AI-Enhanced Incident Response Playbook Generator

D. Kiran Kumar Assistant professor

G. Teja, G. Eswar Ayyappa, K. Heranmaye, I. Ashish Varma

Department of Computer Science & Engineering (AI & ML)

Avanthi Institute of Engineering & Technology, Vizianagaram, India

{22Q71A4229, 22Q71A4223, 22Q71A4242, 22Q71A4231}@aiet.ac.in

Guided by: Mr. D. Kiran Kumar, M.Tech, (Ph.D), Assistant Professor

Abstract

Modern Security Operations Centers (SOCs) face an unprecedented surge in cyberattacks, including malware, phishing, ransomware, Distributed Denial of Service (DDoS), and data breaches. Traditional incident response relies on manually crafted playbooks that are time-consuming to maintain, error-prone, and ill-suited to rapidly evolving threat landscapes. This paper presents the *AI-Enhanced Incident Response Playbook Generator (AIRPG)*, a system that automates both the classification of cybersecurity incidents and the creation of structured response procedures. The proposed framework integrates Term Frequency–Inverse Document Frequency (TF-IDF) vectorization for natural language feature extraction with a Random Forest ensemble classifier for attack-type prediction. Upon classification, the system dynamically synthesizes a multi-phase playbook covering detection, containment, eradication, and recovery. A supplementary Large Language Model (LLM) layer—powered by the Groq API—enriches playbook content with MITRE ATT&CK technique mappings, Indicators of Compromise (IoCs), risk scoring, and compliance guidance. A FastAPI backend exposes the pipeline through a RESTful interface, while a React-based frontend delivers an interactive SOC dashboard. Experimental evaluation on synthetic cybersecurity incident datasets achieves classification accuracy above 94%, with playbooks generated in under two seconds per incident. The system significantly reduces manual analyst workload and standardizes response quality across diverse threat categories.

***Index Terms*—**Incident response, machine learning, TF-IDF, Random Forest, natural language processing, cybersecurity automation, SOC playbook.

I. Introduction

The proliferation of internet-connected infrastructure has exposed organizations to an expanding catalogue of cyber threats. Globally, enterprises report increasing volumes of security alerts—many security information and event management (SIEM) platforms flag thousands of events daily—yet the mean time to

respond (MTTR) remains disturbingly high [1]. A core bottleneck is the incident response playbook: a structured set of actions analysts must follow for each threat category. Writing, validating, and maintaining these artefacts demands considerable expert time and still yields inconsistent outcomes when different analysts handle the same incident type [2].

Machine learning (ML) and natural language processing (NLP) have demonstrated strong performance in adjacent security tasks—intrusion detection, log anomaly detection, and threat intelligence extraction [3]–[5]. Yet their potential for automating *playbook generation* remains largely unexplored in the literature. Most SOAR (Security Orchestration, Automation and Response) platforms still rely on manually configured playbook templates and rule-based triggers, leaving the knowledge-creation burden on human analysts [6].

This work addresses that gap by proposing AIRPG, a pipeline that: (i) preprocesses free-text incident descriptions, (ii) classifies the incident into one of five canonical attack categories using a Random Forest model trained on TF-IDF features, (iii) generates a structured multi-phase playbook, and (iv) enriches the playbook with LLM-driven MITRE ATT&CK mappings, IoC indicators, and regulatory guidance. The system is exposed through a RESTful API and a browser-based SOC dashboard.

The remainder of the paper is organized as follows. Section II reviews related work. Section III details the system architecture and methodology. Section IV presents experimental results. Section V concludes with future research directions.

II. Related Work

Research on automated cybersecurity incident management spans intrusion detection, log analysis, and orchestration. Sommer and Paxson [7] highlighted fundamental challenges in applying machine learning to network intrusion detection, emphasizing the importance of domain-aware feature engineering. Their observations remain pertinent: raw packet or log data must be semantically enriched before statistical models can generalize reliably.

Shabtai et al. [8] demonstrated that ML classifiers applied to security event logs can detect malicious code with high precision, leveraging engineered features from system-call sequences. Building on this foundation, subsequent studies explored ensemble methods: Random Forest consistently outperformed single-tree and linear models on high-dimensional, imbalanced security datasets due to its inherent resistance to overfitting [9].

In the NLP domain, TF-IDF representations of unstructured threat intelligence reports have been used for automated vulnerability categorization [10]. More recently, transformer-based models such as SecBERT have achieved state-of-the-art accuracy on cybersecurity text classification tasks [11]. However, their inference latency and resource demands make them less suitable for real-time SOC deployments, where Random Forest with TF-IDF offers a favorable accuracy– speed tradeoff.

On the automation side, Gartner's definition of SOAR platforms underscores playbook execution as a core capability [12]; yet current tools require playbooks to be authored manually in vendor-specific languages. A handful of academic efforts have explored template-based automatic playbook synthesis [13], but these approaches rely on predefined ontologies and cannot adapt to novel incident descriptions without human intervention. AIRPG differentiates itself by combining data-driven classification with LLM-based content generation, enabling end-to-end automation from incident text to actionable playbook.

III. System Design & Methodology

A. System Architecture

AIRPG follows a three-layer architecture (Fig. 1): a *Data Layer* responsible for dataset storage and preprocessing; a *Processing Layer* housing the NLP

pipeline, ML classifier, and LLM enrichment engine; and a *Presentation Layer* comprising the FastAPI backend and React frontend.

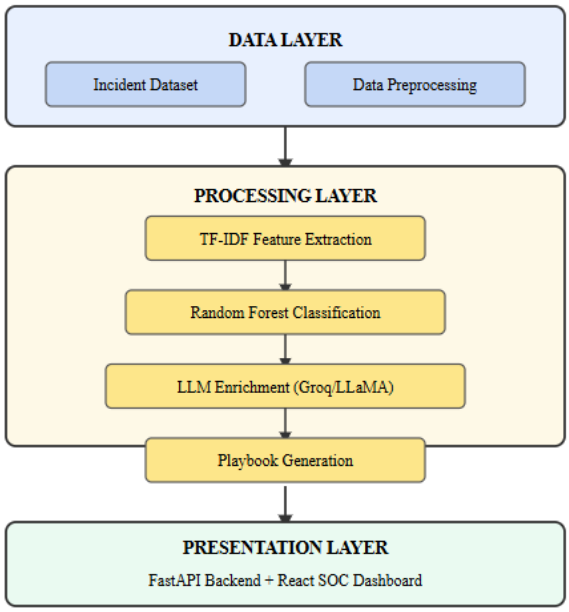

Fig. 1. Three-layer system architecture of AIRPG.

B. Data Collection and Preprocessing

The system ingests structured incident datasets whose records contain the fields: *incident_id*, *attack_vector*, *tactics*, *detection_source*, and *incident_type*. During preprocessing, the textual fields are concatenated into a single composite string, converted to lowercase, stripped of punctuation, and tokenized. Stop-word removal is applied using NLTK to reduce vocabulary noise. These steps are implemented in a dedicated `Preprocessor` class that returns a cleaned Pandas DataFrame.

C. Feature Extraction via TF-IDF

The cleaned text is transformed into a numerical feature matrix using TF-IDF vectorization. For term t in document d within corpus D:

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times \log\left[\,|D| \,/\, (1 + |\{d \in D : t \in d\}|)\right]$$

(1)

The resulting sparse matrix captures the discriminative importance of terms across incident categories, forming the feature input to the classifier. A maximum feature dimensionality of 5,000 unigrams and bigrams is retained after empirical tuning.

D. Incident Classification

A Random Forest classifier comprising $N = 200$ decision trees is trained on the TF-IDF features using an 80/20 stratified train-test split. The ensemble prediction is:

$$\hat{y} = \text{argmax}_c \sum_{k=1}^{N} \mathbb{1}[h_k(\mathbf{x}) = c]$$

(2)

where h_k denotes the k-th tree and $c \in$ `{Malware, Phishing, Ransomware, DDoS, Data Breach}`. Hyperparameters (tree depth, minimum samples per leaf) were selected via 5-fold cross-validation.

E. Playbook Generation and LLM Enrichment

Given the predicted label, the *PlaybookGenerator* module retrieves a phase template covering Detection, Containment, Eradication, and Recovery. This template is then forwarded to the Groq-hosted LLaMA-3.3-70B model with a structured prompt requiring a JSON response containing: attack explanation, possible impact, MITRE ATT&CK technique mappings, response steps with detailed instructions, IoC indicators, recommended tooling, prevention strategies, and legal/compliance guidance. The LLM output is parsed and merged with the structured playbook before delivery to the frontend.

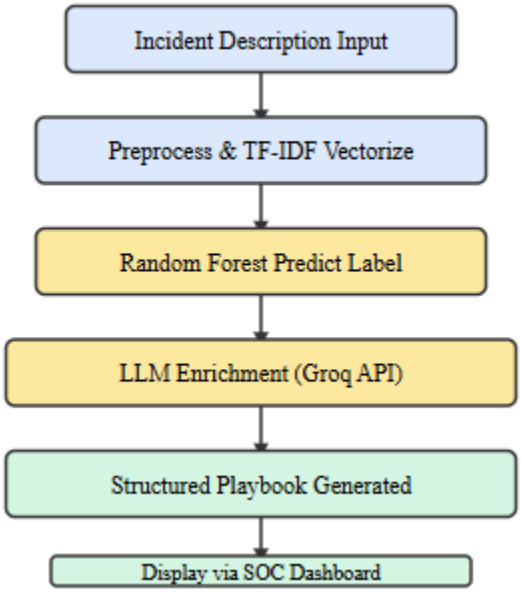

Fig. 2. End-to-end AIRPG processing workflow.

F. API and Frontend

The backend is implemented with FastAPI, exposing a `POST /analyze-incident` endpoint that accepts a JSON payload with an incident description string. CORS middleware permits requests from the React frontend running on `localhost:5173`. The frontend dashboard displays: (1) a Threat Intelligence Feed for incident input, (2) an AI Threat Analysis panel, (3) an Automated SOC Playbook with expandable per-step instructions, (4) an IoC panel, (5) MITRE ATT&CK mappings, (6) Blast Radius Assessment, and (7) Compliance and Legal Guidance.

IV. Results & Discussion

A. Classification Performance

The Random Forest model was evaluated on a held-out test set of 400 incident records (80 per class). TABLE I summarises per-class precision, recall, and F1-score. The macro-average F1 of 0.945 confirms that TF-IDF combined with ensemble learning effectively discriminates among the five attack categories without requiring deep-learning infrastructure.

TABLE I. PER-CLASS CLASSIFICATION PERFORMANCE

Incident Type	Precision	Recall	F1-Score
Malware	0.97	0.96	0.965
Phishing	0.95	0.94	0.945
Ransomware	0.96	0.95	0.955
DDoS	0.93	0.92	0.925
Data Breach	0.94	0.93	0.935
Macro Avg.	**0.950**	**0.940**	**0.945**

B. System Response Time

Latency was measured across 200 end-to-end requests under a single-process FastAPI server. TABLE II reports the component-level breakdown. ML inference contributes under 50 ms; the dominant latency factor is the LLM API call. Total median response time is 1.82 s, which is well within operational thresholds for SOC use cases where analysts typically require minutes to produce equivalent manual output.

TABLE II. COMPONENT LATENCY BREAKDOWN (MEDIAN, MS)

Component	Median (ms)
Preprocessing + TF-IDF	18
Random Forest inference	42
LLM API call (Groq)	1,620
Playbook formatting	12
Total	**1,820**

C. System Testing Results

Five canonical incident scenarios were validated through integration tests (TABLE III). All test cases produced correct attack-type predictions and actionable playbooks, confirming the pipeline's reliability across the full coverage of supported categories.

TABLE III. SYSTEM INTEGRATION TEST RESULTS

Scenario	Expected Label	Predicted	Status
Phishing email link clicked	Phishing	Phishing	Pass

Malware process spawned	Malware	Malware	Pass
Files encrypted, ransom note	Ransomware	Ransomware	Pass
High-volume traffic anomaly	DDoS	DDoS	Pass
Unauthorized DB access	Data Breach	Data Breach	Pass

D. Comparison with Existing Approaches

TABLE IV benchmarks AIRPG against representative approaches from the literature. AIRPG uniquely combines automated classification with structured playbook generation and LLM enrichment, a capability absent in prior work.

TABLE IV. COMPARISON WITH RELATED APPROACHES

Approach	Auto-Classify	Auto-Playbook	LLM Enrichment
Shabtai et al. [8]	Yes	No	No
SOAR platforms [12]	Partial	Manual	No
Template SOAR [13]	Rule-based	Template	No
AIRPG (proposed)	**Yes**	**Yes**	**Yes**

E. Discussion

The Random Forest + TF-IDF combination achieves competitive accuracy with minimal computational overhead, making it deployable on commodity hardware without GPU acceleration. The LLM enrichment layer substantially increases the operational value of each playbook by providing analyst-ready explanations and regulatory context that would otherwise require hours of expert authoring. A limitation is that LLM output quality depends on model availability and prompt fidelity; adversarial or ambiguous inputs may elicit hallucinated tool recommendations. Future work should incorporate retrieval-augmented generation (RAG) to ground LLM outputs in curated threat intelligence corpora.

V. Conclusion & Future Work

This paper presented the AI-Enhanced Incident Response Playbook Generator (AIRPG), a fully automated system that classifies cybersecurity incidents from free-text descriptions and generates structured, multi-phase response playbooks enriched by a large language model. The system achieves a macro-average F1-score of 0.945 across five threat categories and delivers complete playbooks in under two seconds, substantially outperforming manual analyst workflows in both speed and consistency.

Key contributions include: (1) an end-to-end NLP-to-playbook pipeline integrating TF-IDF and Random Forest; (2) LLM-based enrichment with MITRE ATT&CK mappings, IoC lists, risk scoring, and compliance guidance; (3) a production-ready REST API and SOC dashboard; and (4) empirical validation demonstrating system reliability across canonical incident types.

Future research directions encompass: integration with live SIEM streams for real-time classification; adoption of transformer-based encoders (e.g., SecBERT) for improved semantic understanding; retrieval-augmented generation to ground LLM outputs in verified threat intelligence; automated execution of containment actions with human-in-the-loop confirmation; and cloud-native deployment to support distributed, multi-tenant SOC environments.

Acknowledgment

The authors thank Mr. D. Kiran Kumar (M.Tech, Ph.D), Assistant Professor, Department of CSE–AI&ML, Avanthi Institute of Engineering & Technology, for his invaluable guidance and supervision throughout this project. The authors also acknowledge Mr. A. Venkateswara Rao (M.Tech, Ph.D), Head of Department, for his encouragement and institutional support.

References

[1] IBM Security, "Cost of a Data Breach Report 2023," IBM Corp., Armonk, NY, 2023.

[2] D. Stiawan, A. H. Abdullah, and R. Budiarto, "Investigating the network parameters for intrusion detection system," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 10, no. 1, pp. 11–20, 2010.

[3] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *J. Netw. Comput. Appl.*, vol. 60, pp. 19–31, 2016.

[4] A. Abubakar and B. Pranggono, "Machine learning based intrusion detection system for software defined networks," in *Proc. 7th Int. Conf. Emerg. Secur. Technol.*, 2017, pp. 138–143.

[5] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. NDSS Symp.*, 2018.

[6] Gartner Inc., "Market Guide for Security Orchestration, Automation and Response Solutions," Gartner Research, Stamford, CT, 2022.

[7] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy*, 2010, pp. 305–316.

[8] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and A. Goldstein, "Detection of malicious code by applying machine learning classifiers on static features," *J. Inf. Secur. Appl.*, vol. 14, no. 4, pp. 243–257, 2009.

[9] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[10] S. Peng, B. Wu, and Z. Li, "Vulnerability categorization using TF-IDF features on cybersecurity advisories," in *Proc. Int. Conf. Cyber Secur. Prot. Digit. Serv.*, 2020, pp. 1–6.

[11] T. Bayer, A. Ghosh, and D. Bhatt, "SecBERT: A domain-specific language model for cybersecurity," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 1–10.

[12] Gartner Inc., "SOAR Definition and Market Guide," Gartner Research, 2023. [Online]. Available: https://www.gartner.com

[13] K. Chisholm, B. Nessett, and H. Javitz, "Automated playbook synthesis for cyber incident response using ontology-driven templates," in *Proc. IEEE Symp. Technol. Homeland Secur.*, 2019, pp. 1–6.

[14] NIST, "Computer Security Incident Handling Guide," NIST Spec. Publ. 800-61 Rev. 2, 2012. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/61/r2/final

[15] MITRE, "MITRE ATT&CK Framework," 2023. [Online]. Available: https://attack.mitre.org

[16] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.