Autonomous AI SecOps Log Analyzer: An Evidence-Grounded Agentic Framework for Security Incident Detection, Correlation and Threat Prioritization

Darshan Nikrad[1], Yash Nimbalkar[2], Sainath Pawar[3], Atharva Khatawkar[4], Prof. Madhuri Mane[5]
[1,2,3,4]*Dept. of Computer Engineering, Pune Institute of Computer Technology Pune, India*
[5]*Project Guide, Dept. of Computer Engineering, Pune Institute of Computer Technology Pune, India*

***Abstract*—Security Operations Centres (SOCs) receive far more alerts than analysts can investigate. Rule-based monitoring yields many false positives, machine-learning anomaly detectors often return scores without explanation, and Large Language Model (LLM) assistants can produce plausible but unsupported conclusions. This paper presents the Autonomous AI SecOps Log Analyzer, an evidence-grounded agentic framework that ingests network and application logs, detects suspicious activity with eight rule-based detectors and an Isolation Forest, correlates detections into incidents using graph connected components, prioritizes them into tiers P1–P4, and investigates each incident with a tool-restricted agent. The agent is limited to seven read-only tools and 20 tool calls, tests malicious and benign hypotheses, defends against prompt injection hidden in logs, and rejects any conclusion not linked to verifiable source events. The system runs fully offline, with optional cloud or local LLMs. A five-condition ablation study on a deterministic synthetic dataset of 910 events (810 effective records) shows that correlation reduces 28 detections to 15 incidents (46%) and that all findings are evidence-attributed (100% attribution, 0% unsupported claims). Detection quality is modest (F1 = 0.356, FPR = 34.2%) and the data are synthetic, so the results are preliminary. The system is a decision-support tool and takes no enforcement action.**

***Index Terms*—Security operations centre, log analysis, anomaly detection, event correlation, agentic AI, evidence-grounded investigation, threat prioritization, Isolation Forest.**

I. INTRODUCTION

Security Operations Centres (SOCs) monitor logs from servers, applications and network devices, detect suspicious activity, investigate it and respond. Telemetry volume has grown with cloud infrastructure and remote work, and industry studies report that a typical SOC receives on the order of eleven thousand alerts per day [1], far more than analysts can examine. The result is alert fatigue: genuine threats are buried among false positives and low-priority notifications.

Existing tools address only part of the problem. Rule-based correlation is deterministic and explainable, but it produces many false positives and needs constant manual tuning [2].

Machine-learning detectors such as Isolation Forest can surface previously unseen behaviour [3], [4], but a bare anomaly score gives the analyst little to act on. LLM-based assistants offer natural-language investigation, yet language models can hallucinate indicators of compromise or conclusions absent from the data [5], [6], which is especially dangerous in security work. Logs may also carry prompt-injection text, for example in a User-Agent header or username, that can mislead an unprotected agent.

This paper studies whether these approaches can be combined in a single transparent pipeline.

The contributions are:

- A pipeline of eight explainable rule-based detectors and an Isolation Forest detector whose findings carry measured values, thresholds and source event identifiers;
- Graph-based correlation of detections into incidents, each with a stored grouping explanation, and a transparent multi-factor priority score (P1–P4);
- A tool-restricted investigation agent with seven read-only tools, dual-hypothesis testing, prompt-injection defences and mandatory evidence attribution; and
- A reproducible five-condition ablation study that quantifies the contribution of each layer.

The remainder of the paper is organized as follows. Section II reviews related work, Section III describes the framework, Section IV the experimental setup, Section V the results, Section VI the limitations, and Section VII concludes.

II. RELATED WORK

A. Rule-Based Detection and Threat Taxonomies

MITRE ATT&CK [7] and NIST SP 800-61 [8] provide a taxonomy of adversary behaviour and an incident-handling life cycle that structure the threat categories and workflow used here. Signature and rule-based systems are explainable but suffer from high false-positive rates and weak coverage of novel behaviour [2].

B. Statistical and Machine-Learning Anomaly Detection

Chandola et al. [3] survey statistical, classification, clustering and information-theoretic anomaly detection. Isolation Forest [4] isolates anomalies by random partitioning without profiling normal data and is used here as the unsupervised detector. Autoencoder-based methods [9] need substantial training data and offer limited interpretability, which motivated the choice of a tree-based detector.

C. Alert Correlation

Julisch [2] clusters alarms to support root-cause analysis, Valeur et al. [10] combine scenario, statistical and temporal correlation, and Ning et al. [11] use attack-graph style relationships among alerts. Naive time-window merging can join unrelated events; the correlation engine here uses explicit rules and is tested to never merge unrelated events.

D. LLM Assistants and Tool-Using Agents

ReAct [12] interleaves reasoning with tool actions and Toolformer [13] shows that tool use can improve factual grounding. Commercial assistants such as Security Copilot [6] apply LLMs to incident investigation. Hallucination remains a documented risk [5], motivating the mandatory evidence attribution and tool allowlist adopted in this work.

E. Research Gap

Prior work addresses detection, correlation or LLM assistance largely in isolation. Few designs combine explainable detection, graph correlation, transparent prioritization and an agent whose every conclusion is verified against source events, while also operating fully offline and defending against adversarial log content. This paper targets that gap.

III. PROPOSED FRAMEWORK

The framework is a staged pipeline (Fig. 1). Logs are ingested and normalized, passed through the detection engine, correlated into incidents, prioritized and investigated; results are stored in a database and presented through reports and a dashboard.

The layers are decoupled: the dashboard communicates with the backend only through a REST API, and analytical modules exchange typed domain objects.

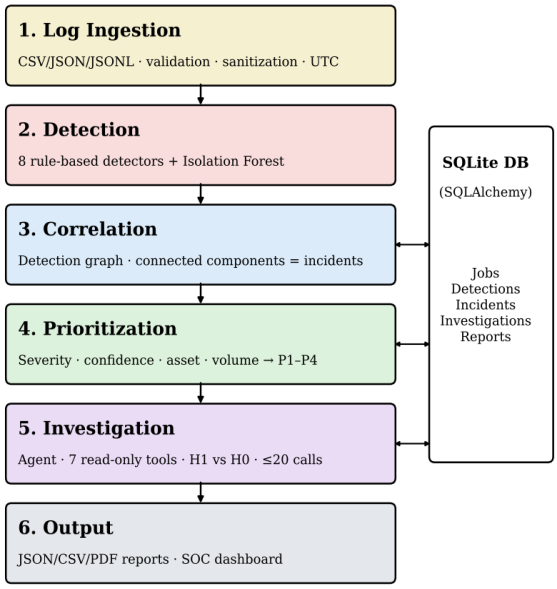


Fig. 1. Architecture of the proposed AI SecOps Log Analyzer.

A. Log Ingestion and Normalization

The ingestion module accepts CSV, JSON (array or envelope) and JSONL files up to 100 MB. Timestamps in ISO 8601, Unix epoch, RFC 2822 and common log formats are normalized to UTC; IP addresses and ports are validated; numeric fields are safely coerced; and text is sanitized to remove null bytes and control or escape sequences.

Duplicates are identified by content hash. Malformed records are reported with line numbers and reasons and are never silently dropped or fabricated.

B. Detection Engine

Let L be the set of normalized records. Each detector maps L to a set of detections D. The eight rule-based detectors cover request flooding (DoS), brute-force authentication, port scanning, HTTP status anomalies, path traversal, SQL/command injection [15], data exfiltration, and authentication anomalies (impossible travel, multi-IP access, off-hours). A ninth, statistical detector applies an Isolation Forest [4], [14] over multiple features. Every detection records its type, severity, confidence, description, measured and threshold values, and source event identifiers.

Baselines use the median, median absolute deviation (MAD), percentiles and interquartile-range (IQR) fencing, which resist contamination by outliers. A value x is flagged against its baseline using the Z-score and the IQR fence:

$$z = (x - \mu) / \sigma , \quad x > Q_3 + 1.5 \cdot \mathrm{IQR} \qquad (1)$$

C. Graph-Based Correlation

Detections are nodes of a graph $G = (D, E)$. An edge joins two detections that share a source address, share a target, belong to the same attack chain, or fall within a temporal window Δt.

The incidents are the connected components of G, and each incident stores an explanation of why its detections were grouped.

D. Threat Prioritization

Each incident I receives a transparent score $S(I)$ in [0, 100]:

$$S(I) = w_s \cdot S_{base} + w_c \cdot C_{conf} + w_a \cdot A_{asset} + w_v \cdot V_{vol} \qquad (2)$$

S_{base} is the base severity (Critical 40, High 30, Medium 20, Low 10), C_{conf} is the mean confidence of the component detections, A_{asset} reflects target-asset criticality and V_{vol} reflects event volume and affected hosts. The score maps to tiers P1_CRITICAL, P2_HIGH, P3_MEDIUM and P4_LOW.

E. Evidence-Grounded Investigation Agent

The agent investigates an incident using only seven allow-listed, read-only tools: *get_incident_details*, *get_related_events*, *get_source_activity_summary*, *get_detection_evidence*, *get_baseline_comparison*, *search_similar_incidents* and *generate_recommendations*. It tests a malicious hypothesis H_1 against a benign hypothesis H_0, makes at most K_{max} = 20 tool calls, and each call returns at most 50 events.

Safety controls are:

- *Tool allowlist:* no write, shell or database-modification capability.
- *Injection defence:* log text is sanitized, screened for injection patterns and isolated as untrusted data before reaching an LLM.
- *Evidence attribution:* a conclusion is accepted only if every cited event identifier exists in the analysed data; otherwise, it is rejected.

With no API key the agent runs a deterministic offline workflow; OpenAI, Anthropic, Google or local LLM providers are optional. Outputs are decision support only, and the system never blocks traffic or takes enforcement action.

F. Implementation

The prototype is written in Python 3.11 using FastAPI for a versioned REST API, Pydantic for validation, pandas and NumPy for data processing, scikit-learn for Isolation Forest, SQLAlchemy 2.0 with SQLite for persistence, and Streamlit with Plotly for a nine-view SOC dashboard. Reports are exported as JSON, CSV and PDF. A GPU is not required.

IV. EXPERIMENTAL SETUP

A. Dataset

A deterministic synthetic dataset of 910 events is generated with a fixed random seed (42): 500 normal events and attack scenarios for DoS (50), brute force (60), port scan (100), path traversal (50), SQL injection (50) and data exfiltration (100). Ground-truth labels are stored in a separate file that the pipeline never reads. Port-scan records with HTTP status code 0 are rejected by validation, leaving 810 effective records.

B. Ablation Conditions

Five conditions isolate each layer: (A) rule-based detection only; (B) A plus the Isolation Forest; (C) B plus correlation and deterministic investigation; (D) B plus correlation and LLM investigation (simulated, since no API key was used); and (E) the complete agentic framework.

C. Metrics

Detection quality is measured by precision P = TP/(TP+FP), recall R = TP/(TP+FN), $F1$ = $2PR/(P+R)$, and false-positive rate FPR =

FP/(FP+TN). Workload reduction is the fall in items to review from detections to incidents. Evidence-attribution accuracy is the share of findings whose cited event identifiers are all verified in the data, and the unsupported-claim rate is the share of claims lacking verifiable identifiers.

V. RESULTS

Table I gives the ablation results; all prototype tests pass (95 tests: 52-unit, 10 integration and 33 security).

TABLE I ABLATION RESULTS ON 910 SYNTHETIC EVENTS (810 EFFECTIVE)

Cond.	Components	Prec.	Recall	F1	FPR
A	Rule-based detection only	0.419	0.295	0.346	0.336
B	A + Isolation Forest	0.424	0.307	0.356	0.342
C	B + correlation + deterministic investigation	0.424	0.307	0.356	0.342
D	B + correlation + LLM investigation (simulated)	0.424	0.307	0.356	0.342
E	Complete agentic framework	0.424	0.307	0.356	0.342

Evidence-attribution accuracy was 100% in all conditions.

Adding the Isolation Forest (A→B) raised recall by 1.2 percentage points (0.295 to 0.307) and F1 from 0.346 to 0.356, at the cost of a slightly higher false-positive rate (0.336 to 0.342) and higher latency (49 ms to 1,129 ms in the reported run). Conditions C–E share identical detection metrics because the investigation layer does not alter detection outputs, and without an LLM key D and E fall back to the deterministic provider. Graph-based correlation reduced 28 detections to 15 incidents, a 46% reduction in items an analyst must review. The agent made 153 tool calls over the 15 incidents (about ten per incident), within the limit of 20, with 100% evidence-attribution accuracy and a 0% unsupported-claim rate.

VI. DISCUSSION AND LIMITATIONS

The results support the architectural claims (workload reduction and full evidence traceability) but not strong detection claims.

The main limitations are:

- the dataset is synthetic, so results may not transfer to real traffic;
- detection quality is modest (F1 = 0.356) with a 34.2% false-positive rate, so thresholds need tuning;
- conditions D and E do not yet show LLM-specific gains, since the deterministic provider was used;
- 100 port-scan records were rejected by validation, reducing the dataset from 910 to 810 records;
- thresholds are static and the Isolation Forest is fitted on every run, adding latency; and
- findings are supporting evidence and do not prove that an event is malicious.

Pipeline complexity is polynomial: parsing is linear, rule detectors cost O(n) to O (n log n), Isolation Forest scoring costs O ($n \cdot t \cdot \log \psi$) for t trees and sub-sample size ψ, and correlation needs at most $O(|D|^2)$ pair checks, so the problem is in class P. No NP-hardness is claimed for attack detection in general.

VII. CONCLUSION AND FUTURE WORK

This paper presented an evidence-grounded agentic framework for security log analysis that combines explainable and statistical detection, graph-based correlation, transparent prioritization and a controlled investigation agent.

The preliminary evaluation shows a 46% workload reduction and fully attributed findings, while detection accuracy is modest and measured only on synthetic data. Future work includes: evaluation on public datasets such as CICIDS and UNSW-NB15 and on real SOC logs; adaptive, time-aware thresholds; evaluation of real LLM providers on investigation quality and hallucination rate; streaming ingestion and pre-trained models to reduce latency; analyst-approved SOAR integration; multi-tenant deployment with PostgreSQL and role-based access control; and usability testing with SOC analysts.

ACKNOWLEDGMENT

The authors thank their guide, Prof. Madhuri Mane, and the Department of Computer Engineering, Pune Institute of Computer Technology, for their guidance and support.

REFERENCES

[1] Ponemon Institute, "The economics of security operations centers: What is the true cost for effective results?" Research Rep., 2023.

[2] K. Julisch, "Clustering intrusion detection alarms to support root cause analysis," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 4, pp. 443–471, 2003.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[4] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 2008, pp. 413–422.

[5] L. Huang et al., "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," arXiv:2311.05232, 2023.

[6] Microsoft Corporation, "Microsoft Security Copilot," 2023. [Online]. Available: https://www.microsoft.com/en-us/security/business/ai-machine-learning/microsoft-security-copilot

[7] MITRE Corporation, "MITRE ATT&CK: A globally accessible knowledge base of adversary tactics and techniques," 2024. [Online]. Available: https://attack.mitre.org/

[8] P. Cichonski, T. Millar, T. Grance, and K. Scarfone, "Computer security incident handling guide," NIST Special Publication 800-61, Rev. 2, 2012.

[9] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in Proc. 2nd Workshop Mach. Learn. Sensory Data Anal., 2014.

[10] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "A comprehensive approach to intrusion detection alert correlation," IEEE Trans. Dependable Secure Comput., vol. 1, no. 3, pp. 146–169, 2004.

[11] P. Ning, Y. Cui, D. S. Reeves, and D. Xu, "Techniques and tools for analyzing intrusion alerts," ACM Trans. Inf. Syst. Secur., vol. 7, no. 2, pp. 274–318, 2004.

[12] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in Proc. Int. Conf. Learn. Represent. (ICLR), 2023.

[13] T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

[15] OWASP Foundation, "OWASP Top 10–2021: The ten most critical web application security risks," 2021. [Online]. Available: https://owasp.org/Top10/