption of deep learning models such as LSTM or CNN for more advanced pattern recognition in network traffic. These models can capture temporal dependencies and improve detection accuracy for complex attack patterns. The system can also be extended by implementing role-based access control, allowing different levels of permissions for administrators, analysts, and auditors. This will enhance system security and usability in multi-user environments. Integration with alert notification systems such as email, SMS, or messaging platforms can ensure that critical threats are immediately communicated to security teams. Additionally, migrating from SQLite to enterprise-grade databases such as PostgreSQL or MySQL will improve scalability and support higher volumes of data. Finally, deploying the system in a cloud environment with microservices architecture can improve performance, scalability, and availability, making it suitable for large-scale cybersecurity operations.

References

[1] W.Stallings, *Network Security Essentials: Applications and Standards*, 6th Edition, Pearson, 2017.
[2] C. Kruegel and G. Vigna, "Anomaly Detection of Web-based Attacks," *Proceedings of the ACM Conference on Computer and Communications Security*, 2003.
[3] M. Tavallaee et al., "A Detailed Analysis of the NSL-KDD Dataset," *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009.
[4] L. Breiman, "Random Forests," *Machine Learning*, Vol. 45, No. 1, 2001, pp. 5–32.
[5] Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," 2024.
[6] Django Software Foundation, "Django Documentation," Version 5.x, 2025.
[7] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016.
[8] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," *IEEE Symposium on Security and Privacy*, 2010.
[9] KK Rakesh, AS Aneeshkumar, "Self-directed Moving Strategy for Cluster Leaders to Maximize the Lifespan of Sensor Network", Semiconductor Optoelectronics, Vol. 42, Issue 2, 2023, pp. 1594-1610.
[10] NumPy Developers, "NumPy Documentation," 2024.
[11] J. Brownlee, "Machine Learning Mastery for Time Series and Security Applications," 2018.
[12] OWASP Foundation, "OWASP Top 10 Security Risks," 2023.
[13] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," NIST, 2007.
[14] Joblib Developers, "Joblib Documentation," 2024.
[15] Bootstrap Team, "Bootstrap Documentation," 2024.
[16] Pandas Development Team, "Pandas Documentation," 2024.

DOI: 10.30726/ijlca/v13.i2.2026.132018