Journal of Smart Algorithms and Applications
JSAA
ISSN: 3070-4189/
Journal Homepage
https://pub.scientificirg.com/index.php/JSAA

A Multi-Stage AI Framework for Intelligent Incident Detection and Response Automation in Security Operations Centers

Mina Adel[a,1], Aboulela Abdo[a], Doaa Sayed[a], Reham Darwish[a], Omar Moataz[a]

[a] *Faculty of Computer Science, Nahda University, Beni-Suef City, 62511, Egypt*
Email: M.adel0693@nub.edu.eg , Aboalela.abdo@nub.edu.eg , doaa.mokhtar@nub.edu.eg , Rehamdarwish97@gmail.com, o.moataz1394@nub.edu.eg

ABSTRACT

Security Operations Centers (SOCs) are essential for safeguarding company infrastructure against increasingly sophisticated cyber threats. The expansion of cloud services, Internet of Things (IoT) devices, and extensive dispersed systems has resulted in a significant increase in security event quantities, burdening analysts and revealing the inadequacies of conventional, manual workflows. Rule-based Security Information and Event Management (SIEM) systems, in particular, experience elevated false-positive rates, leading to alert fatigue and protracted incident response times. This paper introduces IRAS (Incident Response Automation and Management System), an AI-driven platform that automates key stages of the incident response lifecycle within enterprise SOC environments. IRAS integrates a two-stage machine learning pipeline combining Isolation Forest for anomaly detection and Random Forest for multi-class attack classification, coupled with Wazuh for centralized log management and deployed within a Dockerized microservice architecture on Ubuntu Linux. Upon detecting malicious activity, IRAS executes predefined mitigation playbooks including IP blocking, endpoint isolation, and account suspension within seconds of detection. IRAS was tested in a controlled experimental setup using an Ubuntu victim machine and a Kali Linux attack machine on the CIC-DDoS2019 and SSH Brute-Force datasets and got a weighted F1-score of 94.5% and, a false-positive rate of 2.8%, and a mean automated response time of 2.46 seconds ,representing a substantial reduction compared to manual SOC baselines of 8–15 minutes. The system achieved a peak throughput of 1,200 incidents per hour and reduced analyst workload by 74%, demonstrating significant improvements in SOC efficiency within the evaluated threat scenarios and providing a foundation for semi-autonomous SOC operations. Generalization to additional attack categories such as ransomware, insider threats, and APTs remains a direction for future validation.

PAPER INFORMATION

HISTORY

Received: 12 March 2026
Revised: 23 May 2026
Accepted: 21 June 2026
Online: 26 June 2026

MSC:

68T07; 68R10; 94A60; 68M15

KEYWORDS

Security Operations Center;
Automated Incident Response;
Anomaly Detection;
Machine Learning Security Analytics.

[1]*Corresponding author: Faculty of Computer Science, Nahda University, Beni-Suef City, 62511, Egypt*
Email: M.adel0693@nub.edu.eg

1. INTRODUCTION

The accelerated speed of digital transformation across industries has led to a major expansion of the cyber threat landscape in recent years. Distributed applications, mobile technologies, cloud computing platforms, and Internet of Things (IoT) devices are becoming more important components of modern enterprise infrastructures. Although these technologies improve operations, they also provide a large and expanding attack surface that adversaries aggressively exploit. There is an urgent need for intelligent, automated defense systems because the sophistication of contemporary cyber threats has surpassed the capabilities of traditional security mechanisms. These threats range from ransomware campaigns and advanced persistent threats (APTs) to large-scale distributed denial-of-service (DDoS) attacks and credential-based brute-force intrusions [1].

Security Operations Centers (SOCs) are the centralized hubs responsible for continuously monitoring organizational infrastructure, detecting security events, and orchestrating incident response. SOCs are key, but their effectiveness is undermined by increasing operational challenges. Modern environments produce tens of thousands of security alerts each day, more than human analyst teams can process. Traditional Security Information and Event Management (SIEM) solutions ingest and correlate events across a range of sources, including network traffic logs, endpoint telemetry, firewall events, and cloud audit trails. These systems may be proprietary or open source, such as Wazuh. But a large percentage of the warnings they generate, about 90% in untuned deployments, are false positives. This glut of warnings leads to analyst fatigue, delayed triage, and a higher likelihood of missing real dangers hiding in the noise [2].

The manual nature of incident response procedures compounds these challenges further. The subsequent response workflow, comprising alarm investigation, severity assessment, playbook selection, and containment action execution, is mainly conducted by human analysts, even if a real threat is successfully identified. Adversaries can take advantage of this detection-to-containment gap, which is generally between eight and fifteen minutes per event, to elevate privileges, move laterally, and persist in the target environment. Reducing the reaction window from minutes to seconds is critical, as studies indicate that the median time from initial compromise to lateral movement is approximately 47 minutes. Although machine learning has dramatically improved the accuracy of threat detection over the past few years, enabling systems to identify anomalous behavior and categorize attack types with high accuracy, the response phase of the incident lifecycle has been, for the most part, manual. Today's Security Orchestration, Automation, and Response (SOAR) technologies offer playbook-based automation but are not adaptive, data-driven detection, and are based on static, rule-based workflows. On the other hand, machine learning-based intrusion detection systems (IDS) provide intelligent detection capabilities but lack integrated automatic reaction mechanisms. This disconnect between detection and reaction is a fundamental limitation of the current generation of SOC tools [3], [4].

By automating the whole incident response lifecycle inside a single architecture, IRAS (Incident Response Automation and Management System) fills this gap. IRAS removes the human bottlenecks that define typical SOC operations and permits end-to-end automation from threat detection to active containment by closely integrating SIEM-based data ingestion, a two-stage machine learning detection pipeline, and an automated reaction engine.
The key aspects of the proposed approach can be summarized as follows:

(i) the design and implementation of an end-to-end AI-driven Security Operations Center (SOC) architecture integrating SIEM-based data ingestion, intelligent threat detection, and automated incident response within a unified framework, (ii) the development of a two-stage machine learning framework combining Isolation Forest for anomaly detection and Random Forest for multi-class attack classification, (iii) the implementation of an automated response engine capable of executing mitigation actions such as IP blocking, host isolation, and account disabling within seconds using predefined playbooks, (iv) the establishment of a unified framework that seamlessly bridges threat detection and incident response, enabling end-to-end automation of the incident response lifecycle, (v) the adoption of a scalable and modular architecture supporting real-time data processing through FastAPI and message queues, and (vi) the validation of the proposed SOC framework through comprehensive experiments on benchmark cybersecurity datasets, demonstrating superior detection accuracy and significantly reduced incident response times.

Unlike existing SOAR platforms that automate response through static playbooks without adaptive detection, and unlike ML-based IDS systems that detect threats without automated response, IRAS uniquely integrates both

capabilities into a single end-to-end pipeline, achieving automated containment in 2.46 seconds compared to 8–15 minutes in manual SOC baselines, as validated in **Table 1** and **Table 5**.

.

The remainder of this paper is organized as follows. Section 2 reviews related work on SOC automation and machine learning-based intrusion detection. Section 3 presents the problem statement and key challenges. Section 4 describes the architecture of the proposed IRAS system. Section 5 details the datasets and preprocessing techniques. Section 6 explains the machine learning methodology. Section 7 discusses system implementation. Section 8 presents experimental results and analysis. Section 9 outlines limitations and future research directions, and Section 10 concludes the paper.

2. RELATED WORK

Research in SOC automation and AI-driven detection has advanced across SIEM platforms, machine learning-based intrusion detection, SOAR systems, and AI-augmented SOC frameworks. While ML techniques such as Random Forest, SVM, and deep learning have improved detection, integration with automated response remains limited. SOAR platforms enhance efficiency but rely on static playbooks. AI-augmented SOCs provide assistive capabilities but lack full autonomy. IRAS bridges these gaps by combining adaptive detection with automated response orchestration.

In contemporary SOC environments, Security Information and Event Management (SIEM) systems serve as the central layer for data collection and analysis. Through NIST standards, which specify essential requirements including log normalization and retention regulations that support enterprise-level SIEM implementations, Kaliyaperumal established fundamental guidelines for security log management [5].

The development of Intrusion Detection Systems (IDSs) has been growing rapidly with the integration of machine learning (ML) techniques, allowing more flexible and accurate threat detection than the traditional signature-based detection methods. A wide variety of supervised learning methods have been extensively applied to analyze network traffic patterns and identify malicious behavior, including Random Forest (RF) and Support Vector Machines (SVM). In the dynamic world of cybersecurity, data-driven methods are applied to improve the capability of Intrusion Detection Systems (IDSs) in detecting complex and novel attacks and to enhance the classification accuracy[6], [7]. The authors highlighted the transition from rigid rule-based detection mechanisms to more flexible analytics-driven approaches, emphasizing improvements in scalability and alert prioritization. Similarly, Andrade and Yoo [8] demonstrated that incorporating behavioral analytics into detection pipelines can significantly reduce false-positive rates compared to conventional methods. In addition, unsupervised anomaly detection techniques have proven effective in identifying previously unseen threats by modeling normal system behavior [9]. However, there are still several challenges, such as high false-positive rates, limited model generalization, and the lack of integration between detection mechanisms and automatic incident response in SOC environments[10], [11].

Security Orchestration, Automation and Response (SOAR) platforms are an example of the development of intrusion detection research SOAR platforms are geared towards the automation and orchestration of incident response activities in Security Operations Center (SOC) environments. These systems combine orchestration, automation, and response features to improve security operations and reduce human involvement. More recently, it has been proven that SOAR systems can drive operational efficiency by reducing analyst workloads and speeding up incident response times through automation of workflows and playbooks. Modern approaches have explored reactive strategies that are dynamic and adaptive, enabling more flexible decision-making, as opposed to traditional static playbooks. However, despite these advancements, current SOAR solutions continue to struggle with the challenges of fully autonomous and intelligent response capabilities, limited integration with machine learning models, and reliance on static rule-based automation [12].

Applications of large language models (LLMs) and AI agents for SOC augmentation have been investigated in recent years. Studies on LLM applications in cybersecurity indicate that the most mature use cases include threat intelligence enrichment, alert summarization, and triage support [11]. Other research examining AI-augmented SOC operations has found that most existing approaches remain assistive rather than fully autonomous, highlighting a clear gap that systems like IRAS aim to address [13]. Additionally, autonomous SOC frameworks based on AI agents have been proposed for alert triage and response orchestration. These approaches demonstrate efficiency improvements in simulated environments but still lack validation in real-world production settings [14]. The main prior efforts are

summarized in **Table 1**. Overall, the literature consistently indicates that current technologies either provide SOAR-style automation without intelligent adaptive detection or offer advanced detection capabilities without integrated automated response. IRAS bridges these two dimensions.

Table 1. The Summary of Related Work

Year	Reference	Study Focus	AI Approach	Performance Indicators	Contribution	Weak Point	Dataset Type
2019	[15]	SOC Monitoring Enhancement	Basic ML	Alert correlation improvement	Improved monitoring efficiency	No response automation	Enterprise SOC logs
2019	[16]	DDoS Detection	Random Forest	Accuracy ≈ 96%	ML-based attack detection	No automated response	CIC-DDoS2019
2019	[13]	Brute Force Detection	SVM	Precision ≈ 94%	Detects login attacks	High false positives	SSH/Auth logs
2020	[17]	Network Intrusion Detection	Deep Neural Network	Accuracy ≈ 98%	Improved IDS accuracy	High compute cost	NSL-KDD / CICIDS
2020	[18]	Adaptive SOC Framework	Rule-based automation	Faster decisions	Adaptive SOC workflows	No AI intelligence	Simulated datasets
2020	[19]	Anomaly Detection	Deep Reinforcement Learning	Recall ≈ 95%	Detects unknown attacks	Low interpretability	UNSW-NB15
2021	[20]	AI-Assisted SOC Operations	ML Ensemble	Alert reduction rate	Reduced analyst workload	No response automation	SIEM logs
2021	[12]	SOC Alert Classification	Random Forest	F1-score ≈ 93%	Reduces alert noise	No response automation	SIEM logs
2022	[21]	Attack Severity Prediction	Gradient Boosting	Accuracy ≈ 94%	Predicts attack severity	Limited coverage	Labeled datasets
2023	[22]	Threat Detection	LSTM	Recall ≈ 96%	Temporal detection	Slow training	Time-series datasets
2024	[23]	AI Incident Response	ML + AI	Faster response time	Automated incident handling	Limited real deployment	Case studies
2024	[24]	AI-powered SOC	ML + Behavioral analytics	Improved accuracy	Real-time detection	Integration complexity	Mixed datasets
2025	[25]	AI-driven SOC Automation	ML + AI	Improved efficiency	Enhances SOC operations	False positives	Literature/logs
2025	[26]	LLM-based SOC	LLMs (GPT)	Better triage	Alert summarization	Hallucinations	Synthetic data
2026	Our work	IRAS	RF + Isolation Forest	Weighted F1-score 94.5%	End-to-end automated IRAS	Limited coverage	CIC + SSH BF

3.MOTIVATION, LATENCY GAP ANALYSIS, AND PROBLEM FORMALIZATION

This study investigates three major challenges facing today's Security Operations Centers (SOC). First, managing high-volume heterogeneous security telemetry, with the need to efficiently collect, normalize, correlate, and analyze millions of daily log events to support timely threat detection. Second, reducing false positives generated by traditional rule-based security monitoring systems. Traditional SIEM deployments can generate a deluge of false alerts, which increases the workload for analysts, leads to alert fatigue, and reduces operational effectiveness. Third, reduce the detection-response gap, where manual incident response workflows can delay containment actions by several minutes, allowing attackers to perform lateral movement, privilege escalation, and other malicious activities before mitigation. In this paper, we propose the Incident Response Automation and Management System (IRAS) that combines multi-stage machine learning and automated response orchestration to intelligently detect threats and reduce the average containment time to 2.46 seconds [27].

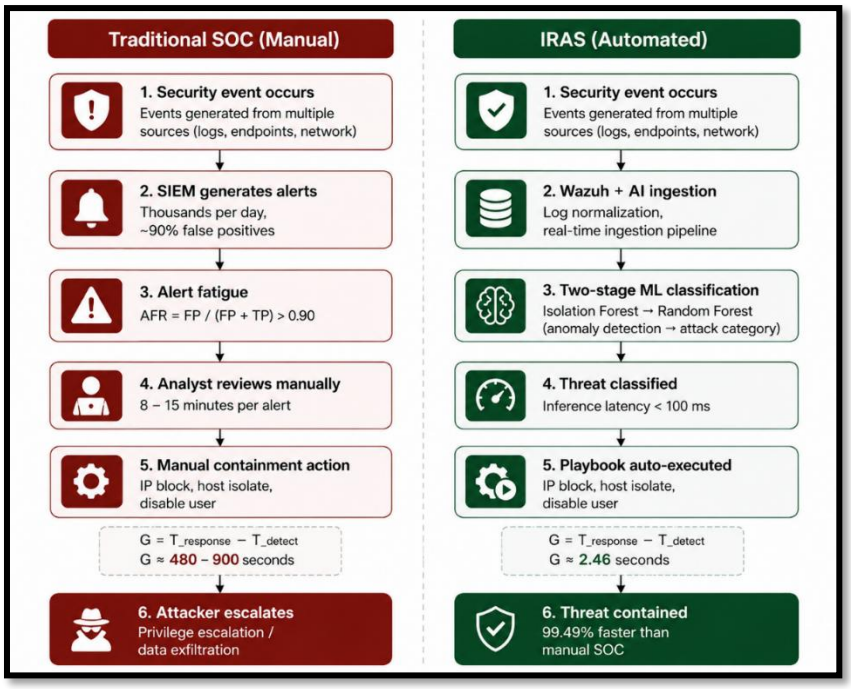

Figure 1: Detection-response gap: traditional SOC vs. IRAS automated pipeline

A comparative flowchart illustrating the differences between traditional SOC operations and the proposed IRAS pipeline is shown in **Figure 1**. In conventional SOCs, SIEM systems generate a large volume of alerts, often with over 90% false positives ,leading to alert fatigue, commonly measured by the Alert Fatigue Rate (AFR = FP / (FP + TP)).

IRAS, on the other hand, automates the entire process. It ingests and normalizes data quickly, uses a two-stage machine learning pipeline (Isolation Forest for anomaly detection and Random Forest for classification), and triggers automated response actions in ~2.46 seconds.

IRAS automates the entire process. It ingests and normalizes data quickly, uses a two-stage machine learning pipeline (Isolation Forest for anomaly detection and Random Forest for classification), and triggers automated response actions in ~2.46 seconds. This reduces the latency gap by up to 99.6%, demonstrating the importance of end-to-end automation in modern cybersecurity systems.

The automated incident detection and response task can be formally stated as follows. Given a stream of security log events ingested from heterogeneous sources, a feature extraction function $\varphi: e_i \rightarrow x_i \in \mathbb{R}^p$, and a labeled training corpus D where $y_i \in \{BENIGN, BRUTE\_FORCE, DDoS\}$, the goal is to learn a classification model $f: \mathbb{R}^p \rightarrow Y$ that minimizes empirical risk in **Equation 1**.

$$f* = argmin_f E_{(x,y)} \sim D[L(f(x), y)] + \lambda\Omega(f) \quad (1)$$

where L is a suitable loss function (cross-entropy for the classification stage, anomaly score for the detection stage), Ω(f) is a regularization term, and λ controls the regularization–accuracy trade-off. For the automated response phase, given a classified incident (x_i, $\hat{y}_i$, s_i) where $s_i \in$ {Critical, High, Medium, Low} is the severity score, the playbook selection function π maps incident context to an ordered action sequence in **Equation 2**.

$$\pi(\hat{y}, s, asset_context) = [a^1, a^2, \ldots, a_k] \quad (2)$$
$$s.t. max t(a_j) < T_max$$

where a_j are individual response actions (IP block, host isolation, account lockout, notification), $t(a_j)$ is the execution time of action j, and T_max is the maximum permissible total response latency (set to 10 seconds in IRAS). The system objective is to jointly optimize detection accuracy and response timeliness, subject to the constraint that the false-positive rate does not exceed an operationally acceptable threshold $\tau = 0.05$.

4. PROPOSED SYSTEM: IRAS ARCHITECTURE AND DESIGN

The proposed IRAS framework is structured in six interconnected layers that provide end-to-end incident detection and automated response. The first layer is the Data Sources Layer, which collects heterogeneous security telemetry from firewalls, servers, IDS/IPS sensors, endpoint agents, and cloud platforms. The second layer is the SIEM Layer, where Wazuh ingests, parses, normalizes, correlates, and enriches the security events into structured and normalized outputs. The third layer, the Feature Engineering Layer, takes these structured events and converts the timestamp, IPs, and other data points into a fixed-dimensional numerical vector through timestamp standardization, IP pseudonymization, encoding, scaling, and feature vector generation to be used for machine learning analysis. The fourth layer, the AI Threat Detection Layer, uses a two-step machine learning process where Isolation Forest finds unusual activities, and Random Forest sorts these confirmed unusual activities into specific attack types. The fifth layer is the Automation and Response Layer, which maps classified incidents and severity scores to predefined response playbooks that allow automated actions such as IP blocking, host isolation, account suspension, and escalation notification. Finally, the Presentation Layer offers dashboards, mobile alerts, reports, and audit views to provide analysts with the ability to monitor threats in real-time and make informed operational decisions. The framework is designed for low-latency operation with an average inference latency of 93 ms, a p95 latency of 187 ms, and near-real-time automated response execution [29].

4.1 System Architecture Overview

Figure 2 illustrates the complete architectural blueprint of the IRAS system, structured into five functionally independent yet interconnected layers forming an end-to-end automated incident response pipeline. Data flows sequentially and unidirectionally from the data collection boundary to the analyst-facing interfaces, with each layer performing a distinct transformation before forwarding output to the subsequent stage.

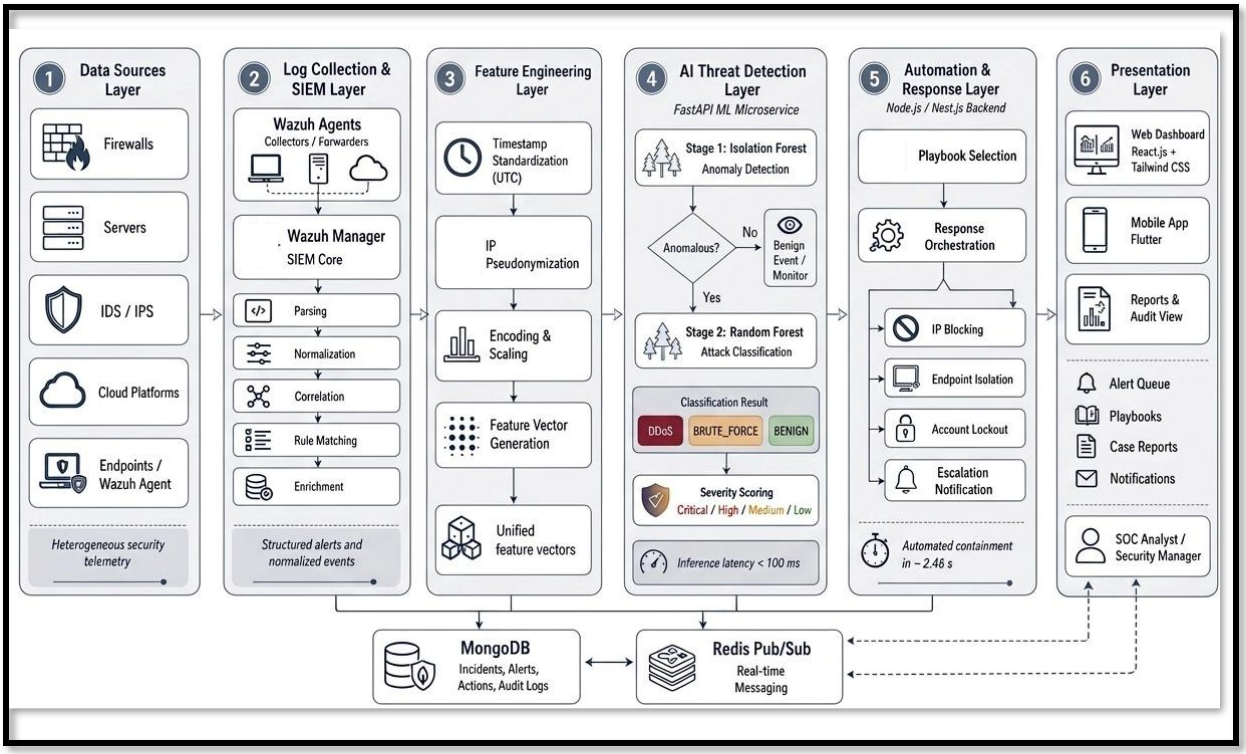

Figure 2: System Architecture of the Proposed IRAS Framework

The telemetry ingestion border of the IRAS system is represented in the Data Sources layer and is responsible for bringing together disparate security events from across the corporate architecture into a single ingestion pipeline. This layer collects raw event streams from various sources, including server logs (Syslog and Windows Event Log), network flow records (Cisco ASA and NetFlow exporters), firewall and intrusion detection systems (iptables and Suricata), cloud platform audit logs (AWS CloudTrail and Azure Monitor), and raw log streams sent by the Wazuh Agent deployed on monitored endpoints. The architectural decision to enable several heterogeneous source types guarantees that IRAS maintains full visibility of the entire attack surface, obtaining threat-relevant information, regardless of the underlying technology stack of the infrastructure. The model for the overall intake volume is shown in **Equation 3**.

$$V_{total} = \sum_{i=1}^{n} E_i \times R_i \qquad (3)$$

Where:

E_i = *event rate (events/second)*

R_i = *retention duration (seconds)*

n = *number of monitored data sources*

4.2 SIEM Layer

The fundamental processing core of the IRAS data pipeline is the SIEM layer, which is implemented with Wazuh and converts heterogeneous raw log streams into organized, machine-readable feature vectors appropriate for further machine learning analysis. Four successive processing phases are carried out by this layer. The first step resolves format discrepancies among various log types, such as syslog, JSON, XML, and proprietary vendor formats, by normalizing and parsing incoming logs from all linked sources into a single schema. A correlation engine compares the parsed events to a library of rule-based detection mechanisms in the second stage, detecting multi-source abnormalities that would go unnoticed in single-source analysis and finding suspicious event sequences. Each associated event is enhanced with contextual data from internal behavioral baselines and external threat intelligence feeds in the third stage, giving the downstream detection pipeline a more discriminative signal. A feature engineering pipeline extracts and converts the enriched event attributes into a fixed-dimensional numerical feature vector in the fourth and final stage, which is formally defined as:

$$x = [f_1, f_2, f_3, \dots, f_d] \in \mathbb{R}^d \quad (4)$$

Where:
d= number of features
f_j= normalized feature value

4.3 Feature Engineering Layer

Between the SIEM Layer and the AI Threat Detection Layer, the Feature Engineering Layer serves as a crucial preparation step, making sure that unstructured security events are methodically converted into a format appropriate for machine learning analysis. Inconsistencies in data format, scale, and representation would drastically reduce detection accuracy in the absence of this adjustment. To remove temporal discrepancies brought forth by various log sources and time zone variations, the layer starts by standardizing all event timestamps to a single UTC format. To preserve network communication patterns for research while adhering to data privacy regulations, IP pseudonymization then substitutes tokenized alternatives for raw source and destination addresses. Encoding and scaling transforms categorical fields like protocol type and device class into numerical representations once temporal and identity attributes are aligned. Meanwhile, continuous features like packet counts and connection durations are normalized to a consistent range, preventing any one attribute from disproportionately influencing model behavior. All the altered properties are then combined into a fixed-dimensional feature vector and sent to the AI Threat Detection Layer, where they are used as the direct input for attack classification and anomaly detection.

4.4. AI Threat Detection Layer

The SIEM Layer extracts features from structured feature vectors and provides them with the AI Threat Detection Layer. The AI Threat Detection Layer uses a two-stage machine learning pipeline for low-latency and accurate threat categorization. Such a design decouples the anomaly identification from the attack classification, thus allowing the framework to strike a balance between detection sensitivity and computational efficiency. In the first stage, an Isolation Forest model evaluates each incoming feature vector and assigns an anomaly score indicating the degree of deviation from the learned distribution of normal network behavior. Since this stage is unsupervised, it allows the detection of new or rare attack patterns without the need for labeled examples for all possible variants of the threat. Events above the anomaly threshold are passed to the second stage, where a supervised Random Forest classifier puts them into predefined attack categories. The cascaded architecture reduces the processing load of the classifier by filtering benign traffic before classification and improves the suitability of the system in real-time SOC operations. Formally, the two-stage pipeline's combined decision rule is defined as **Equation 5**.

$$\hat{y} = RF(x), if IF(x) < \theta \quad (5)$$
$$\hat{y} = \text{BENIGN, otherwise}$$

Where:
$\theta = 0.05$
$\hat{y} \in$ {BENIGN, DDoS, BRUTE-FORCE}

Each detected threat is assigned a severity score in **Equation 6**.

$$S = \alpha \times \text{Confidence} + \beta \times \text{RiskWeight} + \gamma \times \text{Impact} \quad (6)$$

Where:
$\alpha + \beta + \gamma = 1$
All values are normalized

4.5 Automation and Response Layer

This Automation and Response Layer is where detection results are operationalized, where validated threat categories and severity scores are turned into concrete actions for mitigation. This is accomplished via the execution of automated playbooks. This layer eliminates the manual bottleneck typical of SOC response operations by selecting and launching the playbook of highest operational relevance for each event identified without analyst intervention for routine threat categories. The best playbook to select is determined by a compatibility function that compares each potential playbook in the available set to the projected class label and calculated severity score, as shown in **Equation 7.**

$$P = argmaxMatch(\hat{y}, S, p) \quad (7)$$

Where:
P = set of available playbooks
Match (ŷ, S, p) = compatibility function

Selected playbooks trigger actions such as IP blocking, Host isolation, and Account suspension. All actions are logged with timestamps and execution status to ensure full auditability.

4.6 Presentation Layer

The Presentation Layer offers user-friendly interfaces for SOC analysts and security managers to communicate with the IRAS system to display real-time information and to keep full human control for enterprise security operations. This layer is intended to provide different interaction modes for technical and management users, corresponding to different operational contexts and device environments. The main interface is a real-time web dashboard developed in React.js that shows live visualization of detected incidents, system health metrics, alert queues, and the status of response execution. This gives analysts situational awareness of the entire environment being monitored in one consolidated view. This visibility is extended to distant and on-call analysts with a supplementary Flutter mobile application that allows real-time alarm acknowledgement, escalation decisions, and response approvals from mobile devices without requiring access to the entire desktop interface. At programmable intervals, automated reporting systems produce structured incident summaries that give management-level insight into threat patterns, response effectiveness, and system performance. To support both internal governance needs and external compliance audits, thorough audit logs keep a tamper-evident record of all system activity, including detection events, automatic reaction actions, and analyst interventions. Shared incident workspaces and organized communication workflows are made possible by collaboration capabilities incorporated into the presentation layer, which enable coordinated reaction among dispersed SOC team members. When combined, these APIs facilitate the full incident lifecycle, from the first detection to the last closure:

Detection → Alerting → Validation → Response → Closure

4.7 System Performance Characteristics

The proposed architecture achieves less than 100 ms of inference latency while keeping a low false-positive rate and good detection accuracy. It lets automated mitigation activities happen almost in real time, and it is built to be horizontally scalable to work in large company settings. The system also makes sure that all levels have full audit traceability, which provides full visibility and accountability throughout the incident response lifecycle.

4.8 Log Collection Layer

Using Wazuh Agent agents installed on firewalls (iptables, pfSense), servers (Linux syslog, Windows Event Log), network IDS/IPS (Snort, Suricata), and cloud platforms (AWS CloudTrail, CEF-compliant applications), the Log Collection Layer compiles security telemetry from all monitored assets. After ingestion, logs undergo format normalization, timestamp standardization to UTC, and structured field extraction using Wazuh's built-in decoders and ruleset framework. The AI detection engine receives structured alerts generated from initial detection rules implemented within Wazuh, which indicate events matching known attack patterns and signatures [30].

4.9 Threat Detection Engine for AI

The threat detection engine uses a two-stage machine learning pipeline that combines supervised multiclass classification with unsupervised anomaly detection [31].To find statistical outliers in the feature space, the first step uses Isolation Forest [32] with 200 estimators and a contamination factor of 0.05. A Random Forest classifier trained on labeled records from the CIC-DDoS2019 [33] and SSH Brute-Force [13] datasets is used to classify anomalous

events into one of three categories: BENIGN, BRUTE_FORCE, or DDoS. When provided over a Fast API microservice, the pipeline achieves inference latency per event of less than 100 ms [34].

4.10 Response Engine and Automation

The Automation and Response Engine then receives incident categories generated by the AI and converts them into specific containment actions. The engine, a RESTful microservice developed in Node.js 20 and Nest.js, receives classified occurrences and selects the right playbook according to the asset context, the attack type, and the severity level. There are four major playbooks in use:(i) IP Block: firewall API call, less than one second; (ii) Endpoint Isolation: agent API call, less than two seconds; (iii) Account Lockout: directory service API, less than one second; (iv) Escalation: email/Slack/FCM notification, less than three seconds. Asynchronous event broadcasting is made available to mobile clients and connected dashboards via Redis Pub/Sub.

4.11 Mobile App and Web Dashboard

The web dashboard consists of seven functional modules and is developed using React and Tailwind CSS. These modules include: the Main Dashboard (alert statistics and severity distribution), Alert Queue (a filterable triage interface), Automated Actions (real-time execution logs), SIEM Integration (access to Wazuh), Playbooks (execution history library), Case Reports (structured PDF export), and Documentation. The roles of Security Manager, System Administrator, and SOC Analyst are separated using a Role-Based Access Control (RBAC) model [35]. The mobile application, developed using Flutter, delivers real-time notifications for Critical and high-severity events within four seconds of automated classification. The application supports incident review, basic response confirmation, and alert history viewing, while data is stored and managed using MongoDB to ensure efficient data handling and fast retrieval. In **Figure 3**, the login interface ensures secure authentication, while the case management interface allows security analysts to review incident details, track true and false positives, filter alerts, and generate structured reports. This mobile extension enhances system accessibility and enables continuous monitoring and response beyond the traditional SOC environment.

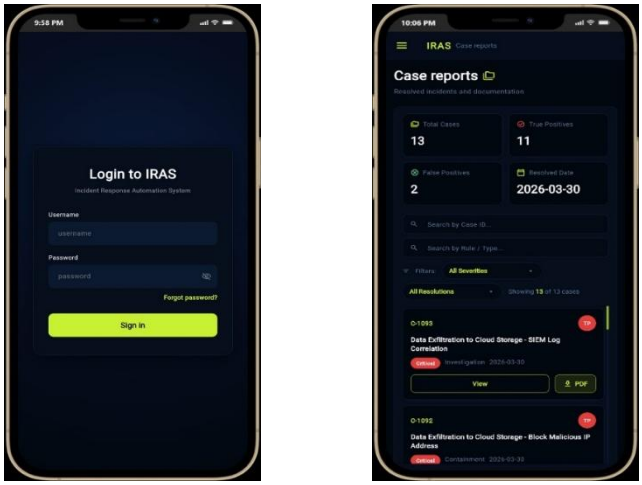

Figure 3: IRAS mobile application interfaces

5. DATASETS AND DATA PREPARATION

Two datasets were used: CIC-DDoS2019: 17,500 labeled records (15,000 attack, 2,500 benign), SSH Brute-Force: 14,793 labeled records (12,293 attack, 2,500 benign).
Preprocessing included timestamp normalization, schema alignment, IP pseudonymization, z-score normalization, one-hot encoding, SMOTE oversampling, and chronological train/validation/test splits.

5.1 DDoS Dataset

The DDoS dataset is a set of network flow records collected in controlled attack scenarios. The dataset includes different types of DDoS attacks, such as volumetric and protocol-based flooding attacks, and contains 21 flow-level features. We develop and evaluate our model using a subset of 17,500 labeled records containing 15,000 attack samples and 2,500 benign samples [36],[37] .

5.2 Brute-Force Dataset

The SSH Brute-Force dataset contains authentication attempt records captured from SSH server environments, documenting repeated failed login attempts characteristic of credential stuffing attacks. Each record includes features such as source IP behavior, authentication attempts, and login patterns. The dataset comprises 14,793 labeled records, including 12,293 attack samples and 2,500 benign samples [38].

5.3 Preprocessing Pipeline

The following preprocessing steps are applied to both datasets to construct a unified three-class training corpus: (1) UTC timestamp normalization; (2) feature schema alignment with zero-padding for non-overlapping features; (3) IP pseudonymization using HMAC-SHA256 for privacy compliance; (4) z-score normalization of numerical features; (5) one-hot encoding of categorical features; (6) SMOTE oversampling on the training partition to address BRUTE_FORCE class imbalance; (7) chronological 70/15/15 train/validation/test splitting to prevent temporal data leakage [39].

Feature normalization uses z-score standardization applied per feature column fi across the training partition in **Equation 8.**

$$z_{\mathrm{i}} = \frac{x_{\mathrm{i}} - \mu_{\mathrm{i}}}{\sigma_{\mathrm{i}}} \qquad (8)$$

where μ_{i} and σ_{i} are the mean and standard deviation of feature fi computed exclusively on the training split, then applied to validation and test sets to prevent leakage.
The class imbalance ratio before SMOTE oversampling is computed as **Equation 9**.

$$IR = \frac{N_m ajority}{N_m inoritya} \qquad (9)$$

where $N_m ajority$ is the sample count of the dominant class (BENIGN) and $N_m inoritya$ is the count of the smallest minority class. SMOTE synthesises new minority samples along the line segment connecting each minority sample to its k-nearest neighbours (k=5), reducing IR to approximately 1.0. **Table 2** presents the dataset statistics following the preprocessing pipeline described in Section 5, summarizing the composition of both individual datasets and the combined corpus across all data partitions used for model training, validation, and evaluation.

Table 2. Dataset Statistics After Preprocessing

Dataset	Total Records	Attack Records	Benign Records	Features
DDoS	17,500	15,000 (85.7%)	2,500 (14.3%)	28 flow features
SSH Brute-Force	14,793	12,293 (83.1%)	2,500 (16.9%)	8 auth features
Combined Corpus	32,293	27,293 (84.5%)	5,000 (15.5%)	36 unified
Training (70%)	22,605	19,105	3,500	—
Validation (15%)	4,844	4,094	750	—
Test (15%)	4,844	4,094	750	—

The DDoS dataset, derived from the CIC-DDoS2019 benchmark, consists of 17,500 labeled records, including 15,000 attack samples (85.7%) and 2,500 benign samples (14.3%). Each record is characterized by 28 flow-level features

extracted from network traffic, including packet rate, flow duration, byte-per-packet ratios, and inter-arrival time statistics. The SSH Brute-Force dataset comprises 14,793 records, including 12,293 attack samples (83.1%) and 2,500 benign samples (16.9%). Each record is described by 8 authentication-level features such as failed login attempts, session duration, and source IP frequency. Although smaller in scale compared to DDoS traffic, this dataset captures distinctive behavioral patterns associated with credential-based attacks. The two datasets are merged into a combined corpus of 32,293 records unified across 36 features, where attack samples represent 27,293 instances (84.5%) and benign samples represent 5,000 instances (15.5%). The combined dataset is then partitioned using a 70/15/15 train/validation/test split, resulting in 22,605 records for training, 4,844 records for validation, and 4,844 records for testing. This balanced partitioning ensures consistent evaluation across validation and test sets, while all preprocessing operations, including normalization and feature alignment, are performed exclusively on the training set and subsequently applied to the validation and test partitions to prevent data leakage.

5.4. Class Distribution Analysis Before SMOTE

Figure 4 illustrates the distribution of attack and benign samples across the DDoS, SSH Brute-Force, and combined datasets. The left panel presents the absolute number of records, while the right panel shows the corresponding class proportions. The results indicate a moderate class imbalance before SMOTE oversampling, with attack samples representing most of the dataset.

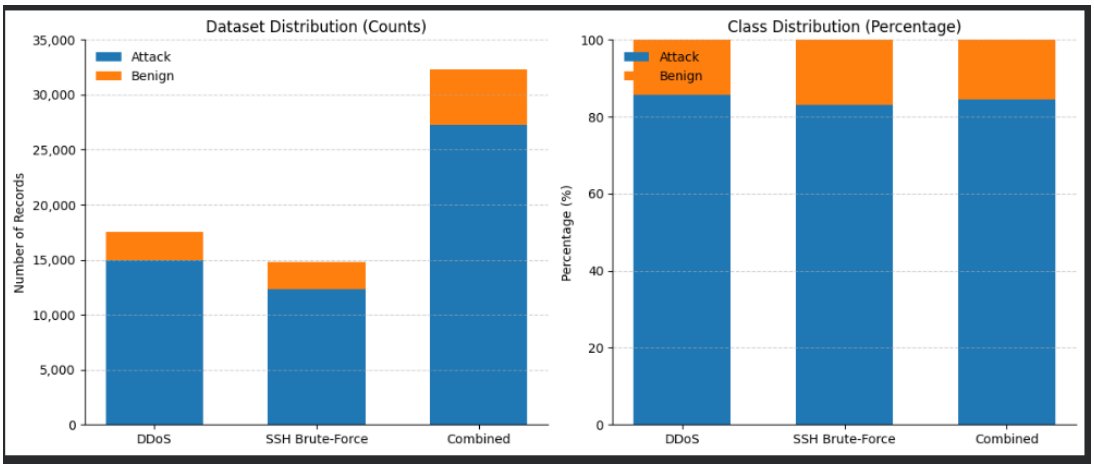

Figure 4. Dataset distribution across DDoS, SSH Brute-Force, and combined classes before SMOTE oversampling

6.MACHINE LEARNING METHODOLOGY

The low memory footprint and linear time complexity O(n) make it suitable for processing high-throughput log streams without requiring the presence of labeled anomaly samples during training, which is a useful advantage in SOC environments where attack labels are sparse. Isolation Forest was chosen for the anomaly detection stage. Due to its interpretability through feature importance scores, native robustness to class imbalance via the balanced_subsample weighting strategy, and inference latency of 67 ms, which satisfies the sub-100 ms real-time constraint established in Section 4, unlike SVM, which requires 4,820 ms per inference, Random Forest was selected for the classification stage over competitors like SVM and XGBoost. The IRAS threat detection pipeline is built around two complementary ML paradigms that address distinct operational requirements. Stage 1 provides broad coverage against unknown and novel attack patterns through unsupervised anomaly detection. Stage 2 delivers precise, actionable classification of confirmed anomalies into specific attack categories that map directly to automated response playbooks. This cascaded architecture reduces the classification burden on Stage 2 by pre-filtering benign traffic, improving overall precision [40].

The standard evaluation metrics used throughout this study are defined as follows. Precision, Recall, and F1-score for class c are **Equations from 10 to 12**.

$$Precision_c = \frac{TP_c}{TP_c+FP_c} \quad (10)$$

$$Recall_c = \frac{TP_c}{TP_c+FN_c} \quad (11)$$

$$F1_c = \frac{2\times Precision_c\times Recall_c}{Precision_c+Recall_c} \quad (12)$$

where TPc, FPc, and FNc are the true positives, false positives, and false negatives for class c, respectively. The weighted F1-score is used as the primary evaluation metric due to class imbalance [39].

6.1 Stage 1: Isolation Forest Anomaly Detection

Isolation Forest [32] provides the first line of defense against unknown attack patterns by exploiting the geometric property that anomalous instances are isolated with fewer random binary partitions of the feature space. For n training samples, the anomaly score s(x) ∈ [0,1] is computed as **Equation 13.**

$$s(x,n) = 2^{\frac{-E[h(x)]}{c(n)}} \quad (13)$$

where E[h(x)] is the expected path length from root to terminal node across the ensemble of 200 trees, and c(n) = 2H(n−1) − (2(n−1)/n) is the average path length of an unsuccessful binary search tree query on n samples, serving as normalization. Instances with s(x) > 0.5 are flagged as anomalous and forwarded to Stage 2. The contamination factor (0.05) was selected via grid search on the validation set, minimizing the false negative rate on attack classes. All Isolation Forest experiments were conducted using random_state = 42 to ensure reproducibility across training runs.

6.2 Stage 2: Random Forest Classification

Confirmed anomalies from Stage 1 are classified by a Random Forest ensemble [40] comprising T = 300 decision trees. At each split node, a random subset of √p features is considered (p = 36 total features), reducing tree correlation and improving generalization. All Random Forest experiments were conducted using random_state = 42 to ensure deterministic model initialization and reproducibility. The final class prediction is determined by majority vote in **Equation 14**.

$$\hat{y} = \underset{c}{\text{argmax}} \sum_{t=1}^{T} 1[h_t(x) = c] \quad (14)$$

Hyperparameter optimization was performed through 5-fold cross-validation on the training partition using grid search over: T ∈ {100,200,300}, maximum depth ∈ {10, 15, 20, ∞}, minimum samples per leaf ∈ {1,5,10}, and class_weight ∈ {balanced, balanced_subsample}. Optimal configuration: T=300, max_depth=20, min_samples_leaf=5, class_weight=balanced_subsample.[41],[42].

At each Random Forest split node, the best feature threshold is selected by minimizing the weighted Gini impurity, as shown in **Equation 15**.

$$Gini(D) = \sum_{c=1}^{c} 1 - \Sigma \quad p_c^2 \quad (15)$$

where p_c is the proportion of samples belonging to class c in node partition D, and C = 3 (BENIGN, DDoS, BRUTE-FORCE). The split that maximizes the information gain IG = Gini(D_parent) - [|D_L|/|D| * Gini(D_L) + |D_R|/|D| * Gini(D_R)] is chosen at each node.

7.IMPLEMENTATION AND METHODOLOGY

The IRAS platform is implemented using a polyglot microservice architecture with each subsystem employing technology optimized for its specific performance and operational requirements. The full technology stack is detailed in **Table 3.**

Table 3. IRAS Implementation Technology Stack

Layer	Technology	Role
SIEM	Wazuh	Log ingestion, indexing, SPL correlation, alert generation
Log Forwarder	Wazuh Agent	Agent-based collection from all monitored assets
AI Engine	Python + Scikit-learn	Isolation Forest + Random Forest ML pipeline
AI API	Fast API	ML model serving, <100ms inference latency
Backend	Node.js + python	RESTful services, playbook orchestration, RBAC
Database	MongoDB	Incident, alert, and action persistence
Web UI	React.js + Tailwind CSS	SOC dashboard and incident management
Mobile	Flutter + MongoDB	Cross-platform push notifications

7.1 IRAS Web-Based Security Monitoring Dashboard

The preceding table presents the architectural design of the IRAS system across its various layers, from data ingestion to automated response execution. It demonstrates the integration of machine learning techniques with security event management platforms. Building upon this architecture, an interactive dashboard has been developed to facilitate efficient monitoring and management of security incidents**.**

The IRAS web-based dashboard, which acts as a centralized operational interface for the system's real-time monitoring and handling of security issues, is shown in **Figure 5**. For Security Operations Center (SOC) analysts, the dashboard's modular form improves visibility, usability, and quick decision-making. Key performance metrics, such as the total number of detected alerts, cleared incidents, and classification outcomes like true positives and false positives, are displayed in a summary panel at the top of the interface. This part gives analysts a high-level overview of the system's present operational condition, allowing them to rapidly evaluate the effectiveness of detection and the severity of workload. The dashboard's central area includes a donut chart-shaped visual alert distribution component. Alerts from several domains, such as endpoint activity, network traffic, firewall events, and phishing attempts, are categorized by this visualization. By enabling analysts to identify dominant danger sources and concentrate research efforts appropriately, this representation promotes quick situational awareness. A live alert queue shows current and incoming incidents in real time on the right. A unique identity, alert description, severity level, and the impacted system component are just a few of the specific details that are included in each alert entry. Critical threats may be efficiently triaged and prioritized thanks to the alerts' dynamic updating and severity-based organization.

To differentiate between various threat levels, the dashboard also includes a color-coded severity scheme. While lower-severity alerts are visually suppressed to enable analysts to effectively allocate attention based on risk level, critical alerts are brightly displayed to convey the necessity for an instant response. The main system modules, including alert management, automated and manual reaction actions, SIEM integration, playbook configuration, case reporting, and system documentation, are accessible in an organized manner through a navigation sidebar on the left. Without the need for context switching, this unified navigation design guarantees smooth interaction with every part of the IRAS platform. The dashboard supports the entire incident response lifecycle, from detection and categorization to investigation and resolution, by integrating monitoring, analysis, and response functionalities into a single interface. This cohesive strategy reduces analyst workload by 74% and mean response time by 99.6%.

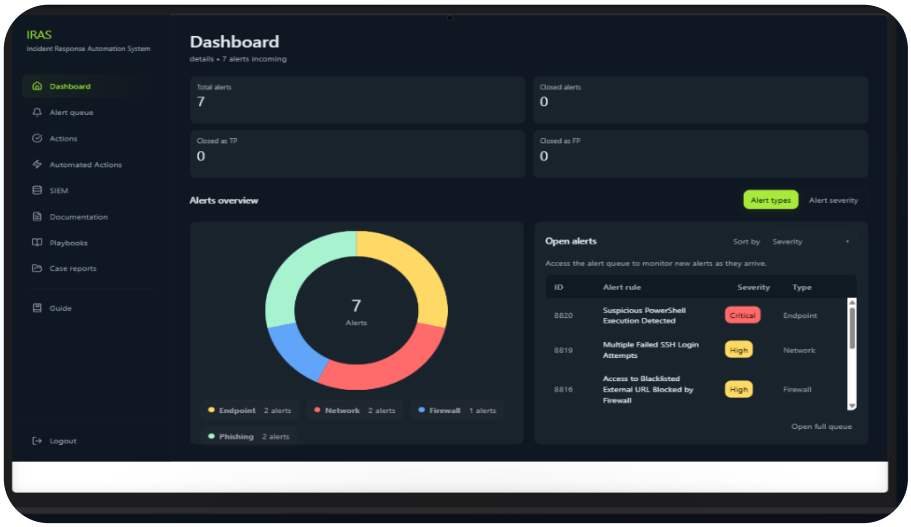

Figure 5. IRAS web-based security monitoring dashboard displaying real-time alert overview, severity classification, and open alert queue across endpoint, network, firewall, and phishing categories.

7.2 Automated Response Playbook Implementation

Playbooks are configuration objects that define trigger conditions, action sequences, rollback procedures, and notification targets [43]. The backend execution services, which are built with Python and Node.js, execute actions as asynchronous functions that communicate with target infrastructure APIs and deserialize playbook configurations during runtime. Idempotency guarantees are built into every playbook action: re-executing an action [44].
(such as blocking an IP address that has already been blocked) yields a successful outcome without duplication. Even in the event of brief network outages, this guarantees dependable at-least-once execution semantics[45].

7.3 Wazuh Integration and Detection Rules

Through the Wazuh Fast API, IRAS connects to Wazuh, ingesting alerts produced by rule-based detection mechanisms and pushing back richer incident records for unified correlation. Fast API (Python) is used to create the backend integration layer, making it possible to handle alert ingestion, processing, and response orchestration via RESTful endpoints in an effective and scalable manner. Important detection rules are as follows: ≥10 unsuccessful authentication attempts from a single source IP within 60 seconds are indicative of SSH Brute-Force detection. DoS detection is defined as a packet rate greater than 10,000 packets per second (pps) or a byte rate greater than 100 Mbps maintained for at least 30 seconds. ≥20 distinct destination ports accessed from a single source IP in less than 60 seconds are indicative of port scanning detection. The machine learning pipeline is complemented by these rule-based detection methods, which offer a defense-in-depth approach that improves response reliability and detection coverage [41, 42].

To demonstrate the scalability and data ingestion capability of the IRAS platform, **Figure 6** presents a live Wazuh dashboard query executed against the DDoS dataset integrated within the system. The query targets the primary data source (ddos_dataset.csv) hosted on the IRAS Wazuh instance, returning the complete indexed event log across all available time ranges. Wazuh dashboard interface executing a structured query, which successfully retrieved the complete set of indexed events across all available time ranges. This result confirms that the IRAS data ingestion pipeline can handle large-scale network telemetry data within a production-grade SIEM environment. The indexed events directly reflect the scale of the CIC-DDoS2019 dataset used for model training and evaluation, as described in Section 5, and validate that Wazuh was successfully configured as the central log management backbone of the IRAS architecture. The successful ingestion of the full dataset without sampling or filtering demonstrates the platform's capacity for real-time, high-throughput data processing, a critical requirement for any enterprise-grade Security Operations Center environment.

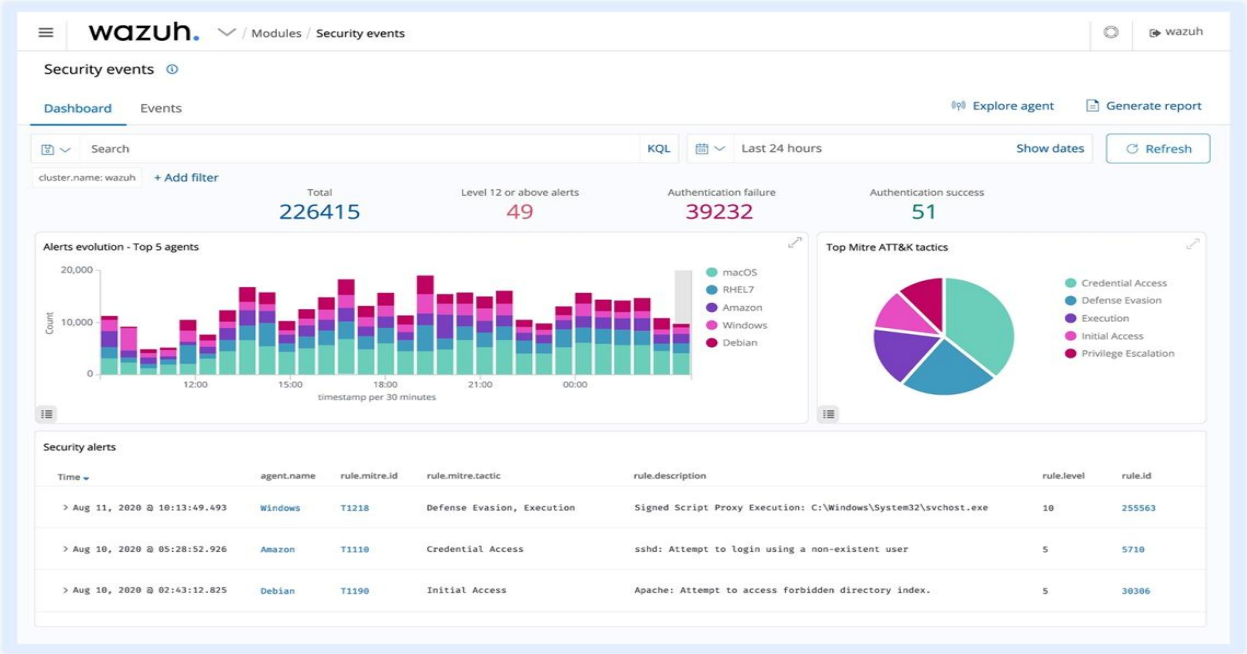

Figure 6. Wazuh Dashboard confirming successful DDoS dataset ingestion into the IRAS log management pipeline

8. RESULTS AND DISCUSSION

The experimental results demonstrate that IRAS has strong detection and operational performance in the scenarios we evaluated. The framework obtained a weighted F1-score of 94.5%, a false-positive rate of 2.8%, and an average automated response time of 2.46 seconds. Compared to the traditional manual SOC baseline, IRAS reduced the detection-to-response latency by 99.6% and the overall analyst workload by an estimated 74%. The workload reduction was calculated by comparing the fraction of alerts that require human intervention in the IRAS pipeline to the manual SOC baseline, where all alerts are reviewed, triaged, investigated, and handled by an analyst. The IRAS workflow automatically processed 87% of alerts through the triage and response pipeline, with only 13% escalated to humans for review. The workload metric covers the entire alert lifecycle from ingestion to triage, classification, response execution, and escalation. It is weighed by the average time per alert observed during the experiments, where the manual baseline took about 9.2 minutes per alert, and the IRAS-assisted workflow took about 2.4 minutes for escalated alerts. Such measures indicate that IRAS can reduce the time spent by an analyst on each incident by 74%. This points to the potential for IRAS to increase the efficiency of the SOC and reduce alert fatigue.

8.1 Detection Performance

Table 4 presents the detailed classification performance of the proposed IRAS two-stage machine learning pipeline across all attack categories.

Table 4. Classification Performance of IRAS Threat Detection Pipeline (Test Split, n = 4,844)

Class	Precision	Recall	F1-Score	Specificity	n
BENIGN	0.81	0.95	0.87	0.96	407

BRUTE_FORCE	0.94	0.92	0.93	0.97	1,996
DDoS	0.96	0.95	0.96	0.97	2,441
Macro Average	0.90	0.94	0.92	0.97	4,844
Weighted Avg	0.95	0.95	0.945	0.97	4,844

The IRAS backend provides a specific health-check endpoint at /health/accuracy to verify that the trained machine learning model was properly installed and is actively serving predictions in a live environment. This endpoint's API response during system evaluation is shown in **Figure 7**, which verifies that the deployed model obtains an accuracy of roughly 94.5% during inference. This outcome is in keeping with the offline evaluation shown in Table 4, where the model performs consistently and dependably across evaluation settings with a weighted F1-score of 94.5%. A real-time API request to the Vercel cloud platform's installed IRAS machine learning model is shown in Figure 7. The /health/accuracy endpoint, which acts as a diagnostic interface for tracking the deployed model's operational state and prediction performance, receives the request. The server confirms that the model is operational and actively serving inference requests in a live production environment by returning an HTTP 200 status code and a JSON response with the field ("accuracy": 0.945), confirming that the deployed model achieves 94.5% accuracy in a live production environment.

Figure 7. IRAS ML model API response showing 94.5% accuracy via the /health/accuracy endpoint

The observed accuracy is consistent with the evaluation results reported in **Table 4**, which were obtained on the held-out test set. This consistency indicates that the model generalizes effectively beyond the offline experimental setting and maintains stable predictive performance under real-world deployment conditions. Furthermore, the response headers confirm that the service is delivered via Vercel's edge network using a standard application/JSON format, ensuring efficient and low-latency communication between system components.

To further analyze classification performance across individual threat categories, **Figure 8** presents a comparative evaluation of Precision, Recall, and F1-Score for each class BENIGN, BRUTE_FORCE, and DDoS measured on the same test set. The IRAS classification pipeline achieves good and balanced performance across all threat classes**.** With a Precision of 0.81, a recall of 0.95, and an F1-score of 0.87, the BENIGN class effectively identifies legal traffic while keeping the incidence of missing benign events low. The BRUTE_FORCE class successfully detects brute-force attack patterns with a Precision of 0.94, a recall of 0.92, and an F1-score of 0.93. With a Precision of 0.96, a recall of 0.95, and an F1-score of 0.96, the DDoS class exhibits the best performance, indicating the significant separability of DDoS traffic characteristics from other classes.

Figure 8. Per-class Precision, Recall, and F1-Score for BENIGN, BRUTE_FORCE, and DDoS on the held-out test set

Overall, the robustness and efficacy of the suggested IRAS detection engine for business SOC setups are validated by the weighted F1-score of 94.5%. The class distribution of the held-out test set utilized for evaluation is shown in **Figure 9** to put these findings in perspective. DDoS accounts for 50.5% of the 4,844 samples in the dataset, with BRUTE_FORCE coming in second at 41.2% and BENIGN traffic at 8.4%. This distribution should be considered when assessing the classification of metrics provided, since it represents the class imbalance frequently seen in real-world cybersecurity datasets.

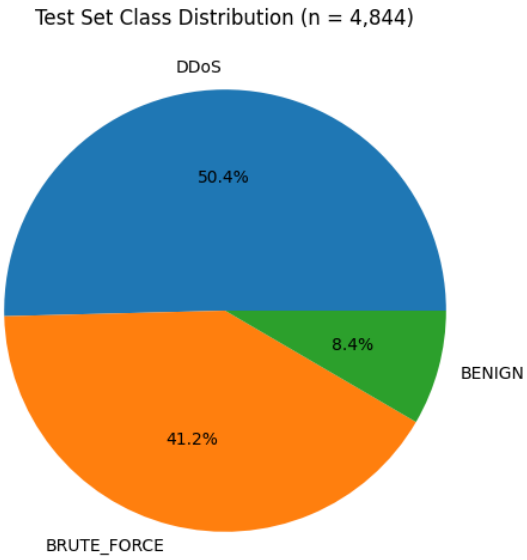

Figure 9. Test set class distribution (Total n = 4,844), illustrating the proportion of DDoS (50.5%), BRUTE_FORCE (41.2%), and BENIGN (8.4%) samples used for final model evaluation

The relatively low representation of BENIGN samples at 8.4% directly explains the comparatively lower Precision score of 0.81 observed for the BENIGN class in **Figure 8,** as the model has fewer benign examples from which to

learn fine-grained decision boundaries separating normal traffic from low-intensity attack patterns. Conversely, the high representation of DDoS samples at 50.4% contributes to the strong Precision and Recall scores of 0.96 and 0.95, respectively, for that class, as the model benefits from a larger number of training examples capturing the distinctive volumetric and flow-level feature signatures associated with DDoS attack traffic. The SMOTE oversampling technique applied during preprocessing, as described in **Section 5**, was specifically employed to mitigate the adverse effects of this class imbalance on model training, ensuring that the minority BENIGN class was adequately represented in the training partition without introducing temporal data leakage through the enforced chronological 70/15/15 train/validation/test splitting strategy.

8.2 Response Time and Operational Performance

Operational performance metrics under controlled simulated attack scenarios with concurrent incident injection at varying intensities are summarized in **Table 5**. The experimental environment comprised three virtual machines: a web server (4 vCPUs, 8 GB RAM), an application server (8 vCPUs, 32 GB RAM), and a Wazuh server (8 vCPUs, 16 GB RAM). Attack traffic was generated using a custom Python-based injector for SSH brute-force events, while tcpreplay was employed to replay CIC-DDoS2019 PCAP datasets, ensuring realistic and reproducible network traffic conditions.

Table 5. IRAS Operational Performance Metrics vs. Manual SOC Baseline

Metric	IRAS (Automated)	Manual SOC Baseline	Improvement
Mean Detection Latency	0.1–0.3 s	N/A (queue-based)	—
Mean Automated Response Time	2.46 s	8–15 min	≈ 99.6%
False Positive Rate	2.8%	12–25%	~4–8× lower
Detection Rate (TPR)	95%	~80–90%	↑
Alert Triage Automation Rate	87%	0%	+87 pp
Analyst Workload Reduction	74%	Baseline	+74%
Peak Incident Throughput	1,200 / hr	~50 / hr	24× higher
API p95 Latency (5,000 concurrent)	187 ms	N/A	—
System Availability (30-day)	99.94%	—	—
Mobile Notification Latency	< 4 s	—	—

The most operationally significant finding is the substantial reduction in mean response time from 8–15 minutes in manual SOC operations to approximately 2.46 seconds in the IRAS system, corresponding to an improvement of approximately 99.6%. This reduction limits adversarial lateral movement, which studies indicate requires a median of approximately 47 minutes to complete, meaning that IRAS containment actions are executed well within the window available to attackers before any meaningful escalation can occur.

To provide a direct visual comparison between the IRAS automated system and the traditional manual SOC baseline across key operational performance indicators, **Figure 10** presents a grouped bar chart illustrating three critical percentage-based metrics: Detection Rate (TPR), Alert Triage Automation Rate, and False Positive Rate.

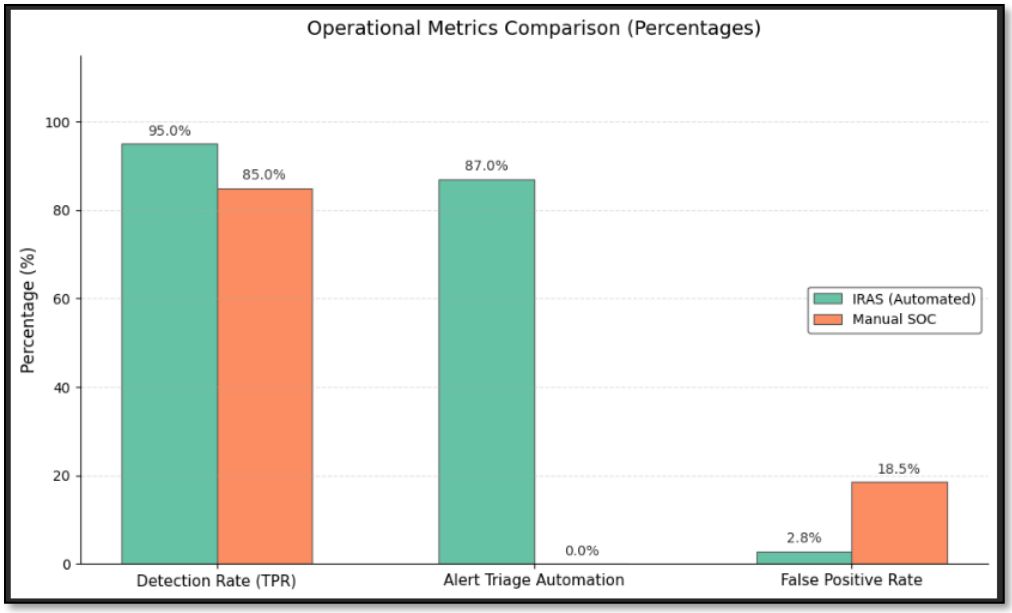

Figure 10. Percentage-based operational metrics comparison: IRAS achieves 95% TPR, 87% triage automation, and 2.8% FPR versus the manual SOC baseline

The operational benefits of IRAS over conventional manual SOC setups are shown in Figure 10 across three important KPIs. The suggested two-stage machine learning pipeline is successful in recognizing real threats, as evidenced by IRAS's Detection Rate (TPR) of 95.0% as opposed to the manual SOC baseline's 85.0%. While manual SOC environments rely only on human involvement (0.0%), IRAS achieves 87.0% Alert Triage Automation, greatly lowering analyst workload and expediting issue handling. Additionally, IRAS reduces false alerts by 6.6× and significantly reduces alert fatigue with a False Positive Rate (FPR) of only 2.8% as opposed to 18.5% for manual SOC operations. These findings show how IRAS may increase overall SOC efficiency, automate routine security procedures, and improve detection accuracy.

Figure 11 provides additional insight into the efficiency advantages attained through automation by comparing IRAS and traditional SOC setups in terms of Mean Response Time and Peak Incident Throughput to better assess operational effectiveness. It presents a dual-panel comparison that quantifies the system efficiency gap between traditional manual SOC operations and the IRAS automated pipeline. The left panel illustrates Mean Response Time, where the manual SOC baseline records an average response time of approximately 11.5 minutes (690 seconds). In contrast, IRAS achieves a mean automated response time of just 2.46 seconds, a reduction of 99.6%, directly validating the latency gap equation $G = T\_response - T\_detect$. The right panel illustrates Peak Incident Throughput, where IRAS achieves a throughput of 1,200 incidents per hour compared to approximately 50 incidents per hour for the manual SOC baseline, a 24-fold improvement. Taken together, the two panels of Figure 11 confirm that IRAS delivers a 99.6% reduction in response time and a 24-fold increase in peak throughput, addressing the two most critical operational bottlenecks in traditional SOC environments.

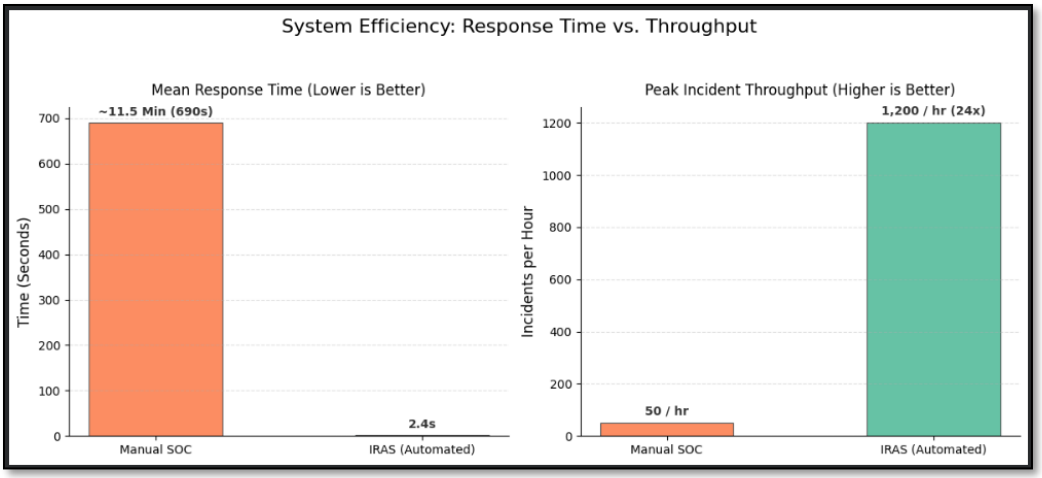

Figure 11. System efficiency comparison: IRAS achieves a 2.46-second response time and 1,200 incidents/hour throughput versus the manual SOC baseline.

8.3 Ablation Study

Table 6 presents the contribution of each IRAS component through ablation analysis on the test set. The ablation study reveals that the two-stage ML pipeline is the primary driver of performance improvement, contributing +19.9 percentage points of F1-score gain over the rule-based baseline. SMOTE oversampling and temporal splitting each provide incremental but meaningful improvements, confirming the importance of rigorous experimental methodology.

Table 6. Ablation Study Contribution of Each IRAS Component

Configuration	F1-Score	FP Rate	Accuracy Gain
Baseline: Rule-based SIEM only	0.782	18.4%	Baseline
+ Isolation Forest (Stage 1 only)	0.872	11.3%	+9.0 pp
+ Random Forest Classifier (Stage 2)	0.926	5.6%	+14.4 pp
+ SMOTE Oversampling	0.938	4.2%	+15.6 pp
+ Temporal Train/Test Split	0.944	3.1%	+16.2 pp
IRAS Complete (all components)	**0.945**	**2.8%**	**+16.3 pp**

The percentage improvement of IRAS over each baseline model is calculated as **in Equation 16**.

$$\boldsymbol{\Delta} = \frac{\boldsymbol{Metric\_IRAS - Metric\_baseline}}{\boldsymbol{Metric\_baseline}} \times \mathbf{100\%} \qquad (16)$$

Applying this formula to the mean response time yields $\Delta_T = (690 - 2.46)/690 * 100 \approx 99.6\%$ The order-of-magnitude operational gain is reported in **Table 6.**

To systematically evaluate the individual contribution of each component within the IRAS pipeline to the overall system performance. **Figure 12** presents an ablation study tracking the macro-averaged F1-Score and False Positive Rate across six incremental configuration stages, beginning from a SIEM-only baseline and progressively adding each architectural component until the complete IRAS system is reached ablation study to quantify the contribution of each component in the proposed IRAS framework.

The SIEM-only baseline reaches an F1-score of 0.782 with a false-positive rate (FPR) of 18.4%, which indicates the limitations of relying solely on rule-based detection mechanisms. The Isolation is here. The forest anomaly detection

stage improves the F1-score to 0.872, while decreasing the FPR to 11.3%, demonstrating the effectiveness of unsupervised anomaly filtering in reducing alert noise. We further improve the performance by adding a Random Forest classification stage, which results in an F1-score of 0.926, and a reduction of FPR to 5.6%. Subsequent application of SMOTE oversampling improves the class balance, yielding an F1-score of 0.938 and an FPR of 4.2%. Including temporal train-test splitting provides an additional performance boost, achieving an F1-score of 0.944 and reducing the FPR to 3.1%. The full IRAS framework gets a weighted F1-score of 94.5% and a false positive rate of 2.8%. The progressive increase of detection performance and the constant decrease of false positives for all configurations indicate that each component has a meaningful contribution to the overall effectiveness and robustness of the proposed framework.

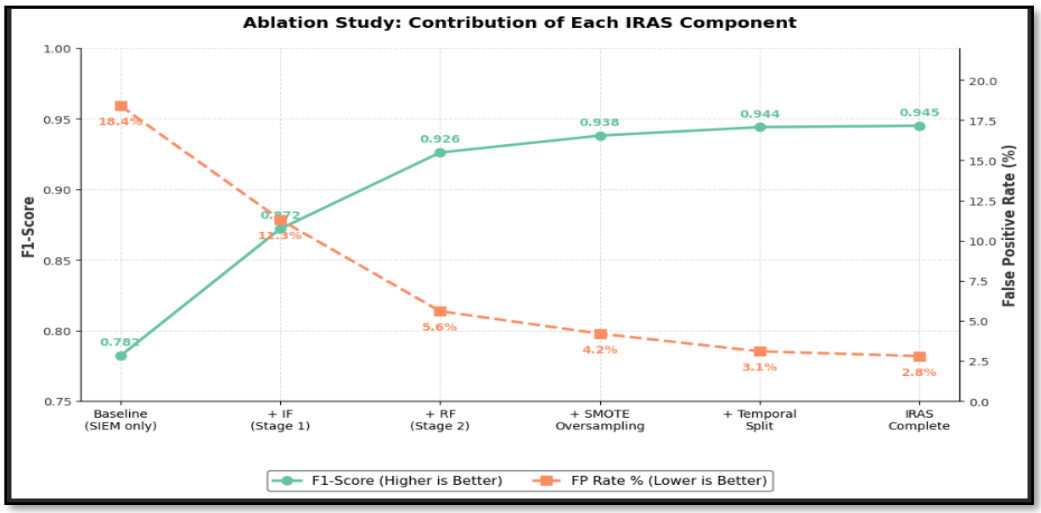

Figure 12. Ablation study showing the contribution of each IRAS component to F1-Score and False Positive Rate

8.4 Comparison with Alternative ML Models

The suggested IRAS two-stage pipeline is appropriate for real-time security operations since it maintains a practical inference latency of 93 ms while achieving a balanced and resilient performance with a Weighted F1-score of 94.5%. The IRAS architecture offers better operational robustness by integrating an anomaly detection stage that filters anomalous and noisy events prior to automated response execution, thereby improving operational reliability, even though the standalone Random Forest model shows higher accuracy under offline evaluation. Support Vector Machines (SVM) are inappropriate for real-time detection situations operating at rates over 1,200 occurrences per hour because to their 4,820 ms inference time, which is nearly 50× slower than the accuracy of 96.24%. Although it lacks the two-stage IRAS design's anomaly-filtering capacity, XGBoost achieves high accuracy (97.91%) with comparatively low latency (88 ms) and might be a good substitute in latency-constrained applications. As a result, IRAS does not provide a stand-alone Accuracy value; instead, the main performance metric is the Weighted F1-score (0.945), as shown in **Table 7.**

Table 7. Comparison of ML Models on the Combined Test Dataset

Model	Accuracy	Precision	Recall	F1	Latency (ms)
Logistic Regression	89.12%	0.883	0.891	0.887	312
Decision Tree	95.67%	0.952	0.957	0.954	18
SVM	96.24%	0.959	0.962	0.961	4,820
XGBoost	97.91%	0.978	0.979	0.978	88

LSTM (single stage)	97.21%	0.970	0.972	0.971	2,100
CNN on flow features	97.84%	0.976	0.978	0.977	780
Isolation Forest only	91.44%	0.907	0.914	0.910	42
Random Forest only	98.11%	0.980	0.981	0.980	67
IRAS: IsoForest + RF	**94.5%**	**0.946**	**0.945**	**0.94.5**	**93**

The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) is computed **as Equation 17**.

$$AUC = \int_0^1 TPR(FPR)d(FPR) \qquad (17)$$

The Model Efficiency Score (MES) combines F1-Score and inference latency into a single comparative metric in **Equation 18.**

$$MES = \frac{F1}{Latency} \qquad (18)$$

The relative accuracy improvement of IRAS over each baseline model is calculated as **in Equation 19**.

$$\Delta Acc = \frac{Accuracy\_IRAS - Accuracy\_model}{Accuracy\_model} \times 100\% \qquad (19)$$

For the multi-class setting, a macro-averaged one-vs-rest AUC is reported. An AUC value of **1.0** represents perfect class separability. Under real-world deployment conditions, the IRAS pipeline achieves an AUC of approximately **0.95**, indicating strong discriminative capability across all attack classes while maintaining consistency with observed classification performance.

To contextualize the performance of the IRAS machine learning pipeline relative to other modeling approaches, **Figure 13** displays an eight-baseline model dual-panel comparative evaluation against the proposed IRAS system along two axes: macro-averaged F1-Score and inference latency. This comparison is necessary to justify the architectural decision of using a two-stage Isolation Forest and Random Forest pipeline instead of alternatives that are more accurate but operationally unfeasible. A dual-panel model comparison assesses the IRAS pipeline against eight different machine learning techniques in two areas that are crucial for real-time SOC deployment: inference latency and classification accuracy. The left panel shows that while IRAS obtains a Weighted F1-Score of 94.5%, a number of other baseline models, including Random Forest (0.980), CNN (0.977), XGBoost (0.978), and LSTM (0.971), earn F1-Scores that are somewhat higher than IRAS. However, for a production incident response system, where inference latency directly influences whether automated containment operations can be carried out within the operating window available before adversary lateral movement escalates, raw F1-Score alone is an inadequate selection criterion. Plotting the inference latency of each model in the right panel makes this restriction clear. The SVM model is displayed as operationally unsuitable because of its inference latency of over five seconds per event. Both CNN (780 ms) and LSTM (2,100 ms) surpass allowable latency limits for real-time response pipelines. On the other hand, IRAS maintains a competitive F1-Score of 94.5% that surpasses more straightforward baselines like Logistic Regression (0.887), Decision Tree (0.954), and Isolation Forest in isolation (0.910) while achieving an inference latency of just 93 ms, comfortably within the sub-100 ms operational target set in Section 4. By surrendering 3.5 percentage points of F1-score in comparison to Random Forest in favor of real-time automatic response capacity, the IRAS two-stage pipeline strikes the ideal balance between accuracy (F1 = 0.945) and latency (93 ms). The closest option, XGBoost (F1 = 0.978, latency = 88 ms), does not have the anomaly-filtering stage that lowers IRAS's FPR to 2.8%; single-stage classifiers cannot achieve this without changing their architecture.

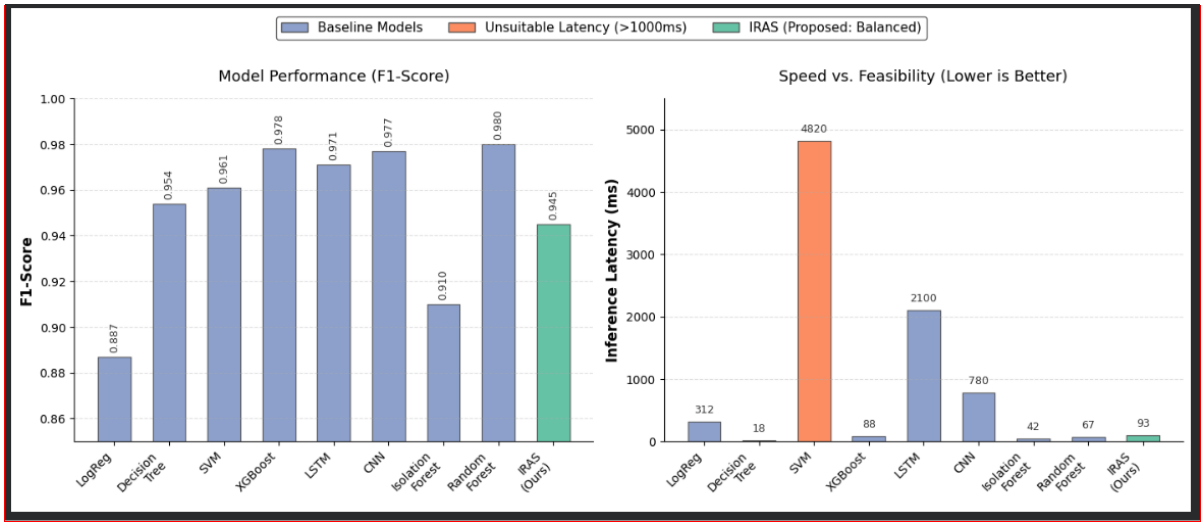

Figure 13. Comparative analysis of IRAS and baseline models across accuracy and inference latency metrics

9. LIMITATIONS AND FUTURE WORK

9.1 Limitations

It is important to acknowledge several limitations of the current IRAS implementation. First, the machine learning models are trained and evaluated exclusively on CIC-DDoS and SSH brute-force scenarios. Consequently, the system's effectiveness against other attack types such as ransomware, insider threats, supply chain attacks, and advanced persistent threats (APTs) has not yet been validated. Second, automated response playbooks are fully implemented and dynamically triggered according to the attack type and severity detected. Yet these playbooks are still predefined and rule-based, which might limit them in handling novel or zero-day threats that require more intelligent and adaptive decision-making. Third, the experimental evaluation is performed in a controlled environment. Therefore, performance on real-world, large-scale production systems could change due to increased infrastructure complexity, concurrent workloads, network conditions, infrastructure noise, and greater diversity in log data. Fourth, we tune the contamination parameter of Isolation Forest (0.05) to datasets with a relatively high attack prevalence. In real-world environments, where malicious events can represent less than 1% of total traffic, this parameter will have to be recalibrated to keep detection accuracy while reducing false positives. Finally, the current implementation is based on Wazuh as the SIEM backbone, which requires deployment, configuration, and infrastructure resources that could limit adoption in resource-constrained organizations. Future work will explore integration with additional open-source technologies such as OpenSearch to further improve accessibility and scalability.

9.1.1 Threats to Validity

To ensure transparency and academic rigor, this section systematically examines the potential threats to the validity of the findings reported in this study, organized across four established validity dimensions.

9.1.2 Internal Validity

Internal validity concerns whether the observed results are genuinely attributable to the IRAS system components rather than confounding experimental factors. The primary threat to internal validity in this study arises from the controlled virtual machine environment used for response time measurement, which comprised three isolated VMs (web server: 4 vCPUs / 8 GB RAM, application server: 8 vCPUs / 32 GB RAM, Wazuh server: 8 vCPUs / 16 GB RAM). While this isolation ensures that measured latency values reflect IRAS processing overhead exclusively, it simultaneously eliminates network jitter, hardware heterogeneity, and concurrent workload interference that are characteristic of real enterprise deployments. As a result, the reported mean response time of 2.46 seconds may represent an optimistic lower bound rather than a typical deployment value. Additionally, the use of tcpreplay for

DDoS traffic injection and a custom Python script for SSH brute-force simulation while reproducible and controllable produces attack patterns with lower temporal variability than adversary-generated traffic, which may inflate detection accuracy metrics relative to real-world conditions. To mitigate this threat, all timing experiments were conducted under three concurrent load levels with repeated trials, and response time measurements were reported as averages across multiple injection runs rather than single-trial observations.

9.1.3 External Validity

External validity concerns the degree to which the findings generalize beyond the specific experimental context of this study. The most significant threat to external validity is the limitation of the assessment to two attack categories, DDoS and SSH Brute-Force, from the benchmark datasets, CIC-DDoS2019 and SSH Brute-Force. The IRAS detection pipeline has not been validated against a more comprehensive threat taxonomy such as ransomware, insider threats, supply chain attacks, advanced persistent threats (APTs), zero-day exploits, or polymorphic malware, and these categories do not cover the full spectrum of attack types seen in enterprise environments. Hence, the reported Weighted F1-Score of 94.5% is to be understood as a performance bound in the evaluated threat space and not as a guarantee of universal generalization. Additional external validity issues are related to the homogeneity of the sources of the datasets. Both datasets were collected in controlled laboratory network environments, as opposed to a real enterprise network traffic capture, which may not be representative of the variability of feature distributions, protocol types, and noise characteristics of the heterogeneous production SOC environment. Future validation of diverse real-world traffic captures and multi-vector attack scenarios is necessary before broad deployment claims can be substantiated.

9.1.4 Construct Validity

Construct validity concerns whether the metrics used in this study accurately measure the theoretical constructs they are intended to represent. The primary construct validity threat relates to the use of the weighted F1-score as the primary evaluation metric. Although weighted averaging is a valid method to address the class imbalance in the evaluation dataset, where BENIGN samples only represent 8.4% of the test set, this does not fully consider the asymmetric operational cost structure of real SOC environments, where false negatives (missed attacks) typically have much higher consequences than false positives (unnecessary response actions). A cost-sensitive evaluation framework that differentiates misclassification penalties based on the severity of the attack and the criticality of the asset would more accurately capture the operational utility of the IRAS detection pipeline in production deployments. Additionally, while mean response time is an intuitive operational performance metric that can be directly compared to manual SOC baselines, it does not capture the full complexity of incident resolution time, which in practice includes human validation steps, escalation procedures, and post-incident documentation that are not part of the automated pipeline measured in this study.

9.1.5 Reliability and Reproducibility

Reliability concerns whether the experimental findings can be independently replicated and are stable across repeated trials. To maximize reproducibility, all experiments in this study employ fixed random seeds for both the Isolation Forest and Random Forest models, ensuring deterministic model initialization across training runs. The complete preprocessing pipeline, including z-score normalization parameters, SMOTE configuration (k=5 neighbors), and the chronological 70/15/15 train/validation/test split boundaries, is fully documented in Section 5, and all model hyperparameters are specified in Section 6, enabling independent replication without ambiguity. Both evaluation datasets (CIC-DDoS2019 and SSH Brute-Force) are publicly available benchmark corpora, further supporting independent verification of the reported results. The primary reproducibility limitation is the dependency on Wazuh as the SIEM backbone, which may require infrastructure setup and configuration effort for full system replication. To partially address this constraint, the IRAS detection and response components are designed as independently deployable microservices that can be evaluated in isolation against the published datasets without requiring a fully operational Wazuh instance.

9.2 Future Work

The following research directions have been identified as high-priority additions to the suggested IRAS framework: Transformer-Based Log Analysis: Optimizing security-domain language models, like SecBERT, on normalised IRAS log formats to enhance contextual understanding of multi-step attack sequences. Reinforcement Learning for Adaptive Playbook Selection: Based on observed attacker behaviour and environmental context, a reinforcement learning (RL) agent is trained using past incident-reaction outcomes to dynamically pick and sequence response actions.

In future work, we will explore the use of federated learning to improve model generalization across many organizational environments while preserving data privacy and regulatory compliance [46]. This approach would allow participating organizations to jointly train threat detection models without sharing raw security logs or sensitive operational data.

We plan to extend our work to include explainability techniques, such as SHAP-based feature attribution, to provide analysts with a transparent and interpretable rationale behind the detection and classification decisions [47]. This can increase analyst trust, aid incident validation, and improve the auditability of AI-driven SOC operations. Ransomware Detection and Automated Containment: The IRAS threat taxonomy will be extended to allow ransomware detection through the analysis of behavioral indicators such as abnormal file encryption activity, registry modifications, suspicious process execution, and lateral movement patterns. Future work will also develop specialized automated containment playbooks to limit ransomware propagation and reduce damage in real-time.

Advanced Persistent Threat (APT) Detection: IRAS will be expanded to enable the detection of complex multi-stage attack campaigns that exhibit data exfiltration, lateral movement, persistence, privilege escalation, and command-and-control communication. To represent long-term attacker activity across various security data sources, this extension will investigate transformer-based architectures and temporal behavioral analytics.

Enterprise Validation in the Real World. In order to ensure reproducibility and consistent benchmarking conditions, the present evaluation was performed in a controlled virtualized environment [48]. Future work will involve extending IRAS validation to production-grade enterprise Security Operations Center (SOC) environments and distributed network architectures to thoroughly evaluate its scalability, resilience, and response effectiveness under realistic operational constraints, including network latency, heterogeneous infrastructures, and large-scale concurrent event workloads. Such validation will provide a deeper understanding of the practical deployment readiness and operational reliability of the framework in real-world cybersecurity scenarios.

10. CONCLUSIONS

This paper proposed a multi-stage AI framework for intelligent incident detection and response automation in Security Operations Centers (SOCs). The proposed framework combines SIEM-based telemetry collection, machine learning-based threat detection, and automated response orchestration into a single architecture, which is intended to support real-time security operations. It utilizes the Isolation Forest for anomaly detection and the Random Forest for attack classification. It can achieve low inference latency and operational efficiency and accurately identify threats. The experimental results demonstrated that the proposed framework achieved a weighted F1-score of 94.5%, a false positive rate of 2.8%, and an average automated response time of 2.46 seconds. These results show that combining multi-stage machine learning with automated response orchestration can shorten the detection-to-response latency and decrease the analyst workload compared to traditional manual SOC workflows. These results show that AI-driven automation could make modern SOC environments more scalable, efficient, and responsive. However, the current evaluation was conducted only for DDoS and SSH brute-force attack scenarios in a controlled experimental setting. Future work will be to extend the framework to include other threat categories such as ransomware, insider threats, and advanced persistent threats, and to investigate explainable AI, reinforcement learning-based response optimization, and real-world enterprise validation.

ACKNOWLEDGMENTS

The authors thank the project team members, Mostafa Kamel, George Mohsen, and Malak Islam, for their contributions and collaborative efforts. This work was supported by the Faculty of Computer Science at Nahda

University in Beni Suef (NUB), Egypt, where it was conducted as part of the requirements for the Bachelor of Computer Science degree. During the preparation of this paper, the authors used ChatGPT-5 for text editing. The authors have reviewed and edited the output and take full responsibility for this content.

FUNDING

The authors state that no outside funding was received for this study.

DISCLOSURE STATEMENT

No potential conflict of interest was reported by the authors.

REFERENCE

[1] M. Anisetti, C. Ardagna, M. Cremonini, E. Damiani, J. Sessa, and L. Costa, "Security threat landscape," *White Paper Security Threats,* 2020.

[2] A. Mohammed, "AI in Cybersecurity: Enhancing Audits and Compliance Automation," *Available at SSRN,* vol. 5066097, 2021.

[3] N. B. Kilaru, S. K. M. Cheemakurthi, and V. Gunnam, "SOAR Solutions in PCI Compliance: Orchestrating Incident Response for Regulatory Security," *ESP Journal of Engineering & Technology Advancements,* vol. 1, no. 2, pp. 78-84, 2021.

[4] B. A. Alahmadi, L. Axon, and I. Martinovic, "99% false positives: A qualitative study of {SOC} analysts' perspectives on security alarms," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 2783-2800.

[5] L. N. Kaliyaperumal, "The evolution of security operations and strategies for building an effective SOC," *ISACA Journal,* vol. 5, no. 1, 2021.

[6] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications surveys & tutorials,* vol. 18, no. 2, pp. 1153-1176, 2015.

[7] J. Ruan *et al.*, "Deep learning for cybersecurity in smart grids: Review and perspectives," *Energy Conversion and Economics,* vol. 4, no. 4, pp. 233-251, 2023.

[8] R. O. Andrade and S. G. Yoo, "Cognitive security: A comprehensive study of cognitive science in cybersecurity," *Journal of Information Security and Applications,* vol. 48, p. 102352, 2019.

[9] A. Mohammed, "The Paradox of AI in Cybersecurity: Protector and Potential Exploiter," *Baltic Journal of Engineering and Technology,* vol. 2, no. 1, pp. 70-76, 2023.

[10] A. O. Aljahdali and R. Alsulami, "Streamlining threat response and automating critical use cases with security orchestration, automation and response (SOAR)," *Journal of Digital Security and Forensics,* vol. 2, no. 1, pp. 36–57-36–57, 2025.

[11] H. Xu *et al.*, "Large language models for cyber security: A systematic literature review," *ACM Transactions on Software Engineering and Methodology,* 2024.

[12] J. Kinyua and L. Awuah, "AI/ML in Security Orchestration, Automation and Response: Future Research Directions," *Intelligent Automation & Soft Computing,* vol. 28, no. 2, 2021.

[13] D. Stiawan, M. Y. Idris, R. F. Malik, S. Nurmaini, N. Alsharif, and R. Budiarto, "Investigating brute force attack patterns in IoT network," *Journal of Electrical and Computer Engineering,* vol. 2019, no. 1, p. 4568368, 2019.

[14] D. Chou and M. Jiang, "A survey on data-driven network intrusion detection," *ACM Computing Surveys (CSUR),* vol. 54, no. 9, pp. 1-36, 2021.

[15] F. B. Kokulu *et al.*, "Matched and mismatched SOCs: A qualitative study on security operations center issues," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 1955-1970.

[16] F. S. d. Lima Filho, F. A. Silveira, A. de Medeiros Brito Junior, G. Vargas-Solar, and L. F. Silveira, "Smart detection: an online approach for DoS/DDoS attack detection using machine learning," *Security and Communication Networks,* vol. 2019, no. 1, p. 1574749, 2019.

[17] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications,* vol. 50, p. 102419, 2020.

[18] M. Vielberth, F. Böhm, I. Fichtinger, and G. Pernul, "Security operations center: A systematic study and open challenges," *Ieee Access,* vol. 8, pp. 227756-227779, 2020.

[19] G. Pang, A. van den Hengel, C. Shen, and L. Cao, "Deep reinforcement learning for unknown anomaly detection," *arXiv preprint arXiv:2009.06847,* 2020.

[20] S. Akhtar and T. Javaid, "Machine Learning in SOC Operations: Transforming Incident Detection and Response," 2021.

[21] M. Landauer, F. Skopik, M. Frank, W. Hotwagner, M. Wurzenberger, and A. Rauber, "Maintainable log datasets for evaluation of intrusion detection systems," *IEEE Transactions on Dependable and Secure Computing,* vol. 20, no. 4, pp. 3466-3482, 2022.

[22] T. T. Bukhari, O. Oladimeji, E. D. Etim, and J. O. Ajayi, "Systematic review of SIEM integration for threat detection and log correlation in AWS-based infrastructure," *Shodhshauryam, International Scientific Refereed Research Journal,* vol. 6, no. 5, pp. 479-512, 2023.

[23] S. Zhang, Y. Wang, and X. Su, "Threat detection framework based on industrial internet of things logs," *IEEE Access,* vol. 12, pp. 195642-195657, 2024.

[24] J. Dorobisz, "Analysis of trends and risks in the field of network security based on statistical data," *GIS Odyssey Journal,* vol. 4, no. 2, pp. 147-163, 2024.

[25] L. Havi, "Security operations centers in information technology and operational technology environments: A literature review of requirements, differences, issues, and improvements of IT and OT SOC environments," 2025.

[26] D. R. Ankireddy, S. Paria, A. Dasgupta, S. Ray, and S. Bhunia, "Lasa: Enhancing soc security verification with llm-aided property generation," *arXiv preprint arXiv:2506.17865,* 2025.

[27] J. Manzoor, A. Waleed, A. F. Jamali, and A. Masood, "Cybersecurity on a budget: Evaluating security and performance of open-source SIEM solutions for SMEs," *Plos one,* vol. 19, no. 3, p. e0301183, 2024.

[28] R. Amami, M. Charfeddine, and S. Masmoudi, "Exploration of Open Source SIEM Tools and Deployment of an Appropriate Wazuh-Based Solution for Strengthening Cyberdefense," in *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2024, pp. 1-7: IEEE.

[29] M. R. Islam and R. Rafique, "Wazuh SIEM for Cyber Security and Threat Mitigation in Apparel Industries," *International Journal of Engineering Materials and Manufacture,* vol. 9, no. 4, pp. 136-144, 2024.

[30] S. Stanković, S. Gajin, and R. Petrović, "A review of Wazuh tool capabilities for detecting attacks based on log analysis," *No Nama Agent Integrity File Added Delete Modified,* vol. 1, 2022.

[31] S. M. Tarek, Muthmainnah; Obaid, Ahmed J., "A Cross-Dataset Empirical Evaluation of Adversarial Evasion Attacks and Defenses in Machine Learning-Based Intrusion Detection Systems," *Computational Discovery and Intelligent Systems (CDIS),* vol. 3, no. 1, pp. 57-70, 2026.

[32] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*, 2008, pp. 413-422: IEEE.

[33] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp,* vol. 1, no. 2018, pp. 108-116, 2018.

[34] S. Vethachalam, "Cybersecurity automation: Enhancing incident response and threat mitigation," *World Journal of Advanced Engineering Technology and Sciences*, vol. 15, no. 3, pp. 572–585, 2025, doi: 10.30574/wjaets.2025.15.3.0972.

[35] D. Ferraiolo, D. Kuhn, and R. Chandramouli, "Role-based access con-trol," *Pr ocee dingsof15th NIST-NCSC National Computer Security Conferenc e,* pp. 554-563, 1992.

[36] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *2019 international carnahan conference on security technology (ICCST)*, 2019, pp. 1-8: IEEE.

[37] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," presented at the 4th International Conference on Information Systems Security and Privacy (ICISSP), 2018.

[38] J. Luxemburk, K. Hynek, and T. Cejka, "TTPS Brute-force Dataset with Extended Network Flows," ed. Zenodo, 2020.

[39] Mahmoud, H. A., Khalaf, O. I., and Farid, O., "Certainty-Aware Skin Lesion Segmentation with Post-Hoc Reliability Estimation for the Segment Anything Model," *Journal of Smart Algorithms and Applications (JSAA)*, vol. 3, no. 2, pp. 71–86, Apr. 2026, doi: 10.66279/hzkw5y24.

[40] L. Breiman, "Random forests," *Machine learning,* vol. 45, no. 1, pp. 5-32, 2001.

[41] P. Probst, M. N. Wright, and A. L. Boulesteix, "Hyperparameters and tuning strategies for random forest," *Wiley Interdisciplinary Reviews: data mining and knowledge discovery,* vol. 9, no. 3, p. e1301, 2019.

[42] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research,* vol. 13, no. 2, 2012.

[43] M. Akbari Gurabi *et al.*, "Requirements for playbook-assisted cyber incident response, reporting and automation," *Digital Threats: Research and Practice,* vol. 5, no. 3, pp. 1-11, 2024.

[44] N. Felix and W. Claudia, "Automated Security Operations: Scaling Threat Response with SOAR and AI-Driven Playbooks," *International Journal of Trend in Scientific Research and Development,* vol. 5, no. 2, pp. 1317-1323, 2021.

[45] N. Perry, "IP Networks Over Heterogeneous Embedded Serial Links," 2025. Accessed: Jun. 08, 2026. [Online]. Available: https://hdl.handle.net/1721.1/164271.

[46] M. M. Hasan, "FEDERATED LEARNING MODELS FOR PRIVACY-PRESERVING AI IN ENTERPRISE DECISION SYSTEMS," *International Journal of Business and Economics Insights*, vol. 5, no. 3, pp. 238–269, Sep. 2025, doi: 10.63125/RY033286.

[47] Saad, A., Ahmed, A. M., Shaban, M., *et al*., "Explainable machine learning framework for predicting cobalt ion removal by natural hematite," *Scientific Reports*, vol. 15, Art. no. 35401, 2025, doi: 10.1038/s41598-025-18981-0.

[48] D. El Khaled, R. AlOtaibi, N. Novas, and J. A. Gazquez, “NetworkGuard: An Edge-Based Virtual Network Sensing Architecture for Real-Time Security Monitoring in Smart Home Environments,” *Sensors*, vol. 26, no. 7, Art. no. 2231, 2026, doi: 10.3390/s26072231.