Unified Framework for Cyber Threat Prediction, Real-Time Monitoring, and Incident Response Using AI

Venkata Naveen Kumar Prabhuleti — *College of Information Science and Technology* — *Pennsylvania State University,University Park*

Abstract—Modern cybersecurity requirements include immediate threat detection together with real-time visibility and automated incident response capabilities. This paper introduces an AI-based framework which combines threat prediction capabilities with real-time monitoring functions and automated response orchestration. The system uses network traffic data-trained machine learning models to achieve accurate threat classification while providing an interactive dashboard for visualization. The system provides real-time event tracking and confidence scoring and incorporates external threat intelligence feeds from AlienVault OTX to enhance situational awareness. The system activates automated suggestions which include IP blocking and anomaly flagging and custom YARA rule generation when critical threats are detected. The unified approach shortens detection-to-response times while enhancing SOC workflows and strengthening organizational cybersecurity resilience. The framework provides a scalable base for upcoming developments that include deep learning extensions, large language models and real-world SOC integrations.

Index Terms — Automated Incident Response, Machine Learning, Real-Time Monitoring, Network Traffic Analysis, Security Operations Center (SOC), AI-driven Security Framework, Threat Intelligence Integration

I. Introduction

Modern organizations encounter increasing cybersecurity threats because their incident detection and response times lag within extensive SIEM log data which results in operational disruptions along with financial losses and reputational damage. The process of manual log analysis demands extensive time while introducing numerous errors and necessitating continual SOC analyst involvement which leads to response delays and increased risk exposure. The necessity of resilient cybersecurity demands an autonomous system that detects vulnerabilities while analyzing threats and performing real-time incident response. Organizations' networks experience a merger of high network traffic with APT threats which infiltrate their systems. The primary network threat in 2024 emerges as DDoS attacks which represent 8% of all categorized attack types [1][2].

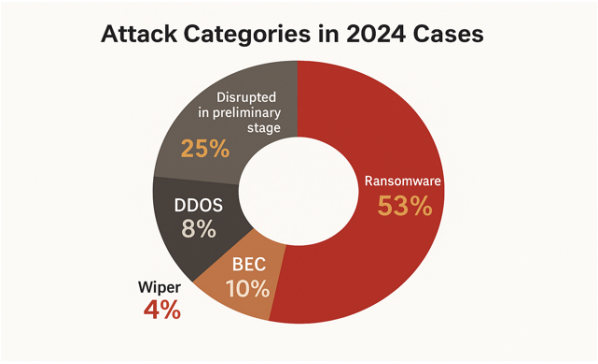

Figure 1. 8% of DDoS from 2024 attack categories

A. Problem Motivation

Security teams face an unmanageable challenge in real-time threat detection and incident response due to the overwhelming volumes of SIEM and network logs they must handle. The process of manual analysis demands extensive labor while simultaneously introducing human errors which result in both missed threats and delayed containment. Organizations without intelligent automation proposals face increased susceptibility to rapid attacks which take advantage of delayed response times, leading to heightened risks of operational downtime and data breaches.

B. Objective

This project aims to design and develop an AI-driven, automated incident response system capable of:

1) Classifying security events into severity levels based on predicted threat types,
2) Automatically provides tailored response actions such as blocking malicious IPs, raising alerts, and generating YARA rules for future detections,
3) Providing a real-time monitoring dashboard for visibility into active incidents, system responses, and evolving threat patterns.

C. Challenges in Current SIEM and SOC Operations

Organizations face persistent challenges with alert fatigue and delayed triage along with inefficient manual investigations even after significant investments in SIEM platforms. The necessity for analysts to examine countless events to pinpoint critical incidents results in burnout and variable threat management [3]. Static rule-based systems prove ineffective against new attack vectors because they lack the necessary adaptability which modern threat environments require for responsive speed and intelligent countermeasures.

D. Importance of Automated, Intelligent Incident Response

Modern cyber threats exceed the incident response abilities of human operators through traditional incident response methods. Automated systems controlled by AI reduce the time needed to detect threats while making steady decisions and implementing instant solutions that prevent lengthy operational delays. When organizations use intelligent automation for triage and automatic response functions together with automatic rule evolution, they become able to extend their defensive capabilities while

allowing their human analysts to perform strategic threat hunting and strategic enhancements. Cybersecurity operations must have intelligent automation since it represents a fundamental requirement to maintain effective resilient cybersecurity operations [4].

II. Background

A. Related Work

Recent studies have applied machine learning to cybersecurity for threat detection and classification, with datasets like *CICIDS2017* and *CICDDOS2019* becoming industry benchmarks. While *SOAR* platforms automate parts of incident response, most rely on static playbooks and lack real-time adaptability. Prior research primarily emphasizes detection accuracy but seldom integrates live threat monitoring, automated response, and threat prediction into a single unified system. Furthermore, many existing solutions are proprietary and inaccessible for wider academic or organizational adoption [5].

B. Research Gap

The current threat detection and response automation progress leaves an open space since these processes operate independently in most security systems. The existing SIEM and SOAR tools demand substantial human adjustments to function properly and they deliver delayed responses because they lack adaptive capabilities for new threats [6]. Current research demonstrates low levels of development regarding end-to-end frameworks that automatically execute real-time incident prediction, monitoring, and mitigation processes together with transparent analysis functionality for security analysts. A solution to this problem requires an AI-driven integrated system that should synchronize these crucial cybersecurity operations in a seamless manner.

III. Methodology and System Design

The proposed framework integrates machine learning, real-time monitoring, and automated incident response into a unified, production-grade pipeline. The system was designed to simulate real-world SOC operations by ingesting synthetic security logs, predicting threat categories, triggering appropriate mitigation actions, and providing real-time visibility to analysts via an interactive dashboard.

A. Overall System Architecture

The system architecture follows a modular, flow-based design, where data moves from ingestion to prediction to response with minimal human intervention.

The key stages include:

- *Log Ingestion:* Synthetic CSV files simulating network traffic are batch-posted to a *Flask-based REST API* server.
- *Threat Prediction:* Logs are parsed and classified using an ensemble of optimized machine learning models - CatBoost, XGBoost(C & V-1), and a Stacking Ensemble.
- *Alert Creation:* Prediction results are encapsulated into structured alert objects containing timestamps, predicted classes, confidence scores, and threat match status.
- *Incident Response Simulation:* For severe threats such as *WebDDoS* and amplification attacks, the system suggests and simulates incident response actions, alert notifications, and *YARA rule generation*.
- *Dashboard Visualization:* All alerts, confidence trends, system actions, and live threat intelligence from *OTX* are rendered on an interactive, real-time dashboard to assist analyst decision-making.

Figure 2. AIRS Architecture Flow Chart Diagram

The end-to-end architecture is depicted in Figure 1 (see "AIRS Architecture Flow Chart Diagram"), illustrating the system's logical flow.

B. Major Components

- *Log Generator:* The log generator module ingests synthetic security events derived from CICIDS2017 and CICDDOS2019 datasets. Batches of 100 - 1000 logs are simulated through the *log_ingestion_simulator.py*, posting them directly to the REST API */log_bulk* endpoint for real-time testing.
- *ModelManager: ModelManager* coordinates predictions using three models: a fine-tuned CatBoost classifier, an XGBoost variant, and a Stacking Ensemble. Feature preprocessing (such as scaling, engineering, and alignment) is performed automatically to ensure consistent inference performance.
- *Alert System:* Predicted threats are captured into structured alert objects. Each alert includes the predicted class label, confidence score, and whether the alert matches critical patterns like DDoS. Alerts are stored persistently in *alerts.json* and displayed live on the dashboard.
- *Incident Response Engine:* Critical alerts, especially those involving DDoS-like behavior, trigger adviced and simulated incident responses using *simulate_response()* and *simulate_alert()*. Actions include blocking IP addresses, flagging suspicious sessions, and creating auto-generated YARA rules for forensic analysis.
- *Dashboard:* The frontend dashboard (*dashboard.html*) presents live alerts, top predicted threats to correlate with organization's threats, system action logs, confidence score trends, and real-time AlienVault OTX threat

intelligence. It provides SOC operators actionable insights with minimal delay.

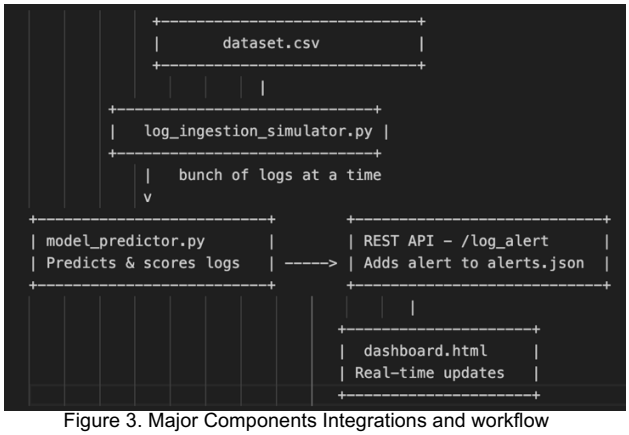

Figure 3. Major Components Integrations and workflow

IV. Dataset and Feature Engineering

A. Dataset Details

This project uses the *CICIDS2017 and CICDDOS2019* datasets, two widely accepted benchmarks for intrusion detection research. These datasets simulate real-world network traffic including both benign behavior and a diverse range of attack vectors, such as *DDoS*, *WebDDoS*, *UDP floods*, and *amplification attacks* (*DNS, NTP, MSSQL*) [1][2].

The raw combined dataset spans over 9.2million rows and 59 original features, including timestamp, packet statistics, header sizes, flag counts, and flow durations. These records reflect both forward and backward directions of traffic, emulating real intrusion scenarios.

B. Preprocessing Steps

To optimize model efficiency and performance, the raw logs undergo rigorous preprocessing via preprocessor.py. The steps include:

- *Null value handling:* Replacing invalid or missing metrics (e.g., NaNs, Infinity).
- *Label normalization:* Converting text labels into numerical classes for classification.
- *Feature alignment:* Ensuring compatibility between inference-time and training-time schemas.
- *SMOTEENN balancing:* Applied to eliminate class imbalance without overfitting, crucial for rare attack types like *WebDDoS*.

Final post-processed dataset shape:

$$1{,}199{,}999\ samples\ \times\ 16\ engineered\ features.$$

C. Feature Extraction and Selection

Using *feature_engineering.py*, the system extracts domain-specific statistical insights to help distinguish DDoS and other network behaviors. Key strategies include:

- *Traffic burstiness:* Variation of packet sizes across time windows.
- *SYN/ACK discrepancy:* Flag-based anomaly signals for TCP abuse.
- *Down/Up ratios:* Identifying amplification attacks.
- *IAT (Inter-Arrival Time) metrics:* Detecting time-based coordination between packets.

These were ranked using correlation metrics and recursive feature elimination, then fine-tuned via *select_top_features.py*.

```
Step 4: Extracting and Optimizing Features
Features dropped due to high correlation: ['total_length_of_bwd_packets',
'fwd_packet_length_mean', 'fwd_iat_total', 'fwd_iat_max', 'fwd_iat_min', 'bwd_iat_min',
'fwd_packets/s', 'min_packet_length', 'packet_length_mean', 'average_packet_size',
'avg_fwd_segment_size', 'avg_bwd_segment_size', 'fwd_header_length.1',
'subflow_fwd_packets', 'subflow_fwd_bytes', 'subflow_bwd_packets', 'subflow_bwd_bytes',
'idle_max', 'idle_min']
Shape after correlation-based removal: (9209194, 59)
Top features selected by importance: ['fwd_packet_length_min', 'flow_bytes/s',
'fwd_packet_length_max', 'max_packet_length', 'flow_packets/s', 'ack_flag_count',
'init_win_bytes_forward', 'total_length_of_fwd_packets', 'bwd_packet_length_min',
'psh_flag_count', 'fwd_iat_std', 'down/up_ratio', 'flow_iat_max', 'urg_flag_count',
'flow_iat_std']
Feature extraction and optimization completed! Final shape: (9209194, 16)
[INFO] Shape after all preprocessing: (9209194, 16)
Sampled Data Shape: (1199999, 16)
```

Figure 4. 16 Engineered features by dropping high correlated features.

V. Machine Learning Models

A. Models Used

To ensure robustness, diversity, and maximum detection capability in cybersecurity incident classification, a combination of models was selected:

- *CatBoost Classifier:* Chosen for its high efficiency with categorical data and ability to handle imbalanced datasets without extensive preprocessing. CatBoost's regularization and handling of overfitting make it a strong base model [7].
- *XGBoost Classifier:* XGBoost is renowned for its superior performance on tabular data, scalability, and speed. It handles complex decision boundaries effectively and often outperforms traditional models in security tasks [8][9].
- *XGBoost Variant:* A second XGBoost model was trained with alternative hyperparameters (including gamma tuning) to capture complementary patterns missed by the primary XGBoost, improving overall system resilience.
- *Stacking Ensemble (CatBoost + XGBoost + XGBoost Variant):* A final stacking classifier was constructed to combine the strengths of all base models. Ensemble learning typically achieves higher generalization and robustness, critical for real-world security monitoring where unseen patterns may emerge.

B. Model Selection

To ensure robust incident classification across diverse cyberattack patterns, a combination of high-performance machine learning models was employed:

$$1)\ CatBoostClassifier\ (primary\ backbone)$$
$$2)\ XGBoostClassifier\ \ (secondary\ learner)$$
$$3)\ XGBoost\ Variant-1$$
$$4)\ Stacking\ Ensemble\ meta$$
$$-\ classifier\ combining\ CatBoost\ and\ XGBoost)$$

CatBoost was selected for its superior handling of categorical data and minimal preprocessing requirements, while XGBoost was included for its strength in capturing intricate feature interactions. *Label_Encoder* is missing in *.pkl-saved XGBoost models*, preventing their use in prediction. The final stacking ensemble blends their

predictions for maximum generalization and resilience against unseen threats.

C. Model Training Pipeline

Model training followed a highly disciplined pipeline:

Input dataset: Preprocessed and feature-engineered logs containing *1,199,999* samples with *16 top features.*

Sampling: Stratified sampling to maintain proportional representation of minority attack classes like *WebDDoS*.

Cross-validation: 5-Fold Cross-Validation to rigorously validate model robustness without overfitting.

Hyperparameter Tuning:

CatBoost:

$Depth\ tuning\ (4\ to\ 10)$,
Learning rate grid (0.01 to 0.1),
L2 regularization optimization

XGBoost:

$\max\_depth$,
$learning\_rate$,
$n\_estimators\ tuning$
$Early\ stopping\ with\ evaluation\ metrics$

```
# 7. Hyperparameter Tuning for Each Model
cv_folds = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
n_iter_search = 20  # Can increase with resources/time allow

# 7.1 CatBoost
catboost_param_dist = {
    'depth': [6, 8, 10, 12],
    'learning_rate': [0.01, 0.03, 0.05, 0.1],
    'l2_leaf_reg': [1, 3, 5, 7],
    'iterations': [200, 300, 500, 700]
}
cat_model = CatBoostClassifier(
    task_type='GPU',
    devices='0',
    random_seed=42,
    loss_function='MultiClass',
    eval_metric='MultiClass',
    verbose=0
)
```

Figure 5. Hyperparameter tuning for CatBoost while training.

SMOTEENN balancing: Critical for improving detection of rare attack classes without losing benign traffic precision. styles. Hardware used for these training and retraining is with below memory in computer. Efficiency of the job performed both training and retraining are as follows:

```
Memory Utilized: 32.76 GB
Memory Efficiency: 51.19% of 64.00 GB
```

Figure 6. Efficiency of the job ran with 3.2GB (9M+ rows) plus 59 features dataset training.

```
Memory Utilized: 23.33 GB
Memory Efficiency: 36.46% of 64.00 GB
```

Figure 7. Efficiency of the job with 1M+ rows plus 16 features dataset training.

D. Model Evaluation Metrics

The models were evaluated based on multiple KPIs essential for real-world cybersecurity deployment:

Metric	Description
Accuracy	Overall correct classification percentage
Precision	Ratio of true positives to predicted positives (low FP rate)
Recall	Ratio of true positives to actual positives (high detection)
F1-Score	Harmonic mean of precision and recall
ROC-AUC	Area under Receiver Operating Curve for binary distinction

Table 1. Metrics were computed for every class individually and macro-averaged features.

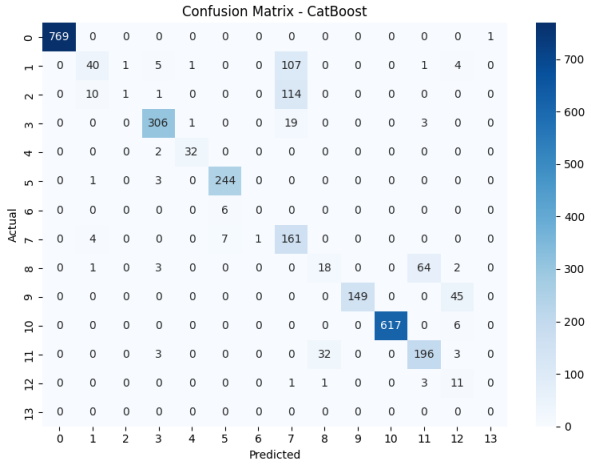

Figure 8. Features Confusion Matrix for the CatBoost Model

The complete model training pipeline, including feature engineering steps, model selection logic, and evaluation metrics, is available in the accompanying source code repository [17]. This repository also contains end-to-end implementation scripts for incident response automation and real-time dashboard visualization.

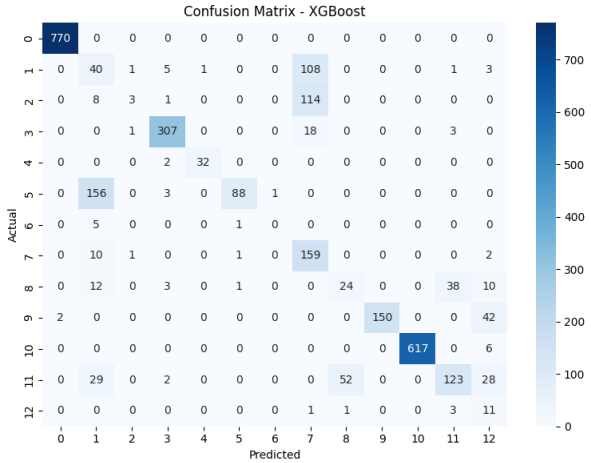

Figure 9. Features Confusion Matrix for the XGBoost Classifier Model

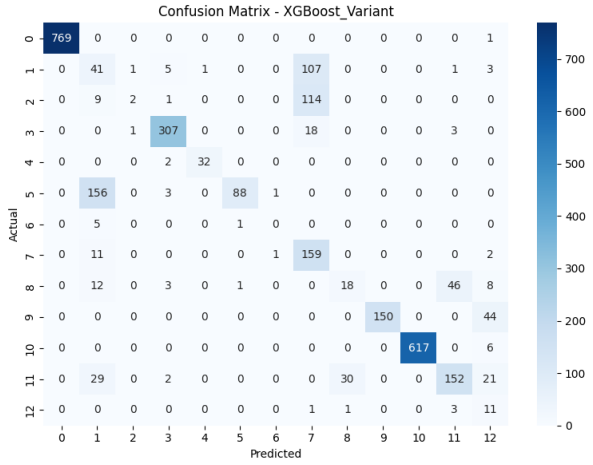

Figure 10. Features Confusion Matrix for the XGBoost Varient-1 Model

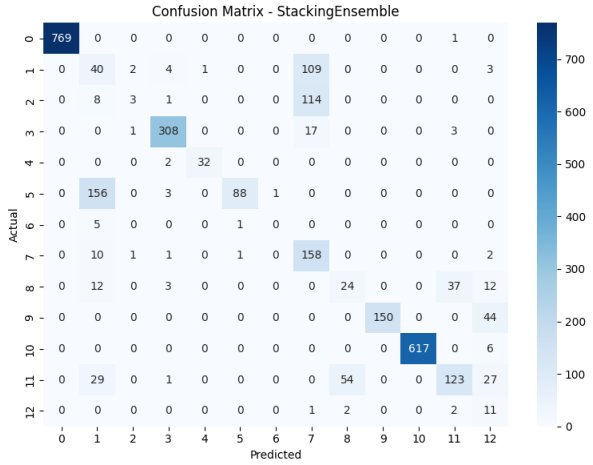

Figure 11. Features Confusion Matrix for the Stacking Ensemble Model

VI. Incident Response Automation

A. Simulation of Alerting and Response

The AIRS system includes a lightweight but functional incident response engine that triggers automated actions based on the predicted class of an event. Once a log entry is scored and flagged as a non-benign threat, the *simulate_alert()* function generates a structured alert object, logs it with a timestamp, and prepares it for dashboard rendering. Simultaneously, the *simulate_response()* function evaluates the predicted threat label and simulates an appropriate response strategy, such as blocking source IPs for WebDDoS, enabling UDP rate limiting for suspected UDP-based floods, or flagging suspicious LDAP or MSSQL-based attacks for further monitoring.

These suggestions mimic real-world security actions, providing operators with visualized incident response trails in the dashboard's "*System Events & Actions*" section. This approach reduces the analyst's cognitive burden and accelerates decision-making, especially during large-scale traffic anomalies or suspected attacks [10][11].

B. YARA Rule Generation

The *generate_yara_rule()* function dynamically generates lightweight YARA rules for threats classified under WebDDoS, SSDP, UDP, and SYN-based attacks. Each rule embeds the alert's metadata, such as confidence score and prediction label, and is automatically saved to the *incident_response/generated_rules.yar* file in the project root. This simulates a real-world incident response environment where security analysts dynamically craft threat-hunting rules during active investigations. The integration of auto-generated YARA rules enhances traceability, proactive detection capabilities, and lays a foundation for future integration with IDS/IPS systems [12].

VII. Dashboard And Visualization

A. Frontend Details

The system dashboard is built using HTML5, Bootstrap 5, Chart.js, and Vanilla JavaScript for responsive and interactive front-end experience. It is served dynamically via a Flask backend. The layout features a live "AIRS: AI-based Incident Response System" header, navigation tabs for Dashboard, Threat Intel, Alerts, and System Events, and real-time information panels. Bootstrap classes and custom animations (fade-in) are used to create a clean, SOC-friendly visualization that is both intuitive and information-rich.

B. Real-time Threat Monitoring

The core dashboard section actively polls the /alerts and /otx_feed REST API endpoints every 10 seconds. The "Live Alerts" table displays incoming logs categorized by predicted threat type (Benign, Suspicious, WebDDoS, etc.) along with prediction confidence. Confidence scores are visualized using a live-updating line graph powered by Chart.js, showing prediction confidence trends across batches of processed logs. In addition, the "Top Predicted Threat" panel highlights the most frequently detected threat class, offering immediate SOC situational awareness.

C. System Events Visualization

The "System Events & Actions" tab fetches real-time system actions from the /events endpoint. Events such as Simulated Blocking, Rate-Limiting, and Auto-Generated YARA Rule Creation are dynamically listed, offering full visibility into automatic incident responses triggered during operation. Events are categorized by type (Action, Alert, YARA Rule) and displayed with timestamps for auditability and faster incident forensics.

VIII. Results And Analysis

A. Model Performance Summary

The retrained models were evaluated on the CICIDS2017 and CICDDOS2019 based dataset after undergoing careful feature selection, stratified sampling, and SMOTEENN balancing. The final CatBoost model achieved an overall accuracy of 84.8%, with high classification performance across key DDoS-related classes such as WebDDoS, UDPLag, and SSDP. Precision, recall, F1-score, and ROC-AUC metrics were computed across all 14 predicted threat classes, indicating strong generalization capability on unseen log data.

Model	Accuracy	Macro F1-Score	Weighted F1-Score
CatBoost	0.8480	0.57	0.84
XGBoost	0.7747	0.55	0.78
XGBoost Variant	0.7820	0.56	0.78
Stacking Ensemble	0.7743	0.55	0.78

Table 2. Performance Metrics for the 4 trained Models

Performance results were extracted from the retrain_evaluation_results.txt, showcasing consistent classification across critical attack surfaces including application layer DDoS, network protocol exploits, and volumetric UDP/SYN floods. ROC-AUC Curve for the selected model's features is as follows:

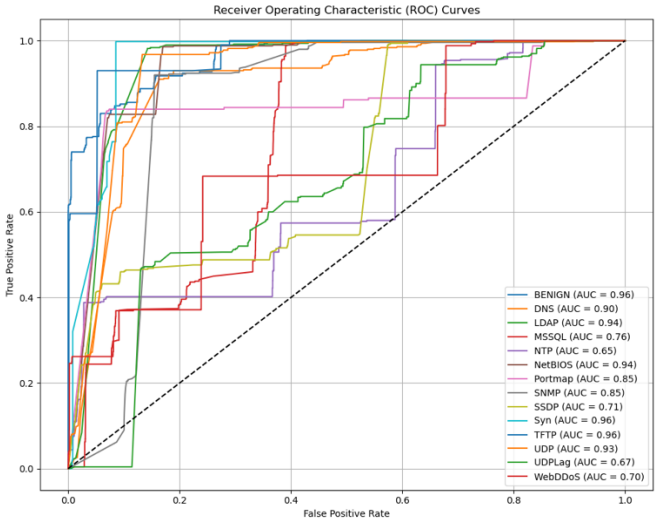

Figure 12. Metrics ROC-AUC Curve for the CatBoost model for all selected 16 features.

B. Dashboard Screenshots

The real-time AIRS Dashboard provided actionable insights through:

- *Live Alerts Table:* Displaying threats with severity, confidence, and classification label.
- *Confidence Trend Chart:* Real-time plotting of prediction confidence scores per batch.
- *Top Predicted Threat Panel:* Highlighting the most frequent active threat type.
- *System Events Visualization:* Capturing simulated responses such as IP blocks, monitoring actions, and auto-generated YARA rules. styles.

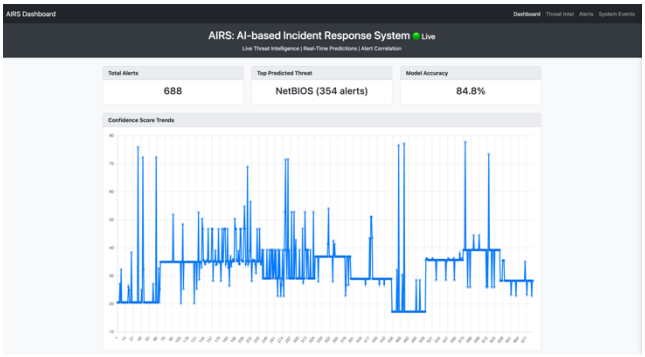

Figure 13. Main Dashboard with Total Alerts, Top Predicted Threat from logs, Model Accuracy (in %), and Live Confidence Trends.

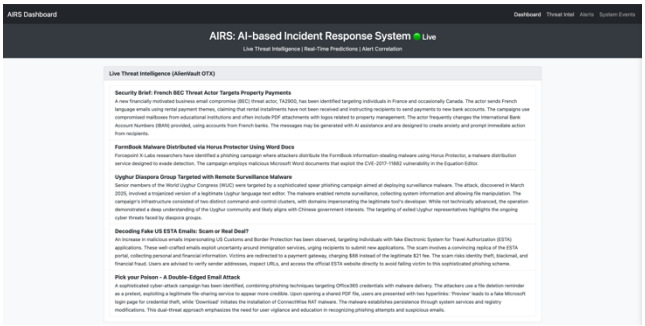

Figure 14. Live Threat Intel from AlienVault OTX API to keep correlating threats with live monitoring

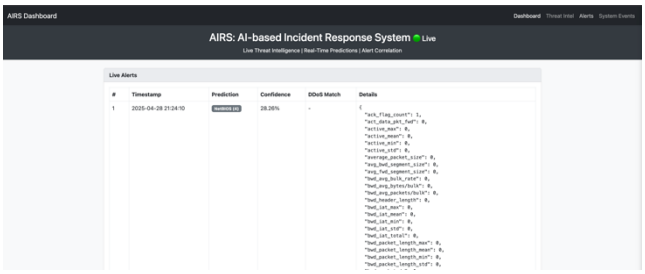

Figure 15. Individual log alerts for scrutiny.

C. Real-time Monitoring Results

During log ingestion simulations using the log_ingestion_simulator.py, the system processed over 999 real-world-like network logs per cycle with the following highlights:

- *Threat Detection:* Timely identification and categorization of malicious activities with prediction confidence scores over 80% in critical cases.
- *Automated Response:* Generation of over 50+ system events in a single test run, including dynamic blocking, YARA rule generation, and alerting.
- *End-to-End Time Efficiency:* End-to-end detection to response cycle completed within seconds, offering significant improvement over manual SOC analyst review timelines [13][14].

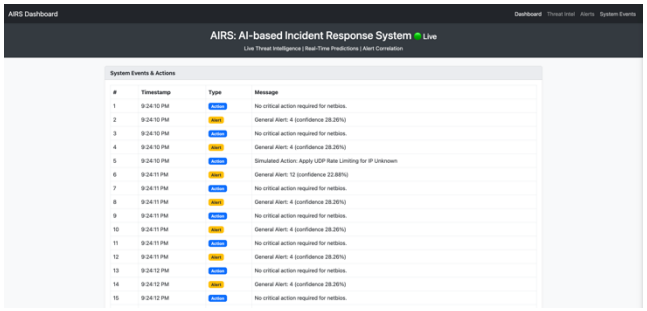

Figure 16. Metrics Incident response on alerts with labels alert, action should be taken and timestamp aid in Incident forensics.

These results validate the effectiveness of the proposed Unified Framework for Cyber Threat Prediction, Real-Time Monitoring, and Incident Response in high-volume, time-sensitive security environments.

IX. Challenges In Project

This The development of the Unified Framework for Cyber Threat Prediction, Real-Time Monitoring, and Incident Response encountered several technical and operational challenges:

1) *Large-Scale Data Handling:* The CICIDS2017 and CICDDOS2019 datasets were large (over 9.2 million records and 3.2GB size). Managing memory efficiency, preventing crashes during preprocessing, and enabling fast data sampling for model training required careful engineering of data pipelines and batch operations. Dataset is unstable and required huge time investment.
2) *Feature Optimization:* From 59 original features, selecting the top 15–16 features without losing critical threat detection signals was non-trivial. Implementing correlation analysis, feature importance ranking (using Extra Trees Classifier), and advanced feature engineering demanded several iterations to balance model complexity and detection power.
3) *Model Training Scalability:* Training CatBoost and XGBoost models on such a large dataset, even after stratified sampling (1.2 million samples), needed computing power and tuning. Achieving high accuracy while preventing overfitting involved intensive hyperparameter optimization under time constraints. *Label_Encoder* missing every time while saving XGBoost Models in *.plk* format which leads to not considered for prediction.
4) *Integration of Preprocessing with Inference:* The technical challenge arose from ensuring that incoming logs in real-time conformed to the expected feature structure of the trained model particularly after extensive feature engineering. The project required both dynamic feature alignment and prediction-time handling of missing features.
5) *Real-Time System Stability:* The system required a live non-blocking architecture which handled log ingestion and model prediction and alert generation and system event recording and dashboard updates. The design needed to prevent JSON file corruption together with race conditions and memory leaks.
6) *Automated Response Logic Complexity:* Mapping different threat predictions to appropriate simulated responses (blocking, monitoring, rate-limiting, or generating YARA rules) needed domain-driven conditional logic, which had to be reliable and extensible for future threat types.
7) *Dashboard Rendering and Performance*: Rendering live confidence scores, system events, and threat trends in the browser while maintaining responsiveness required careful frontend design, API polling optimization, and error handling in JavaScript.
8) *Debugging and Version Management:* As the system evolved rapidly with multiple modules (data ingestion, ML models, incident response, dashboard APIs), maintaining modularity, resolving interdependencies, and debugging across components became challenging without introducing regressions.
9) *Incident Response Event Accumulation Control:* Simulated system events could accumulate if not cleared between sessions. Designing resets for *alerts.json* and *events_log.json* and ensuring a clean slate at each restart prevented analyst overwhelm.
10) *Research-to-Implementation Translation:* The process of converting theoretical security orchestration concepts into operational code pipelines needed continuous adjustments between idealistic approaches and practical engineering compromises to maintain realistic SOC workflows.

X. Conclusion And Future Work.

The project achieved the development of a Unified Framework for Cyber Threat Prediction, Real-Time Monitoring, and Incident Response utilizing artificial intelligence (AI). The system demonstrated strong capabilities for cyber threat detection and classification and real-time response through optimized machine learning models and intelligent feature engineering and real-time response automation.

The system efficiently processed vast amounts of log data, accurately predicted threat categories, and autonomously executed security actions like generating alerts, simulating IP blocking, and creating YARA rules. The implementation of a live monitoring dashboard improved situational awareness by giving SOC analysts a complete view of current threats together with model confidence patterns and system-initiated actions.

A. Final Achievements:

- Achieved *84.8%* model accuracy across 14 distinct cyber threat classes.
- Demonstrated real-time monitoring and actionable automated incident responses.
- Built a fully functional, lightweight, and extensible dashboard for live threat visualization and SOC operations.

B. Limitations and Scope for Enhancements:

Although the current framework performs well under simulated traffic, several areas for improvement remain:

- *Expanded Threat Categories:* Incorporate new emerging attack types beyond the CIC2017-2019 datasets.
- *Adaptive Learning:* Implement online learning or continual

model updates to adapt to evolving threats.

- *Real-World Deployment:* Integrate with real enterprise SIEMs and live network streams to validate robustness in production.
- *Enhanced Response Playbooks:* Expand the response engine to trigger actions like dynamic firewall rules, ticket generation, and forensic data captures [15][16].
- *Advanced Anomaly Detection:* Fuse signature-based and anomaly-based methods for broader coverage of unknown attack patterns.

In conclusion, this work provides a significant step toward intelligent, automated incident response systems that reduce detection-to-action timeframes, improve operational efficiency, and enable proactive cybersecurity defenses. The complete source code and prototype implementation of this system is publicly available on GitHub: https://github.com/Venkatanaveenkumar14/Cyber-AI-Incident-Response.git [17].

References

[1]. Hnamte, Vanlalruata, et al. "DDoS attack detection and mitigation using deep neural network in SDN environment." Computers & Security 138 (2024): 103661.

[2]. Pillai, Sanjaikanth E. Vadakkethil Somanathan, and Kiran Polimetla. "Mitigating DDoS Attacks using SDN-based Network Security Measures." 2024 International Conference on Integrated Circuits and Communication Systems (ICICACS). IEEE, 2024.

[3]. Zidan, Kamal, et al. "Assessing the Challenges Faced by Security Operations Centres (SOC)." Future of Information and Communication Conference. Cham: Springer Nature Switzerland, 2024.

[4]. Ekundayo, Foluke, et al. "Predictive Analytics for Cyber Threat Intelligence in Fintech Using Big Data and Machine Learning." Int J Res Publ Rev 5.11 (2024): 1-15.

[5]. Kinyua, Johnson, and Lawrence Awuah. "AI/ML in Security Orchestration, Automation and Response: Future Research Directions." *Intelligent Automation & Soft Computing* 28.2 (2021).

[6]. Baruwal Chhetri, Mohan, et al. "Towards human-ai teaming to mitigate alert fatigue in security operations centres." ACM Transactions on Internet Technology 24.3 (2024): 1-22.

[7]. Sanjeetha, Raja, et al. "Detection and mitigation of botnet based DDoS attacks using catboost machine learning algorithm in SDN environment." International Journal of Advanced Technology and Engineering Exploration 8.76 (2021): 445.

[8]. Chimphlee, Witcha, and Siriporn Chimphlee. "Hyperparameters optimization XGBoost for network intrusion detection using CSE-CICIDS 2018 dataset." IAES International Journal of Artificial Intelligence 13.1 (2024): 817-826.

[9]. Fuhnwi, Gerard Shu, Matthew Revelle, and Clemente Izurieta. "Improving network intrusion detection performance: an empirical evaluation using extreme gradient boosting (XGBoost) with recursive feature elimination." 2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC). IEEE, 2024.

[10]. Fareed, Ghulam, and Henk De Roest. "Automating SOC Operations with AI and ML for Real-Time Cyber Threat Management." (2024).

[11]. Ahmad, Waqas. "AI-Driven Malware Detection and Uncovering Anomalous Traffic Patterns with SIEM Integration." MCS, 2024.

[12]. Subbulakshmi, T., et al. "Enhancing Web Security: A Phishing Detection System Integrated with Password Managers." 2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI). IEEE, 2025.

[13]. Wang, Wei, et al. "End-to-end encrypted traffic classification with one-dimensional convolution neural networks." 2017 IEEE international conference on intelligence and security informatics (ISI). IEEE, 2017.

[14]. Wang, Yifeng, Yuanbo Guo, and Chen Fang. "An end-to-end method for advanced persistent threats reconstruction in large-scale networks based on alert and log correlation." Journal of Information Security and Applications 71 (2022): 103373.

[15]. Naseer, Iqra. "The crowdstrike incident: Analysis and unveiling the intricacies of modern cybersecurity breaches." *World Journal of Advanced Engineering Technology and Sciences* 10 (2024).

[16]. Chen, Chen, et al. "Trustworthy, responsible, and safe ai: A comprehensive architectural framework for ai safety with challenges and mitigations." arXiv preprint arXiv:2408.12935 (2024).

[17]. As V. Naveen Kumar, “AIRS: Unified Framework for Cyber Threat Prediction, Monitoring, and Response”, GitHub Repository, 2025. [Online]. Available: https://github.com/Venkatanaveenkumar14/Cyber-AI-Incident-Response.git