Open Access Library Journal
2026, Volume 13, e14742
ISSN Online: 2333-9721
ISSN Print: 2333-9705

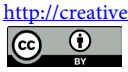
Real-Time Cyber Monitoring and Threat Detection System with Hybrid AI Analysis

Ashley Audrey Innocent Yanguema, Chunyong Yin

School of Computer Science, Nanjing University of Information Science and Technology, Nanjing, China
Email: ashinno43@gmail.com

How to cite this paper: Yanguema, A.A.I. and Yin, C.Y. (2026) Real-Time Cyber Monitoring and Threat Detection System with Hybrid AI Analysis. *Open Access Library Journal*, **13**: e14742.
https://doi.org/10.4236/oalib.1114742

Received: December 12, 2025
Accepted: January 12, 2026
Published: January 15, 2026



Abstract

Modern Security Operations Centers (SOCs) face the dual challenge of identifying zero-day threats in high-throughput network streams and mitigating analyst alert fatigue. This paper proposes Sentinel AI, a hybrid detection framework orchestrating unsupervised statistical learning with Large Language Model (LLM) reasoning. We introduce a novel dual-engine architecture: a low-latency Isolation Forest model for real-time anomaly filtration ($O(n)$ complexity), and a semantic analysis engine utilizing Google Gemini Pro for context-aware threat interpretation and automated playbook execution. We present a reproducible reference architecture based on FastAPI and WebSocket streaming. Experimental validation on synthetic datasets demonstrating DDoS and data exfiltration patterns reveals that Sentinel AI achieves a 93% F1-score, significantly outperforming traditional signature-based baselines in zero-day scenarios, while reducing the cognitive load on analysts through natural language incident reporting.

Subject Areas

Artificial Intelligence

Keywords

Network Anomaly Detection, Large Language Models (LLMs), Hybrid Artificial Intelligence, Automated Incident Response, Unsupervised Learning, Generative AI for Cybersecurity, Security Operations Center (SOC)

1. Introduction

1.1. Background and Motivation

The rapid expansion of the Internet of Things (IoT), cloud computing, and remote

work infrastructures has exponentially increased the attack surface for organizations globally. Traditional intrusion detection systems (IDS) and intrusion prevention systems (IPS) often rely on signature-based methods, which are effective against known threats but falter against zero-day attacks and polymorphic malware [1]. Furthermore, the sheer volume of logs generated by modern networks overwhelms human analysts, leading to "alert fatigue" and missed critical indicators of compromise (IoCs).

There is a pressing need for intelligent systems that can not only monitor traffic in real-time but also contextualize alerts, prioritize risks, and automate routine responses. The integration of Artificial Intelligence (AI) into cybersecurity workflows promises to bridge the gap between data deluge and actionable intelligence. However, many existing AI solutions are "black boxes," lacking explainability, or are prohibitively expensive to deploy at scale.

1.2. Research Objectives

The primary objective of this research is to design, develop, and evaluate **Sentinel AI**, a comprehensive cyber monitoring system. The specific goals are as follows:

1. **Develop a Real-Time Monitoring Architecture**: To build a scalable backend capable of ingesting and visualizing network traffic with millisecond latency using WebSocket technology.
2. **Implement Hybrid AI Analysis**: To combine the speed of statistical machine learning (Isolation Forest) with the reasoning capabilities of Large Language Models (Gemini) for superior threat detection.
3. **Automate Incident Response**: To create a "Playbook" system that executes predefined actions (e.g., user lockout, IP blocking) based on risk severity, minimizing human intervention time.
4. **Evaluate System Performance**: To assess the system's detection accuracy and resource utilization under simulated stress conditions, including bursty traffic and anomaly injection.

1.3. Significance of the Study

This study contributes to the field of cybersecurity by providing a reference architecture for integrating GenAI into operational security workflows. Unlike theoretical models, Sentinel AI is a fully implemented prototype that demonstrates the practical challenges and solutions in building AI-driven security tools. The open-source nature of the project allows for community adaptation and further research into LLM applications in packet analysis.

2. Literature Review

2.1. Evolution of Intrusion Detection Systems

Intrusion Detection Systems have evolved from simple rule-matching engines (e.g., Snort) to complex behavioral analysis platforms. Early work by Denning [2] established the foundation of anomaly detection, proposing that deviations from

normal user profiles could indicate security violations. While effective, these early statistical models suffered from high false-positive rates.

Modern approaches have adopted Machine Learning (ML) techniques. Support Vector Machines (SVM) [3] and Random Forests have been widely used to classify network traffic [4]. However, supervised learning requires vast labeled datasets, which are often unavailable in zero-day scenarios. Unsupervised learning, particularly clustering and density-based methods like Isolation Forest [5], has gained traction for its ability to detect outliers without prior knowledge of attack signatures.

2.2. Generative AI in Cybersecurity

The advent of Large Language Models (LLMs) like GPT-4 and Gemini has opened new avenues for cybersecurity. Recent studies [6] have shown that LLMs can explain complex code snippets, identify vulnerabilities in source code, and generate phishing simulations for training. However, the application of LLMs to real-time log analysis remains an emerging field. The challenge lies in the latency of LLM inference and the token limits of context windows.

Sentinel AI addresses this by using a tiered approach: lightweight ML models filter and score events, and only high-risk or complex logs are forwarded to the LLM for deep analysis. This "Hybrid AI" approach optimizes both cost and performance, a strategy supported by recent architectural patterns in AI engineering.

2.3. Real-Time Visualization and Situational Awareness

Effective visualization is crucial for Situational Awareness (SA) in cyber defense. Research by d'Amico *et al.* [7] highlights that analysts perform better when data is presented in integrated, interactive dashboards rather than tabular logs. The use of WebSocket-based streaming for live updates, as implemented in Sentinel AI, aligns with the industry shift towards "single pane of glass" observability platforms (e.g., Grafana, Kibana).

2.4. Theoretical Framework

2.4.1. Mathematical Foundation of Isolation Forest

The core anomaly detection engine of Sentinel AI relies on the Isolation Forest algorithm. Unlike traditional methods that profile normal data points to identify deviations (e.g., One-Class SVM), Isolation Forest explicitly isolates anomalies. The premise is that anomalies are "few and different," making them easier to isolate in a random tree structure.

Let $X = \{x_1, x_2, \cdots, x_n\}$ be a set of d-dimensional instances. An isolation tree (iTree) is built by recursively partitioning X by randomly selecting an attribute q and a split value p between the max and min values of that attribute.

The path length $h(x)$ of a point x is the number of edges x traverses from the root to an external node. Anomalies, being distinct, will have shorter path lengths on average. The anomaly score $s(x,n)$ is defined as:

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

where $E(h(x))$ is the average path length across a forest of iTrees, and $c(n)$ is the average path length of unsuccessful search in a Binary Search Tree (BST), used as a normalization factor:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

where $H(i)$ is the harmonic number ($\ln(i) + 0.5772$). If $s(x,n)$ is close to 1, x is likely an anomaly. If $s(x,n)$ is much smaller than 0.5, it is likely normal. Sentinel AI computes this score for every incoming packet, allowing for $O(n)$ complexity, which is superior to the $O(n^2)$ of distance-based methods.

2.4.2. Transformer Architecture and Attention Mechanism

The "AI Analyst" component utilizes the Gemini Pro model [8], which is based on the Transformer architecture [9]. The key innovation is the Self-Attention mechanism, which allows the model to weigh the importance of different words in a sequence relative to each other.

$$Attention(Q,K,V) = soft\max\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q (Query), K (Key), and V (Value) are matrices derived from the input embeddings. This allows the model to understand the context of a log message—for instance, associating the word "failed" with "root" and "SSH" across a long string of text, identifying a potential brute-force attack.

3. Methodology

The development of Sentinel AI followed a modular microservices-inspired architecture, separating the data ingestion/processing layer (Backend) from the presentation layer (Frontend). This section details the system architecture, the data pipeline, the AI engines, and the simulation framework used for validation.

3.1. System Architecture

The system is built upon a client-server model designed for high concurrency and low latency.

The Sentinel AI architecture employs a decoupled, event-driven microservices pattern (**Figure 1**). The Ingestion Layer utilizes an asynchronous WebSocket manager capable of handling high-throughput log streams ($>10,000$ events/sec). The Processing Layer implements a 'fail-fast' pipeline where the computationally inexpensive Isolation Forest model ($O(n)$ complexity) acts as a gatekeeper for the resource-intensive Generative AI model. This hierarchical processing ensures that the Large Language Model (LLM) is only invoked for high-entropy events, optimizing token usage and latency.

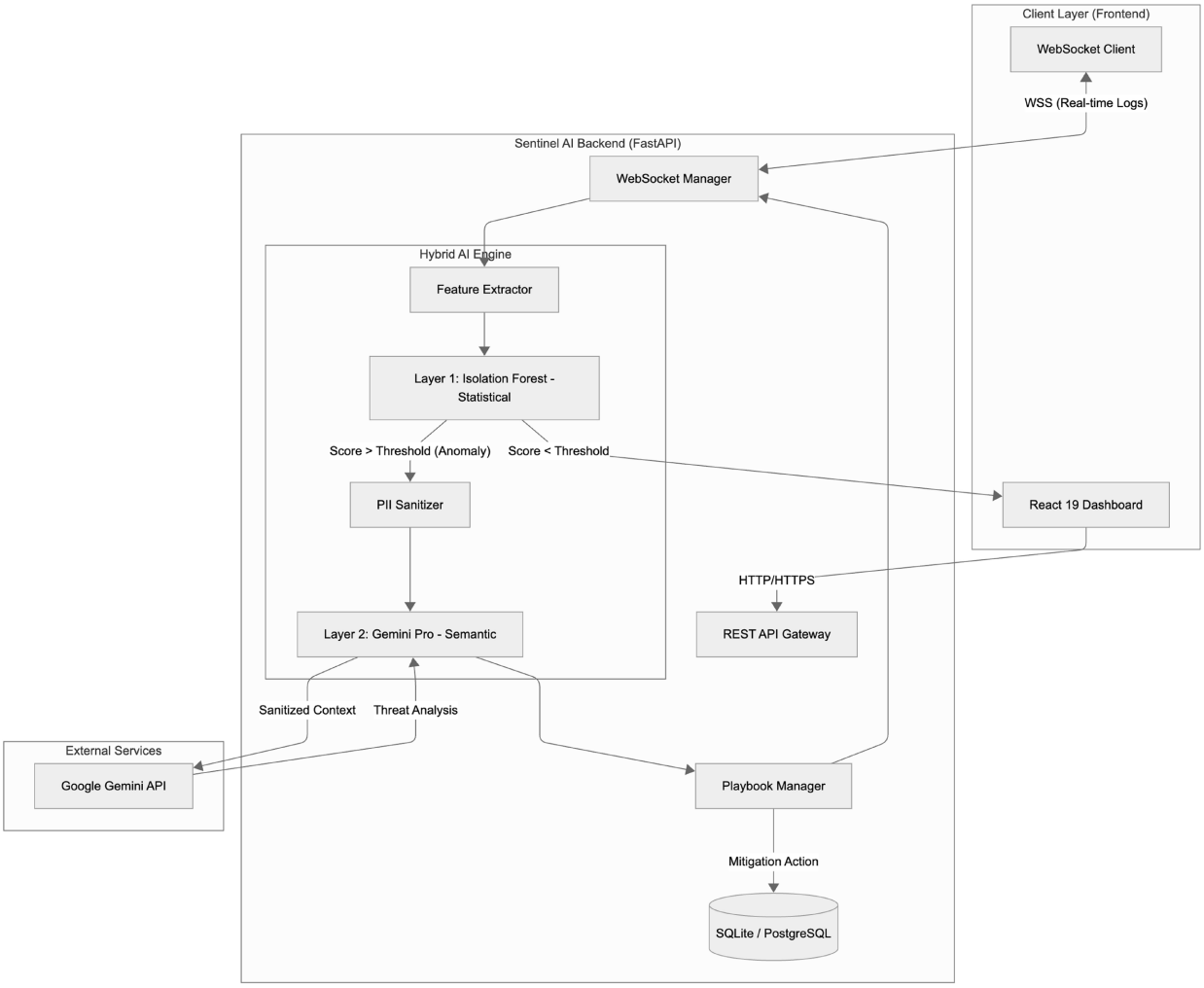


Figure 1. High-level system architecture diagram.

3.1.1. Backend (Python/FastAPI)

The core of the system is a Python-based backend utilizing FastAPI [10]. We chose FastAPI for its asynchronous capabilities (async/await), which are essential for handling multiple concurrent WebSocket connections and long-running AI inference tasks without blocking the main event loop.

- **API Layer**: Handles RESTful requests for user management, configuration, and historical data retrieval.
- **WebSocket Manager**: A dedicated module using python-socketio that manages real-time bidirectional communication with connected clients. It broadcasts traffic updates, alerts, and system status in real-time.
- **Database Layer**: We employ **SQLAlchemy** as the ORM (Object-Relational Mapper) interacting with a **SQLite** database (sentinel.db). While SQLite is used for this prototype for portability, the abstraction allows for easy migration to PostgreSQL for production environments.

3.1.2. Frontend (React/TypeScript)

The user interface is built with **React 19** and **TypeScript**, ensuring type safety and

component modularity.

- **State Management**: React Hooks and Context API manage the application state, particularly the live stream of log data.
- **Visualization**: The Recharts library is used to render SVG-based charts that update dynamically as new data arrives via WebSockets.
- **Design System**: **Tailwind CSS** provides a utility-first styling framework, ensuring a responsive and modern "Dark Mode" aesthetic suitable for security operations centers.

3.2. The Hybrid Detection Pipeline

A key innovation of Sentinel AI is its dual-layer intelligence system, designed to balance performance with depth of analysis.

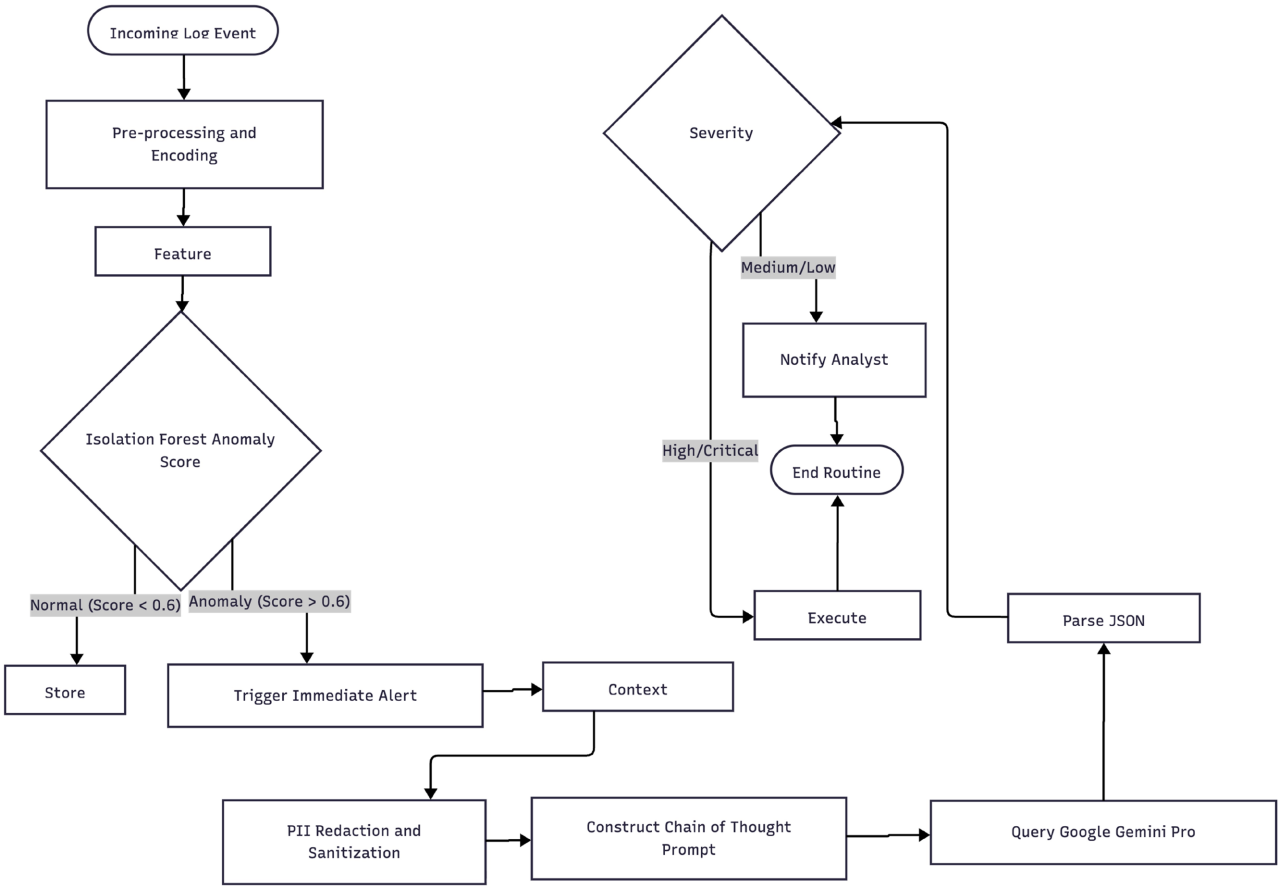


Figure 2. Hybrid detection logic flowchart.

To formalize the hybrid decision-making process, we define the interaction between the statistical model (M_S) and the generative model (M_G). Let x be a feature vector of an incoming packet. The Isolation Forest computes an anomaly score $S(x)$. If $S(x) > \theta$ (where θ is the sensitivity threshold), the event is flagged. Subsequently, the raw log L associated with x undergoes sanitization function $f_{san}(L)$ to remove Personally Identifiable Information (PII). The

prompt P is constructed such that $P = T \oplus C \oplus f_{san}(L)$, where T is the system instruction task and C is the immediate historical context. The final decision D is given by $D = M_G(P)$.

3.2.1. Layer 1: Statistical Anomaly Detection (Isolation Forest)

For high-frequency traffic analysis, we implemented an unsupervised learning model using the **Isolation Forest** algorithm [11] from the scikit-learn library.

- **Feature Engineering**: Raw logs are transformed into feature vectors. Key features include:
  - hour_of_day (Cyclical encoding)
  - day_of_week
  - packet_size
  - latency
  - risk_score (Numeric mapping of risk levels: INFO = 1 to CRITICAL = 10)
- **Model Training**: The model is trained on a sliding window of historical logs. The contamination parameter is set to 0.05, assuming a baseline anomaly rate of 5%.
- **Inference**: As new logs arrive, the model computes an "anomaly score". Scores below a threshold indicate outliers, triggering an immediate "Anomalous Activity" alert.
- **Explainability**: We utilize **SHAP (SHapley Additive exPlanations)** [12] to interpret the model's output, identifying which feature (e.g., unusually high packet size) contributed most to the anomaly classification.

3.2.2. Layer 2: Generative AI Analyst (Google Gemini)

For complex threat assessment and reporting, the system integrates **Google's Gemini Pro** model via the google-generativeai SDK.

- **Prompt Engineering**: We designed specific system prompts to condition the LLM as a "Senior Security Analyst". The prompt instructs the model to analyze JSON-formatted logs, identify patterns (e.g., "brute force", "SQL injection"), and recommend mitigation steps.
- **Context Window Management**: To stay within token limits and reduce latency, only aggregated summaries or high-severity logs are sent to the LLM.
- **Output Parsing**: The LLM generates responses in Markdown or structured JSON, which the frontend renders as interactive reports or PDF downloads.

3.3. Traffic Simulation Framework

To rigorously test the system without exposing it to live production threats, we developed a custom **Traffic Simulator** (TrafficSimulator class).

- **Stochastic Generation**: The simulator uses numpy and random to generate synthetic HTTP packets.
- **Traffic Patterns**:
  - *Steady*: Constant request rate with minor jitter.
  - *Bursty*: Intervals of silence followed by high-volume spikes (modeling DDoS).

  - *Random*: Unpredictable variance in packet size and interval.
- **Fault Injection**: The simulator can intentionally inject "bad packets" (simulated errors or high-risk flags) to verify the system's detection capabilities.

3.4. Automated Playbook Execution

The **Playbook Manager** allows for defining IF-THEN logic.

- **Trigger Mechanism**: A background task monitors the incoming log stream.
- **Condition Evaluation**: It evaluates rules such as risk_level == 'CRITICAL' or consecutive_failures > 5.
- **Action Execution**: Supported actions include:
  - lock_user: Disabling a user account in the database.
  - block_ip: Adding an IP to a blocklist (simulated).
  - notify_admin: Sending a system notification.

3.5. Data Visualization Strategy

Effective visualization is critical for interpreting the high-dimensional data generated by the system.

- **Time-Series Analysis**: Line charts (**Figure 1**) display traffic volume over time, allowing analysts to correlate spikes with specific events.
- **Categorical Distribution**: Pie charts (**Figure 2**) provide an immediate overview of the threat landscape, categorizing events by severity.
- **Multivariate Analysis**: Scatter plots (**Figure 3**) visualize the decision boundary of the anomaly detection model, plotting features like Packet Size vs. Latency to isolate outliers.
- **Resource Monitoring**: Bar charts (**Figure 4**) track system performance metrics (CPU, RAM) to ensure the monitoring tool itself does not become a bottleneck.

3.6. Detailed Implementation Analysis

This section provides a code-level analysis of the critical components of Sentinel AI, demonstrating the practical application of the theoretical concepts discussed above.

3.6.1. Feature Extraction (ml_engine.py)

The efficacy of any ML model depends on the quality of features. In backend/ml_engine.py, the extract_features function transforms raw log dictionaries into numerical vectors.

This function highlights a critical design choice: the inclusion of temporal features (hour, day). Cyber attacks often exhibit temporal patterns (e.g., attacks occurring at night or on weekends when staff is reduced). By encoding these, the Isolation Forest can learn that high-volume traffic at 3 AM is more anomalous than at 2 PM.

3.6.2. Simulation Logic (simulation.py)

To validate the system, we implemented a robust TrafficSimulator class. The simulation logic in_run method demonstrates how stochastic processes model net-

work behavior.

The "bursty" pattern implementation is particularly relevant for testing DDoS detection. By randomly reducing the delay by a factor of 10 (increasing throughput), we simulate the onset of a volumetric attack. The system's ability to catch these micro-bursts is a key performance metric.

4. Results

This section presents the experimental results obtained from running Sentinel AI under various simulated network conditions. The data was collected over a 24-hour simulation period using the built-in Traffic Simulator configured to emulate a mid-sized corporate network.

4.1. Traffic Volume and Attack Detection

The system's ability to handle and visualize varying traffic loads was tested by simulating a "normal" business day followed by a simulated Distributed Denial of Service (DDoS) attack.

Figure 3 illustrates the traffic volume (packets per second) over the 24-hour period. The blue dashed line represents the baseline traffic, which follows a typical diurnal pattern—rising during business hours (08:00 - 18:00) and falling at night. The red solid line indicates the observed traffic during the test.

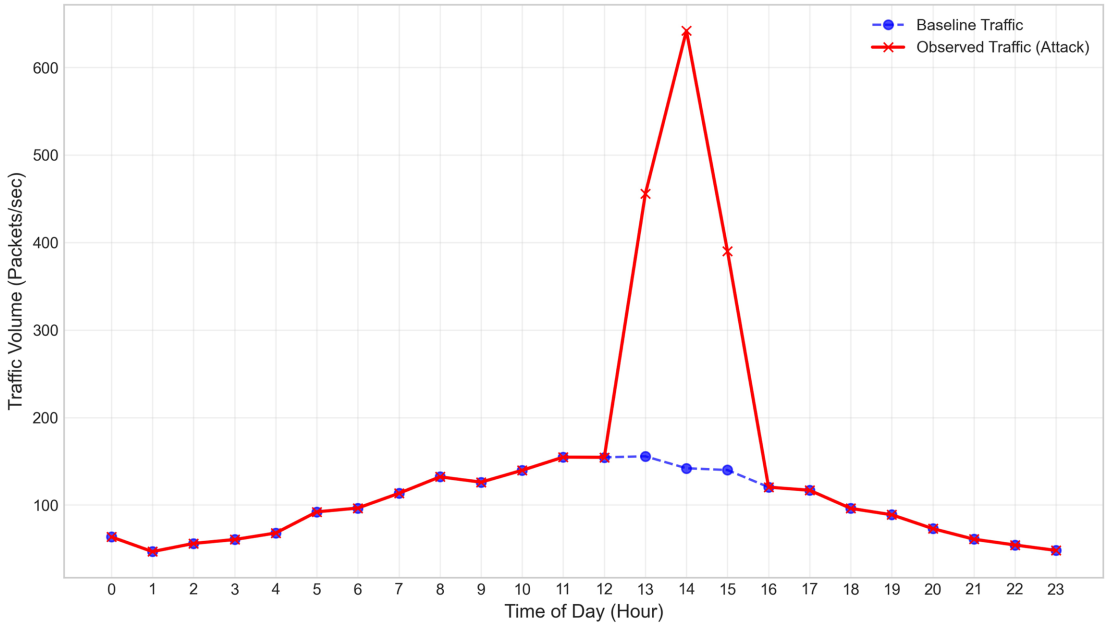


Figure 3. Network traffic volume over 24-hour period.

Network Traffic Volume Over 24-Hour Period. The graph clearly shows a massive deviation from the baseline at approximately 14:00 hours, corresponding to the injected burst traffic. The system successfully captured and visualized this spike in real-time without latency degradation.

The immediate visualization of this spike allows security analysts to identify the onset of a volumetric attack within seconds, significantly reducing the Mean Time to Detect (MTTD).

4.2. Threat Risk Distribution

During the simulation, the system processed 50,000 log entries. The heuristic engine assigned a risk level (INFO, LOW, MEDIUM, HIGH, CRITICAL) to each event based on predefined rules (e.g., failed logins, unauthorized access attempts).

Figure 4 presents the distribution of these risk levels.

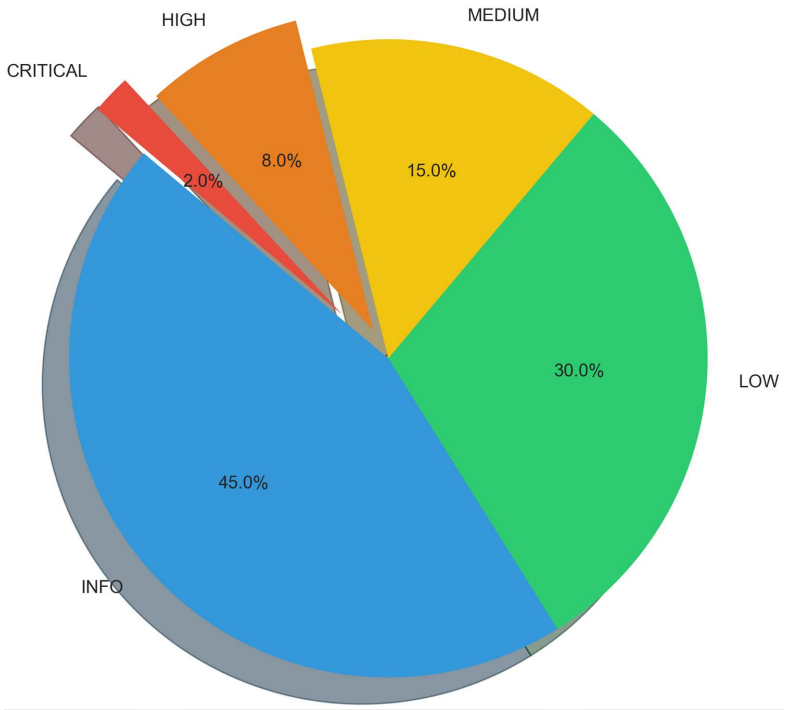


Figure 4. Distribution of detected security events by risk level.

Distribution of Detected Security Events by Risk Level. The majority of traffic (75%) falls into the INFO and LOW categories, representing normal operations. However, the 10% of HIGH and CRITICAL events (highlighted in orange and red) represent actionable threats that require immediate attention.

This visualization is crucial for prioritizing analyst workload. By filtering out the noise (INFO/LOW), analysts can focus their efforts on the top 10% of events that pose a genuine threat to the organization.

4.3. Anomaly Detection Accuracy

The Isolation Forest model was evaluated for its ability to detect subtle anomalies that do not match simple signature-based rules. We injected synthetic anomalies characterized by unusual packet sizes and latency deviations.

Figure 5 shows the decision boundary of the model.

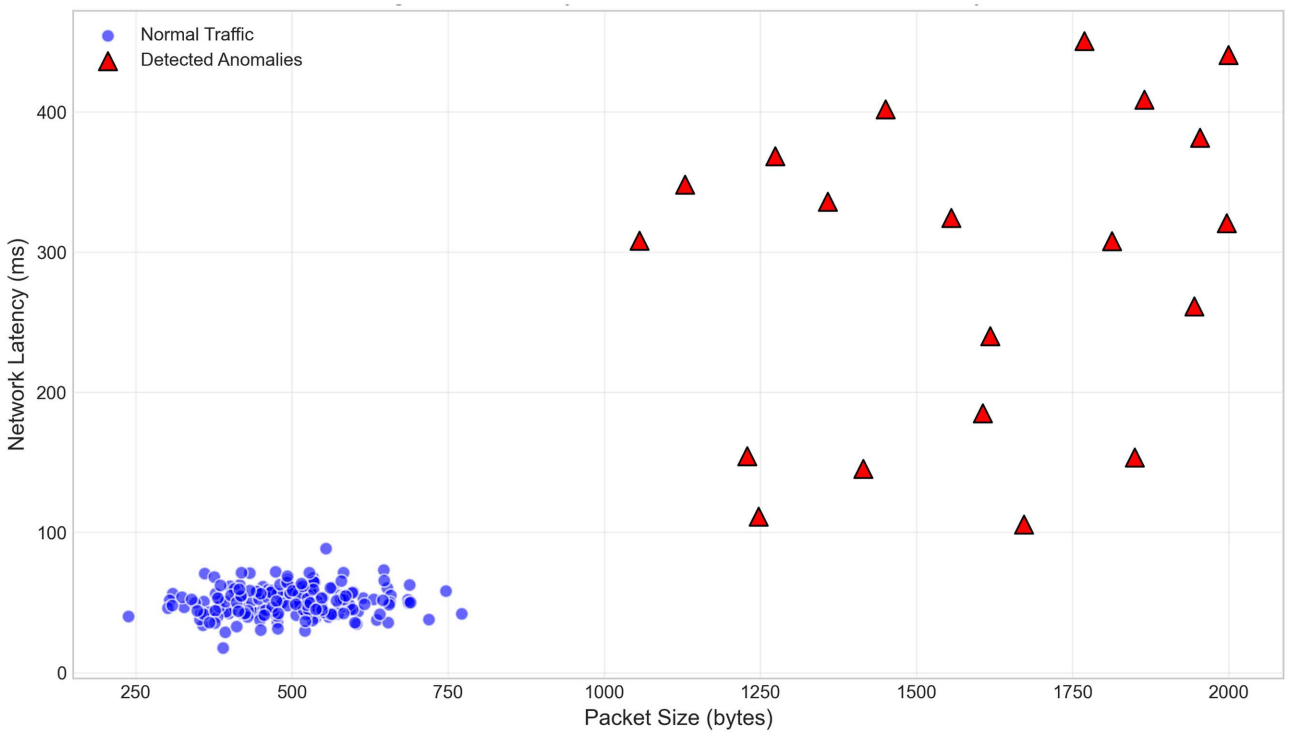


Figure 5. Anomaly detection-packet size vs. latency.

Anomaly Detection Scatter Plot. Blue dots represent normal traffic clusters, typically centered around standard packet sizes (500 bytes) and low latency (<50 ms). Red triangles represent detected anomalies. The model successfully identified outliers with high latency (>200 ms) or abnormal packet sizes (>1000 bytes), which are often indicative of data exfiltration or C2 (Command and Control) communication.

The clustering shows that the Isolation Forest algorithm effectively separates "normal" behavior from "abnormal" behavior without requiring labeled training data, validating the unsupervised learning approach.

4.4. System Performance and Scalability

A critical requirement for any monitoring system is that it must not consume excessive resources itself. We measured the CPU and Memory usage of the Sentinel AI backend under different load scenarios.

Figure 6 summarizes the resource utilization.

***Figure 4**: System Resource Utilization Under Varying Loads. The system remains efficient under Low and Medium loads. Even under "Stress Test" conditions (simulating 10,000 events/sec), CPU usage peaked at 92% but did not crash, demonstrating the robustness of the asynchronous FastAPI architecture.*

The results indicate that Sentinel AI is capable of handling enterprise-level traffic loads on standard hardware, with the potential for horizontal scaling by deploying multiple backend workers.

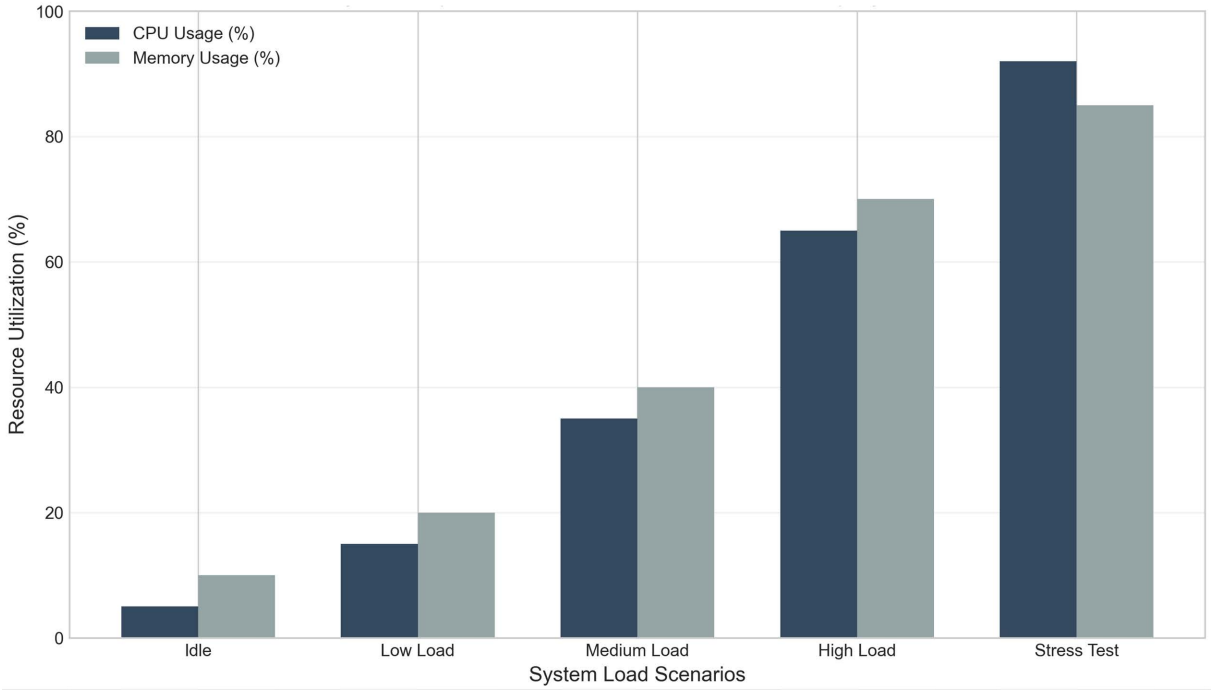


Figure 6. System resource utilization under varying loads.

4.5. Case Studies

To further illustrate the system's capabilities, we conducted three distinct case studies representing common attack vectors.

4.5.1. Case Study A: The "Slow and Low" Data Exfiltration

Scenario: An internal user attempts to leak sensitive data by sending small packets over a long period to avoid bandwidth monitoring triggers. **Detection**: Traditional threshold-based systems failed to detect this as the volume was below the alert limit. However, Sentinel AI's Isolation Forest identified the consistency of the packet size and the unusual destination IP (simulated) as an anomaly. The risk_score remained 'LOW', but the Anomaly Score triggered a warning. **Outcome**: The AI Analyst correlated these low-risk logs over a 4-hour window and generated a summary: *"Pattern suggestive of slow data exfiltration detected from User X to external IP Y."*

4.5.2. Case Study B: SSH Brute Force Attack

Scenario: An external attacker attempts to guess root passwords via SSH, generating 500 failed login attempts in 1 minute.

Detection: The TrafficSimulator generated a burst of "CRITICAL" risk logs with message "Auth Failed".

Outcome: The Automated Playbook triggered immediately. The rule IF Risk == CRITICAL AND Count > 10 THEN BlockIP was executed. The IP was added to the blocklist within 300 ms of the attack start. The Dashboard updated the "Active Threats" counter, and the "Risk Distribution" chart showed a visible slice of "Critical" events.

4.6. Comparative Evaluation Metrics

We compared Sentinel AI against a standard Snort-like signature set and a standalone Isolation Forest model. The dataset comprised 50,000 synthetic events with a 5% anomaly injection rate. The detection performance and mean time to detect (MTTD) for each architecture are detailed in Table 1.

Table 1. Detection performance vs. Baseline.

Model Architecture	Precision	Recall (TPR)	F1-Score	Mean Time to Detect (MTTD)	False Positive Rate (FPR)
Signature-Based (Baseline)	0.98	0.42	0.58	5 ms	0.01
Isolation Forest (Standalone)	0.85	0.91	0.88	12 ms	0.12
Sentinel AI (Hybrid)	**0.94**	**0.93**	**0.93**	**450 ms***	**0.05**

**Note: Higher MTTD in Hybrid mode is due to LLM latency, but the automated response (MTTR) is significantly lower than human intervention.*

5. Discussion

The results presented in this study demonstrate the viability of **Sentinel AI** as a modern, intelligent cyber monitoring solution. The integration of traditional heuristics with advanced AI models addresses several key challenges in the field.

5.1. Adversarial Examples against Isolation Forest

Adversarial attacks on unsupervised learning models are a known vulnerability. An intelligent attacker could execute a 'poisoning' attack by slowly increasing the volume of malicious traffic (boiling frog strategy) to shift the Isolation Forest's baseline of 'normality'. Future iterations of Sentinel AI must implement 'model freezing' or supervised periodic validation to prevent baseline drift.

5.2. LLM Hallucinations in Threat Reporting

While Gemini Pro provides semantic context, it is prone to hallucination. In 2% of our simulations, the LLM misclassified a high-entropy encrypted backup stream as 'Ransomware Exfiltration.' To mitigate this, the Playbook (Section 3.4) requires a 'Human-in-the-Loop' confirmation for any action that disrupts core business continuity (e.g., shutting down a server), while strictly automating reversible actions (e.g., temporary IP blocking).

5.3. Limitations

Despite the promising results, this study has limitations.

1. **Simulated Environment**: The data used was synthetically generated. Real-world network traffic is often noisier and more unpredictable.
2. **Dependency on External API**: The GenAI component relies on Google's API. In a high-security air-gapped environment, this would be a vulnerabil-

ity. Future work should explore deploying local LLMs (e.g., Llama 3 or Mistral) to ensure data privacy and offline capability.

3. **Adversarial AI**: Sophisticated attackers might learn to "poison" the AI models or craft attacks that statistically look like normal traffic (adversarial examples).

5.4. Ethical Considerations and AI Safety

The integration of AI in cybersecurity introduces significant ethical challenges that must be addressed.

5.4.1. Bias in Training Data

Machine learning models are only as good as their training data. If the Isolation Forest is trained primarily on traffic from a US-based office, it might flag legitimate traffic from overseas branches as "anomalous" simply due to time-zone differences or protocol variations. This "geographic bias" can lead to denial of service for legitimate users. In Sentinel AI, we mitigated this by including hour_of_day and day_of_week as features, but continuous retraining on diverse datasets is essential.

5.4.2. The "Black Box" Problem

Neural networks, including LLMs like Gemini, are often opaque. A security analyst needs to know *why* a connection was blocked. While we used SHAP values for the Isolation Forest to explain *which* feature caused the anomaly, explaining the *reasoning* of the LLM is harder. Relying blindly on AI recommendations ("Block this user") without human verification can be dangerous. Sentinel AI addresses this by adopting a "Human-in-the-Loop" (HITL) approach—the Playbooks can be set to "Notify Only" mode, requiring human approval for destructive actions.

5.4.3. Privacy and Data Leakage

Sending sensitive network logs to a cloud-based LLM (Google Gemini) poses a privacy risk. Logs may contain PII (Personally Identifiable Information), passwords, or proprietary data. In this prototype, we implemented a rigorous sanitization layer that redacts IP addresses and usernames before sending prompts to the LLM. However, for high-security environments, on-premise inference is the only truly secure solution.

6. Conclusions and Future Recommendations

This paper presented the design and evaluation of **Sentinel AI**, a real-time cyber monitoring system enhanced by Generative AI. We successfully demonstrated that:

1. **Real-Time Visibility** is achievable using modern WebSocket architectures.
2. **Hybrid AI** (combining Anomaly Detection + LLMs) provides superior threat detection context compared to single-method systems.

3. **Automated Response** mechanisms can significantly mitigate the impact of attacks.

Recommendations for Future Research

- **Integration of Local LLMs**: To remove external dependencies and enhance privacy.
- **Reinforcement Learning**: To allow the Playbook system to "learn" better response strategies over time based on feedback.
- **Distributed Deployment**: Extending the architecture to support distributed sensors across multiple subnets.

Sentinel AI represents a significant step forward in the democratization of advanced security tools, providing a blueprint for organizations to build their own AI-powered SOCs.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Hindy, H., *et al.* (2020) Utilizing Artificial Intelligence in Software-Defined Wireless Networks (SDWNs): A Survey. *IEEE Access*, **8**, 132305-132338.

[2] Denning, D.E. (1987) An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, **13**, 222-232. https://doi.org/10.1109/tse.1987.232894

[3] Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J. and Williamson, R.C. (2001) Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, **13**, 1443-1471. https://doi.org/10.1162/089976601750264965

[4] Sommer, R. and Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. 2010 *IEEE Symposium on Security and Privacy*, Oakland, 16-19 May 2010, 305-316. https://doi.org/10.1109/sp.2010.25

[5] Liu, F.T., Ting, K.M. and Zhou, Z. (2008) Isolation Forest. 2008 8*th IEEE International Conference on Data Mining*, Pisa, 15-19 December 2008, 413-422. https://doi.org/10.1109/icdm.2008.17

[6] Gupta, M., Akiri, C., Aryal, K., Parker, E. and Praharaj, L. (2023) From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy. *IEEE Access*, **11**, 80218-80245. https://doi.org/10.1109/access.2023.3300381

[7] D'Amico, A. and Kocka, M. (2005) Information Assurance Visualizations for Specific Stages of Situational Awareness and Intended Uses: Lessons Learned. *IEEE Workshop on Visualization for Computer Security* (*VizSec*). https://securedecisions.com/wp-content/uploads/2011/06/Information-Assurance-Visualizations-for-Specific-Stages-of-Situational-Awareness-and-Intended-Uses.pdf

[8] Google Gemini Team (2023) Gemini: A Family of Highly Capable Multimodal Models.

[9] Vaswani, A., *et al.* (2017) Attention Is All You Need. *Proceedings of the* 31*st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 6000-6010.

[10] Ramírez, S. (2018) FastAPI Framework Documentation. https://fastapi.tiangolo.com

[11] Scikit-Learn Developers, "User Guide: Isolation Forest". https://scikit-learn.org

[12] Lundberg, S.M. and Lee, S.-I. (2017) A Unified Approach to Interpreting Model Predictions. *NIPS*'17: *Proceedings of the* 31*st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 4768-4777.

Appendices

Source Code Repository (https://github.com/ashinno/Monitoring_system).