WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

AI-BASED SOC ASSISTANT FOR EVENT CORRELATION, LOG ANALYSIS, AND THREAT DETECTION

Tamada Mohan Krishna[1], Pullari Sailaja[2], Seeramreddi Gowtham Kumar[3], Ponnana Dilli Babu[4], M Harshavardhan[5]

[1] *Computer Science and Engineering GMRIT Deemed to be University Rajam, Andhra Pradesh, India*
[2] *Computer Science and Engineering GMRIT Deemed to be University Rajam, Andhra Pradesh, India*
[3] *Computer Science and Engineering GMRIT Deemed to be University Rajam, Andhra Pradesh, India*
[4] *Computer Science and Engineering GMRIT Deemed to be University Rajam, Andhra Pradesh, India*
[5] *Computer Science and Engineering GMRIT Deemed to be University Rajam, Andhra Pradesh, India*

ABSTRACT:

Security Operations Centers (SOCs) process large volumes of endpoint, network, identity, and application telemetry. This paper presents an AI-Based SOC Assistant that combines telemetry ingestion and normalization, temporal event correlation, deterministic threat detection, incident management, AI-assisted investigation, and analyst-facing presentation. The project architecture identifies Splunk Boss of the SOC (BOTS) v1 CSV data (15.8 GB and approximately 13.4 million rows) and a sample JSONL collection (1,508 events) as input sources; these are source-inventory figures and do not establish that the entire BOTS dataset was processed in a benchmark run. Events are parsed, validated using a Pedantic schema, and stored in SQLite. Four correlation strategies group activity by source IP, user, multiple destination ports, and authentication sequence, using configured sliding windows described in the technical dossier as 300–600 seconds. A deterministic engine specifies seven MITRE ATT&CK-mapped detection categories: brute force (T1110), suspicious successful logon (T1078), privilege escalation (T1068), suspicious process execution (T1059.001), port scanning (T1046), lateral movement (T1021), and unauthorized service installation (T1543.003). Related detections are grouped into incidents with severity and lifecycle tracking. A local Ollama model (Qwen 2.5 7B or Llama 3) supports evidence-based summaries and remediation guidance, with an expert fallback described in the project specification. The Stream lit dashboard presents Command Center, AI Workbench, and Telemetry Explorer views through a Fast API service. The technical dossier reports performance measurements, but they are reproduced here as reported values rather than independently verified results. Consequential response actions require authorization, safeguards, and analyst approval.

Keywords: Security Operations Center, SIEM, SOAR, Event Correlation, Log Analysis, Deterministic Threat Detection, Large Language Models, Ollama, MITRE ATT&CK, Incident Management.

I. INTRODUCTION

Modern organizations generate security telemetry across operating systems, identity services, firewalls, endpoint agents, cloud platforms, and applications. A single alert may not provide enough context to distinguish benign activity from an attack sequence. Security analysts therefore need a workflow that collects and normalizes records, links related activity, detects suspicious patterns, manages incidents, and presents traceable evidence. The project describes six functional layers: data sources; ingestion and normalization; analysis; incident management; AI intelligence; and presentation. The architecture identifies Splunk BOTS v1 CSV data (15.8 GB, approximately 13.4 million rows) and sample JSONL logs (1,508 events) as source inputs. These figures describe the input inventory in the architecture, not a verified processed-event count. The implementation view identifies Python, Fast API, Pedantic, SQLite, Ollama, and Stream lit as the principal technologies.

AI can assist SOC operations at different stages. Deterministic rules identify configured patterns; optional machine-learning models can classify events or score deviations only when trained and evaluated on suitable labeled data; and large language models (LLMs) can summarize retrieved evidence, explain why an alert may be suspicious, and draft reports. Workflow coordination can divide an investigation into intake, enrichment, event correlation, risk assessment, recommendation, and reporting tasks. These components must remain distinguishable: an LLM explanation is not proof that an attack occurred, and an anomaly score is not automatically a calibrated probability.

This paper describes the project architecture and its documented implementation design for evidence-backed security investigations. It focuses on log analysis, event correlation, and deterministic threat detection, with AI used to support analyst understanding and decision-making. The paper distinguishes implemented system components from architecture-specified or benchmark-reported capabilities. Real response actions remain subject to authorization, safety checks, and analyst approval.

II. PROBLEM STATEMENT AND OBJECTIVES

The central problem is the time and effort required to triage large numbers of security events and assemble evidence from different sources. Analysts must determine whether an event is suspicious, identify affected assets, correlate related records, estimate severity, and choose a response. Manual processes can be inconsistent and difficult to audit.

The project objectives are: (1) ingest and normalize security events; (2) detect suspicious activity using transparent rules and optional trained models; (3) automatically create alert and investigation records; (4) coordinate modules for enrichment, investigation, risk assessment, recommendations, and reporting; (5) maintain event timelines and audit logs; and (6) provide human approval for consequential response actions.

III. SOC, SIEM, AND AI INTEGRATION

A. Security Operations Center (SOC)

A SOC is the people, processes, and technology responsible for monitoring, investigating, and responding to cybersecurity incidents. SOC analysts examine alerts, validate evidence, assess impact, coordinate containment, and document lessons learned. The AI-Based SOC Assistant is designed as an analyst-assistance application rather than a replacement for the SOC team.

B. Security Information and Event Management (SIEM)

A SIEM receives data only from sources configured to send or expose logs. Typical sources include operating-system audit logs, identity providers, firewalls, endpoint detection and response (EDR) products, DNS services, cloud audit trails, and application logs. Collectors forward available events to the SIEM or application, where parsers normalize fields, rules correlate events, and indexed storage supports investigation. If a device or service is not connected or does not emit the relevant event, the system cannot reliably analyze that activity.

C. Where AI Is Used

AI is placed after ingestion and normalization in the documented workflow. The deterministic rule engine evaluates known patterns. Supervised classifiers or anomaly models are optional extensions and require representative labeled data, a defined feature schema, and reproducible evaluation. The local LLM supports explanation and report drafting from retrieved incident evidence. The workflow manager routes tasks and validates outputs. The paper does not treat optional ML models as evaluated project components unless implementation and test evidence are supplied.

Component	Purpose	Input	Output
SIEM / log connector	Collect and index supported logs	Source events	Searchable event records
Rule engine	Detect known patterns	Normalized fields and thresholds	Rule match and evidence
ML classifier (optional)	Classify trained event patterns	Features matching training schema	Predicted class / score
Anomaly model (optional)	Flag unusual records	Numeric/categorical features	Anomaly score
LLM assistant	Explain and summarize evidence	Retrieved event context	Draft explanation/report
module orchestrator	Sequence specialized tasks	Alert and module results	Workflow state and audit trail

IV. RELATED WORK AND RESEARCH GAP

Prior research addresses security-event correlation, log analysis, LLM-assisted cybersecurity, alert triage, and SOC automation. Kotenko, Gaifulina, and Zelichenok [1] review security-event correlation methods. Palma et al. [2] examine large language models for scalable and explainable cybersecurity log analysis, while Vaarandi and Bahşi [3] study LLM-based template detection from security event logs. Ismail et al. [4] investigate a retrieval-augmented copilot for Wazuh security-event response. Singh et al. [5] study human–AI collaboration in SOC operations, and Srinivas et al. [6] survey LLMs and agents for security automation. Kazi and Poinkar [7] discuss explainable alert triage, while Garware and Zisad [8] present parameter-efficient LLM log analysis. A dashboard-oriented BDR design is discussed in [9], and Zhang et al. [10] describe a task-driven framework for verifiable LLM-powered SOC analysis. Maosa et al. [11] examine hierarchical security-event correlation.

Kotenko, Gaifulina, and Zelichenok [1] review 127 publications on security-event correlation and classify approaches by correlation method, knowledge extraction, data sources, architecture, and evaluation strategy. Their review shows that event correlation is not a single algorithm: approaches may use rules, temporal relationships, statistical features, attack scenarios, or knowledge-based representations. This is relevant to an assistant that connects login events, endpoint events, and network observations while preserving a traceable explanation of why records were grouped.

Palma et al. [2] investigate the use of large language models for scalable and explainable cybersecurity log analysis. This work is relevant to the analyst-assistance layer, where an LLM can convert retrieved technical evidence into a readable explanation. In the present project, language generation is separated from deterministic detection: the model may summarize a detection record, but the triggering rule and source events remain the authoritative evidence.

Vaarandi and Bahşi [3] study the use of large language models for template detection from security event logs. Their work is relevant to the challenge of interpreting heterogeneous log messages. The project therefore uses a normalization stage to map available source fields into a common event representation and should preserve the original record and parsing metadata so an analyst can verify any transformation.

Recent SOC research also considers retrieval-augmented response and human–AI collaboration. Ismail et al. [4] describe a Wazuh security-event response copilot driven by retrieval-augmented generation, while Singh et al. [5] report an empirical study of human–AI collaboration in SOCs. Srinivas et al. [6] survey LLMs and agents for security automation. These works motivate a workflow in which model-generated investigation assistance is connected to retrieved evidence and remains subject to analyst review.

Explainable alert triage is addressed by Kazi and Poinkar [7], and parameter-efficient LLM log analysis is explored by Garware and Zisad [8]. The dashboard-oriented BDR design in [9] provides additional context for presenting security events and response state. Zhang et al. [10] emphasize task-driven analysis and verifiable insights, supporting the design principle that an LLM should summarize structured telemetry rather than invent events or act as the sole detector. Maosa et al. [11] provide further background on hierarchical event correlation for real-time threat detection and response.

The research gap addressed by this project is primarily one of practical integration and traceability. Existing research examines correlation, alert filtering, log anomaly detection, and LLM-assisted investigation, but a student prototype still needs a clearly defined pipeline that joins these functions, records why an alert was generated, and exposes the evidence to a human analyst. The proposed contribution is a modular SOC assistant design centered on event correlation, log analysis, and threat detection. It is not presented as a new state-of-the-art detection algorithm, and its effectiveness must be demonstrated through reproducible experiments.

The proposed framework integrates established roles into an auditable SOC workflow rather than claiming a new state-of-the-art detection algorithm. It groups related records, retains event-level evidence, and uses AI to produce analyst-oriented explanations. The literature identifies event correlation as an established research area [1], [11], while recent work explores LLM-based log analysis, template detection, and SOC assistance [2]– [6], [8], [10]. Explainable alert triage and dashboard presentation are also relevant to analyst usability [7], [9]. The contribution of this project is the integration design and evidence-grounded workflow; its effectiveness must be judged from reproducible tests and documented operating conditions.

Aspect	**Common limitation**	**the AI-based SOC assistant design response**
Alert triage	High volume and fragmented context	Normalize events and group related evidence
Detection	Rules may be narrow; ML depends on data quality	Use rules as baseline; add validated models where available
Investigation	Manual correlation takes time	Investigation module builds a traceable timeline
Threat context	Indicators may lack context	Threat-intelligence module enriches configured indicators
Reporting	Reports may be inconsistent	Generate structured incident summary with evidence links
Response	Automated actions can be risky	Require approval and audit response actions

V. PROPOSED SYSTEM ARCHITECTURE

The architecture shown in Fig. 1 describes six functional layers: data sources; ingestion and normalization; analysis; incident management; AI intelligence; and presentation. The source inventory identifies Splunk BOTS v1 CSV data (15.8 GB and approximately 13.4 million rows) and sample JSONL logs (1,508 events); these figures are not evidence that all records were ingested in a single run. The ingestion path uses a collector interface (Base Collector and SampleLogCollector), a parser/normalizer with Pedantic validation, and SQLite storage (soc_data.db). The analysis layer contains an event-correlation engine and a deterministic threat-detection engine. Four correlation strategies are specified: source-IP activity, user activity, multi-port network behavior, and authentication sequences. The technical dossier describes 300–600 second sliding windows. Seven rule categories are mapped to MITRE ATT&CK: brute force (T1110), suspicious successful logon (T1078), privilege escalation (T1068), suspicious process execution (T1059.001), port scan (T1046), lateral movement (T1021), and unauthorized service installation (T1543.003). The incident layer groups detections by entities such as source IP or username, assigns severity from LOW through CRITICAL, and tracks NEW, INVESTIGATING, CONTAINED, and CLOSED states. The AI layer uses a local Ollama model, with Qwen 2.5 7B or Llama 3 identified in the project materials, and an expert fallback. The presentation layer uses Stream lit views for Command Center, AI Workbench, and Telemetry Explorer, supported by a Fast API backend. The project report also presents a five-tier implementation hierarchy; this is a component-level implementation view and is distinct from the six functional layers in Fig. 1. Endpoint and test counts are omitted here because they require confirmation from the deployed source code and test output.

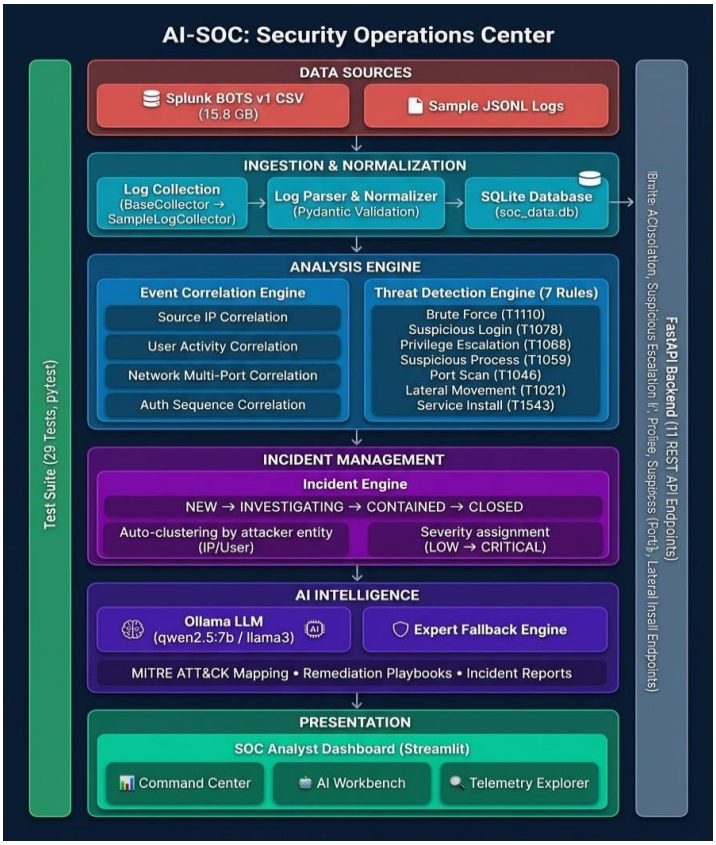


Fig. 1. Supplied architecture of the AI-SOC assistant, including ingestion, event correlation, rule-based threat detection, incident management, local LLM assistance, and dashboard presentation.

A. Event Sources and SIEM

For a prototype, the source may be a local JSON/CSV file or a controlled event simulator. For a live integration, the selected SIEM must be configured to collect the required sources and expose events through an API, webhook, query, or forwarding mechanism. The application should display whether an event is live, replayed, or simulated.

B. Alert and Incident Storage

An alert record should contain a unique alert ID, event timestamp, ingestion timestamp, source system, event type, severity, detection method, evidence, status, and links to related events. Incident records should track module results, analyst decisions, response approvals, and report versions. Raw events should be preserved where policy permits.

C. Dashboard

The dashboard can present summary metrics, new investigations, investigation history, a timeline, threat-intelligence context, risk assessment, recommendations, response plans, reports, and live module logs. A UI status such as 'Workflow Complete' should mean the configured workflow stages completed; it should not imply that the attack was conclusively confirmed unless the evidence supports that conclusion.

VI. functional modules and responsibilities

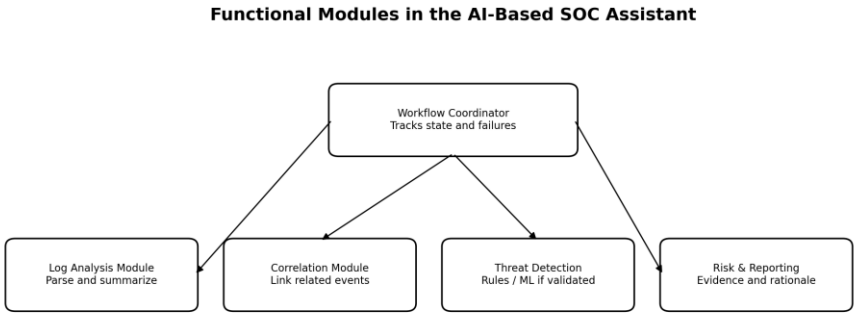


Fig. 2. Functional modules coordinated by the workflow manager.

module	Responsibilities	Expected structured output
Manager / Orchestrator	Route tasks, track stage status, handle failures	Workflow ID, stage state, timestamps
Intake module	Validate schema, summarize event, identify missing fields	Normalized incident summary
Detection module	Run configured rules/model and preserve evidence	Candidate label, score, rule/model ID
Threat Intelligence module	Look up IP/domain/hash in configured sources	Enrichment result, source, lookup time
Investigation module	Correlate related events and build timeline	Evidence list, sequence, unresolved questions
Risk Assessment module	Estimate severity using documented factors	Severity, rationale, affected assets
Recommendation module	Suggest safe investigation/containment options	Actions, rationale, prerequisites
Report module	Summarize evidence, decisions, and open questions	Incident report and audit references

The modules in the supplied architecture are cooperating software components, not necessarily independent AI agents. The collector interface isolates source-specific behavior; the parser and normalizer validate and standardize events; SQLite stores normalized records; the correlation engine groups events using four configured strategies; and the detection engine evaluates seven MITRE ATT&CK-mapped rules. The incident engine manages entity grouping, severity, and lifecycle state. The AI layer produces analyst-oriented summaries and remediation suggestions from incident evidence. Fast API connects backend services to the Stream lit dashboard. A five-tier implementation hierarchy is used in the project report to describe technology and responsibility at the component level, while the six-layer diagram presents the broader functional flow.

VII. METHODOLOGY AND END-TO-END WORKFLOW

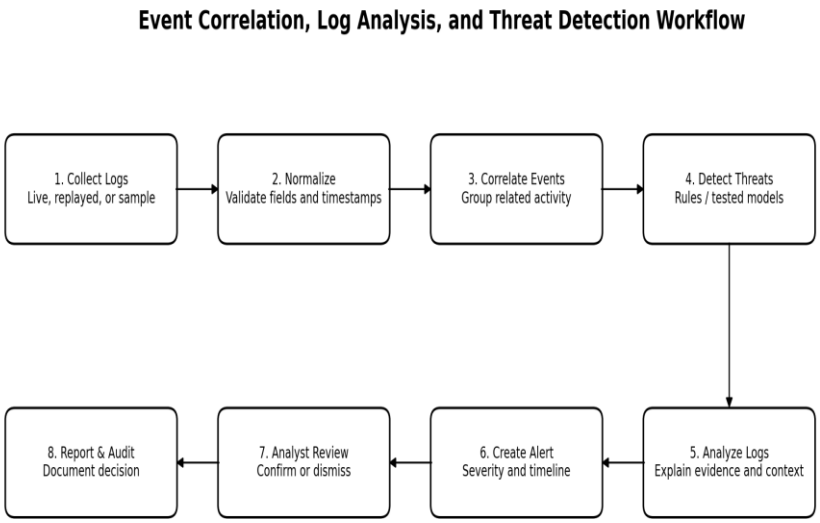


Fig. 3. Event correlation, log analysis, and threat-detection workflow.

The end-to-end workflow begins with collection from configured sources. The project materials identify Splunk BOTS v1 CSV data and sample JSONL logs as inputs. The collector abstraction separates source-specific reading from downstream parsing. The technical dossier also discusses Syslog, Windows Security Events, Linux Audited, Suricata IDS, and AWS CloudTrail as possible telemetry formats; these should be treated as supported or planned sources only where a connector is configured and tested. The collector should record ingestion time separately from the source timestamp because delayed delivery and clock skew can distort event ordering.

Normalization converts each source-specific record into a common schema. The documented Normalized Event model includes event_id, timestamp, source, source_ip, destination_ip, source_port, destination_port, username, event_type, action, status, severity, and message. Pedantic v2 is specified for schema validation; IP fields use IP-aware validation, and port values are constrained to 1–65535. Fields that are unavailable should remain null or be marked unavailable rather than fabricated. Timestamp parsing, IP validation, and required-field checks should generate quality flags. The original event and parsing metadata should be retained under an access-controlled policy so an analyst can inspect the source record when the normalized version is ambiguous.

Event correlation groups records that are likely to belong to the same activity. The project specifies four strategies: source-IP correlation, user-activity

correlation, multi-port sweep correlation, and authentication-sequence correlation. The technical dossier describes sliding temporal windows in the range of 300–600 seconds. For example, repeated authentication failures followed by a successful login for the same user or source may be grouped into one candidate incident. A more advanced implementation may use weighted scoring based on temporal proximity, shared identity, host relationship, and event sequence. Correlation thresholds must be validated because an overly broad window can combine unrelated events, while a narrow window can split one activity into separate alerts.

The documented detection engine uses deterministic rules for known patterns. The seven categories and their technique mappings are: brute force (T1110), suspicious successful logon following failures (T1078), privilege escalation (T1068), malicious LOLBin or process execution (T1059.001), network service port scan (T1046), lateral movement using explicit credentials (T1021), and unauthorized service installation (T1543.003). Machine-learning classifiers and anomaly models are optional extensions and should be described as implemented only if the final source code, trained model, feature schema, and evaluation results are available. A rule match indicates that a configured condition was met; it does not independently prove malicious intent.

The AI analysis component receives a structured incident object containing the detection reason, related event IDs, timestamps, relevant entities, rule version, and available context. The project materials identify local Ollama models such as Qwen 2.5 7B and Llama 3, with an expert fallback when the local model is unavailable. The component can generate an incident summary, explain the supplied evidence, identify missing context, and suggest investigation steps. To reduce unsupported output, factual statements should be linked to event IDs and the explanation should distinguish evidence from interpretation. If evidence is incomplete, the assistant should state that the activity is suspicious but unconfirmed. The model must not independently disable accounts, block addresses, or isolate hosts; consequential actions require documented authorization and approval.

A. Ingestion and Normalization

The ingestion layer accepts records from configured collectors or sample-event files. It validates required fields, normalizes timestamps and field names, assigns an event identifier where needed, and retains the original evidence. Invalid records should be rejected or quarantined with a reason rather than silently discarded. This separation of collection, normalization, and correlation is consistent with the need to handle heterogeneous security-event formats [1], [3], [11].

B. Detection Logic

The threat-detection engine described in the project report contains seven configured rule categories: brute force (MITRE ATT&CK T1110), suspicious successful logon (T1078), privilege escalation (T1068), malicious LOLBin or process execution (T1059.001), network service port scan (T1046), lateral movement using explicit credentials (T1021), and unauthorized service installation (T1543.003). The event-correlation engine separately groups source-IP activity, user activity, multi-port network behavior, and authentication sequences. The technical dossier specifies example trigger conditions, including at least five failed authentications from one source within 300 seconds for brute-force detection, and at least five distinct destination ports for port-scan detection. Thresholds and signatures must be checked against the final implementation; technique mappings alone do not prove that a real attack was detected.

C. Alert Creation and Logging

When a rule or model meets its configured alert condition, the backend writes a structured alert and records the detection reason. The event can be forwarded to a SIEM index if that integration is configured. Generated alerts should be tagged so they are not re-ingested as original source events, preventing event loops.

D. Investigation and Enrichment

After a detection rule fires, the incident engine groups related alerts by entities such as source IP or username, assigns severity from LOW through CRITICAL, and moves the case through NEW, INVESTIGATING, CONTAINED, and CLOSED states. The project specification describes severity propagation using the maximum severity among detections included in an incident. Incident transitions should record timestamps, actor, rationale, and supporting event IDs. Clustering should avoid merging unrelated users or hosts solely because they share an IP address; the correlation rationale must remain visible to analysts.

E. Risk and Recommendations

Risk assessment combines severity of the detection, asset criticality, evidence strength, and potential impact according to a documented policy. Recommendations should include the reason, evidence, expected effect, and potential operational impact. The report should state uncertainty and alternative explanations.

The system presents suggested actions to an analyst. The technical dossier describes SOAR actions such as BLOCK_IP, QUARANTINE_USER, ISOLATE_HOST, and TERMINATE_PROCESS, together with inverse rollback procedures where an inverse operation is possible. A demonstration should use simulation or a controlled lab unless live actions have been explicitly authorized. Real response requires permission checks, analyst approval, safeguards, an audit record, and a tested rollback plan. Process termination may be irreversible and should not be described as fully rollback able.

The system presents suggested actions to an analyst. For a demonstration, actions such as isolating a host or disabling an account should be simulated. A real action requires explicit approval, permission checks, safeguards, and an audit record. The final report contains the timeline, evidence, model/rule outputs, module decisions, analyst decision, and unresolved questions.

G. Illustrative Log Collection and AI-Assisted Analysis

The collection layer is specified for Splunk BOTS v1 CSV data and sample JSONL logs. The architecture identifies the BOTS v1 input as approximately 15.8 GB and 13.4 million rows, and the sample JSONL collection as 1,508 events. These values describe the input inventory and should not be interpreted

as the number of records successfully processed in the reported benchmark. The collector abstraction (Base Collector → SampleLogCollector) separates collection behavior from source format. The parser validates expected fields with Pedantic, normalizes timestamps and event attributes, and writes normalized records to SQLite (soc_data.db). Batch ingestion should record successful, skipped, and rejected records, together with processing time.

Table I presents illustrative sample records for demonstrating the processing path. These are synthetic examples created for explanation; they are not evidence of a real incident, and the IP addresses, accounts, hosts, and event counts must not be interpreted as live telemetry. A production deployment should preserve raw logs separately, redact unnecessary personal data, validate timestamps, and record parsing errors rather than silently discarding malformed events.

Event ID	Timestamp (UTC)	Source / Host	Illustrative event	Key fields
E-101	2026-10-20 08:41:02	Linux server / web-01	SSH authentication failed	src_ip=198.51.100.24; user=admin; result=failed
E-102	2026-10-20 08:41:08	Linux server / web-01	SSH authentication failed	src_ip=198.51.100.24; user=root; result=failed
E-103	2026-10-20 08:41:15	Linux server / web-01	SSH authentication failed	src_ip=198.51.100.24; user=guest; result=failed
E-104	2026-10-20 08:42:11	Identity service / id-01	Successful login after repeated failures	src_ip=198.51.100.24; user=admin; result=success
E-105	2026-10-20 08:43:00	Firewall / fw-01	Outbound connection observed	dst_ip=203.0.113.18; dst_port=443; action=allowed

During normalization, the parser maps source-specific field names to a common schema containing event_id, event_time, ingest_time, source, host, user, source_ip, destination_ip, event_type, outcome, and raw_event_reference. The correlation stage groups events that share a source IP, account, host, or other configured entity within a time window. For example, several failed logins followed by a successful login from the same source may be grouped into one investigation. Correlation indicates a relationship for investigation; it does not by itself prove that an account was compromised.

The detection layer evaluates transparent rules and, only were implemented and validated, machine-learning scores. A sample rule might flag repeated authentication failures from one source within a configurable interval and raise severity if a successful login follows. The AI analysis component then converts the correlated evidence into a concise explanation, highlights missing context, and suggests next investigative steps. The detection result and its evidence remain authoritative; generated text is an assistant explanation and must not invent facts absent from the logs.

H. Example Threat-Detection Output

Table II shows an example output that the dashboard could present after processing the synthetic events in Table I. Severity labels and thresholds are illustrative configuration values, not measured model performance. The event count in a real alert must be calculated from the actual correlated records, and the assistant should include event IDs and timestamps to support analyst verification.

Output field	Illustrative generated result
Alert ID	SOC-ALERT-0007 (sample only)
Detection label	Possible SSH brute-force activity followed by successful authentication
Severity	High (illustrative policy outcome)
Evidence	Repeated SSH failures from 198.51.100.24 for multiple usernames, followed by a successful login for admin; related event IDs E-101–E-104
Correlation rationale	Same source IP and host; events occur within a short time window; a success event follows repeated failures
AI-generated summary	The supplied sample records show repeated failed SSH authentication attempts from one source, followed by a successful login to the admin account. This sequence is suspicious and should be checked against expected administrator activity and approved source ranges.
Recommended analyst checks	Verify whether the source is authorized; inspect MFA and session details; review process and command history; search for related activity on other hosts; preserve relevant logs.
Suggested response	If confirmed unauthorized, follow the incident-response procedure to restrict the source or disable the affected account. Require analyst approval before taking disruptive action.

Confidence / limitations	Rule-triggered illustrative alert; no calibrated probability is claimed. Sample records alone cannot establish compromise.

The example illustrates the separation between event evidence, detection logic, and AI-generated explanation. The detection engine should produce a machine-readable alert object with a rule or model identifier, threshold, event references, and creation time. The language model may summarize that object but should not independently assign a factual attack conclusion without evidence. If required fields are absent or events conflict, the output should explicitly state the uncertainty and request further investigation.

For auditability, each investigation can retain the normalized events, correlation group, detection rule version, model version where applicable, prompt/template version for generated summaries, analyst disposition, and any approved response action. This supports repeatability and helps identify false positives, data-quality issues, and changes in detection behavior over time.

VIII. ALGORITHMS AND MODEL OPTIONS

Technique	Role in project	Important condition
Threshold / correlation rules	Detect known patterns such as repeated login failures	Thresholds must be configurable and documented
Logistic Regression / Random Forest / XGBoost	Candidate supervised event classifier	Train and evaluate on labeled data; avoid claiming use unless implemented
Isolation Forest	Candidate anomaly-scoring method	Score is not automatically a probability or attack label
LLM	Evidence-grounded summaries and report drafting	Constrain tools, cite event IDs, validate outputs
workflow coordination	Coordinate tasks and preserve workflow state	modules may be code modules, not separate ML models

The core detection approach described in the project report is deterministic rule evaluation combined with temporal event correlation. The LLM is used after incident construction for evidence summarization and analyst guidance. Supervised classifiers such as Logistic Regression, Random Forest, or XGBoost and anomaly methods such as Isolation Forest are optional research extensions; they are not presented as evaluated project components in this paper without corresponding training and test evidence.

IX. IMPLEMENTATION DETAILS

Component	Technology stated in project report	Responsibility
Telemetry ingestion hub	Python, Fast API, Pedantic v2	Reads configured telemetry, validates fields, and creates normalized events.
Correlation and detection	Python rule engine, SQLite (WAL)	Performs temporal grouping and evaluates seven deterministic detection rules.
AI triage and reasoning	Local Ollama (Qwen 2.5 7B / Llama 3), expert fallback	Summarizes incident evidence and prepares analyst-oriented explanations and playbooks.
SOAR hub	Python SOAR driver, SQLite	Describes configured containment actions, audit records, and rollback where supported.
SOC Command Center	Stream lit, Plotly, HTML/CSS	Presents incident queues, operational views, telemetry exploration, and reporting.

The presentation layer is a Stream lit SOC Analyst Dashboard with three named views: Command Center, AI Workbench, and Telemetry Explorer. The Command Center provides an overview of incidents and severity; the AI Workbench presents evidence-grounded summaries and remediation guidance; and the Telemetry Explorer supports filtering and inspection of normalized events. The backend uses Fast API to expose the configured application workflow. The project report's architecture also describes a five-tier implementation hierarchy, while the research paper's Fig. 1 groups the same system into six functional layers. The number of REST endpoints is not stated here because the available project documents differ and the final deployed route list has not been independently verified.

X. DATA MODEL AND EXAMPLE INCIDENT

The following example is illustrative and is not a claim that a real attack occurred. It demonstrates the fields that may be used by the intake and investigation pipeline.

Field	Example value
Event type	SSH authentication failures
Host	SRV-102
Source IP	185.XX.XX.50 (masked example)
Observed behavior	150 failed SSH logins within 5 minutes
Target account	admin
Detection basis	Configured threshold rule
Initial status	Candidate alert — needs investigation
Next steps	Review successful logins, source reputation, host logs, and account activity

A useful test should include benign login failures, authorized vulnerability scanning, brute-force-like bursts, and successful login events following failures. These scenarios help measure false positives and whether correlation improves triage. Use a controlled lab or synthetic records; do not generate attacks against systems without authorization.

A reproducible prototype should document the operating environment, log source, schema, event window, detection rules, model version, thresholds, and test procedure. A synthetic dataset is acceptable for demonstrating the pipeline, provided it is clearly labeled as synthetic and is not used to claim real-world detection performance. A stronger evaluation uses a documented public dataset or an authorized organizational dataset with ground-truth labels. The dataset description should include the number of records, attack and benign classes, time period, preprocessing steps, and any class imbalance.

Evaluation should measure both event-level and incident-level behavior. Precision indicates how many generated alerts are true positives among all alerts; recall indicates how many labeled threats were detected; F1-score balances precision and recall. False-positive rate estimates the proportion of benign events incorrectly flagged. Operational measures should include ingestion throughput, correlation latency, time from final evidence event to alert creation, and analyst-review time where measurable. Results should be compared against a simple baseline, such as individual rule alerts without correlation, to determine whether correlation actually improves investigation quality.

A useful ablation study compares (i) rules without correlation, (ii) rules plus event correlation, and (iii) rules plus correlation and AI-generated summaries. This helps distinguish detection improvements from usability improvements. For example, AI summarization may reduce the time required to understand an alert without changing the number of threats detected. The paper should report that distinction clearly. If the system has only been demonstrated with sample logs, the results section should describe functional checks and limitations rather than claim a measured precision, recall, or accuracy.

XI. EXPERIMENTAL SETUP AND EVALUATION

The technical dossier identifies Splunk BOTS v1 as the principal benchmark dataset and describes a separate benchmark run with 28,720 telemetry events. The architecture also lists a BOTS v1 source inventory of approximately 13.4 million rows and a separate sample JSONL collection of 1,508 events. These figures refer to different source descriptions and should not be added together or treated as one processed dataset. A reproducible evaluation should state which input subset was actually processed, its source and license, preprocessing steps, event counts, class distribution, and any ground-truth labels. The dossier's performance values are reproduced below as reported measurements; the available materials do not include the complete benchmark scripts and raw logs needed for independent verification.

Evaluation area	Metric	What it measures
Detection	Precision	Fraction of predicted alerts that are true positives
Detection	Recall	Fraction of true positive events detected
Detection	F1-score	Harmonic mean of precision and recall
False alerts	False-positive rate	Benign events incorrectly flagged
Latency	Event-to-alert time	Time from event receipt to alert persistence
Throughput	Events per second	Ingestion and processing capacity
Workflow	Stage completion rate	Proportion of investigations reaching intended end state
Reliability	Tool failure rate	Failed or timed-out module/tool calls

For binary detection, precision = TP/(TP+FP), recall = TP/(TP+FN), and F1 = 2PR/(P+R), where TP, FP, and FN denote true positives, false positives, and false negatives. The false-positive rate is FP/(FP+TN). Report class-wise results and a confusion matrix when labeled ground truth is available. For the modular workflow, define completion and failure criteria before testing. Processing throughput and latency should be measured with documented input sizes, hardware, software versions, and repeated runs.

The separate AI-SOC technical dossier reports throughput, detection, clustering, AI-generation, SOAR, rollback, end-to-end response, and test-suite measurements. These values are included in Table III as reported by that dossier, not as independently reproduced results in this paper. The supplied documents do not provide the full benchmark scripts, raw timing logs, hardware-specific run records, or test output needed to independently verify each value.

XII. RESULTS AND DISCUSSION

Table III summarizes the benchmark values stated in the technical dossier. They indicate the performance claimed for the described test environment, but they should not be generalized to other hardware, datasets, or production deployments. In particular, throughput and latency depend on input batch size, storage settings, event complexity, model availability, and the measurement boundaries used by the benchmark. The reported test-suite pass count is also a dossier statement and should be supported by the corresponding test output before publication.

Table III. Performance measurements reported in the AI-SOC technical dossier (not independently verified).

Metric	Value reported in technical dossier	Qualification
Log ingestion throughput	3,420 events/second	Reported benchmark; input batch and hardware details require reproduction.
Detection latency	8.2 milliseconds	Reported rule-execution latency; measurement boundary requires confirmation.
Incident clustering	124 milliseconds	Reported for correlation across 5,000+ detections.
AI investigation generation	1.8 seconds (Ollama) / 4 milliseconds (expert matrix)	Reported generation times; model, prompt, and hardware details require verification.
SOAR containment execution	18 milliseconds	Reported action-registration time; controlled execution evidence required.
SOAR rollback latency	12 milliseconds	Reported rollback time; action-specific verification required.
End-to-end response cycle	Less than 2.5 seconds	Reported end-to-end value; exact start/end events require documentation.
Test-suite pass rate	32/32 tests passed (100%)	Reported in dossier; attach the original test output to substantiate.

The intended benefit of the workflow is a more coherent incident view: the analyst can inspect the triggering rule, linked events, event timeline, severity, and a concise explanation in one investigation record. Correlation may reduce duplicate triage when several alerts describe related activity. AI-generated summaries can help analysts navigate unfamiliar log formats, but they introduce a separate reliability risk. Every summary should remain linked to source evidence, and generated explanations should be treated as decision support rather than authoritative proof. The benchmark figures in Table III are reported by the technical dossier and require independent reproduction.

Important failure modes include missing or delayed logs, clock skew, inconsistent usernames, NAT or shared IP addresses, incorrect field mapping, and poorly selected correlation windows. Machine-learning models may fail under class imbalance, distribution shift, or changes in attacker behavior. Language models may misinterpret ambiguous fields or produce unsupported claims. Mitigations include data-quality checks, versioned rules, evidence references, constrained prompts, analyst review, access controls, and periodic evaluation using updated datasets.

The current evidence base supports an architecture and a set of documented workflow and benchmark claims, but it does not establish general production performance. The benchmark values are reported in the supplied technical dossier and should be reproduced using retained scripts, test logs, hardware details, and dataset provenance. If a feature is demonstrated with sample or synthetic events, it should be identified as a functional demonstration rather than evidence of live operational detection.

XIII. SECURITY, ETHICS, AND LIMITATIONS

Security logs can contain usernames, IP addresses, hostnames, tokens, and other sensitive details. Apply access control, encryption in transit, retention limits, and redaction where appropriate. Treat log content as untrusted input: an attacker may place prompt-injection text in a username, URL, process argument, or log message. LLM prompts should clearly separate instructions from evidence, and the model should not be allowed to execute arbitrary commands.

Other limitations include incomplete log coverage, changing attack patterns, class imbalance, unreliable threat-intelligence sources, false positives, model drift, and LLM hallucinations. A system cannot detect an event that was never collected. A prototype using sample logs does not by itself demonstrate production SIEM integration. Automatic containment must not be enabled without approval, testing, rollback plans, and appropriate authorization. The reported benchmark results require reproducible evidence before they can be treated as independently validated.

XIV. CONCLUSION AND FUTURE WORK

This paper presents an AI-Based SOC Assistant that integrates telemetry ingestion and normalization, temporal event correlation, deterministic MITRE ATT&CK-mapped threat detection, incident management, AI-assisted investigation, and analyst-facing presentation. The project materials identify Splunk BOTS v1 CSV and sample JSONL inputs, Pedantic-based validation, SQLite persistence, four correlation strategies, and seven rule categories. The system separates deterministic detection from LLM-based explanation and describes a hybrid local Ollama and expert-fallback analysis path. The Stream lit dashboard provides Command Center, AI Workbench, and Telemetry Explorer views through a Fast API backend. The technical dossier reports benchmark values for throughput, latency, AI investigation, SOAR actions, rollback, and test-suite execution; these values are reproduced as reported and require independent verification before being treated as validated results. Future work includes reproducible evaluation on labeled datasets, broader telemetry integration, stronger access controls, and systematic assessment of AI explanation quality and safety.

REFERENCES

[1] I. Kotenko, D. Gaifulina, and I. Zelichenok, "Systematic Literature Review of Security Event Correlation Methods," IEEE Access, vol. 10, pp. 43387–43420, 2022, doi: 10.1109/ACCESS.2022.3168976.

[2] G. Palma, G. Cecchi, M. Caronna, and A. Rizzo, "Leveraging Large Language Models for Scalable and Explainable Cybersecurity Log Analysis," Journal of Cybersecurity and Privacy, vol. 5, no. 3, Art. no. 55, 2025, doi: 10.3390/jcp5030055.

[3] R. Vaarandi and H. Bahşi, "Using Large Language Models for Template Detection from Security Event Logs," International Journal of Information Security, vol. 24, Art. no. 104, 2025, doi: 10.1007/s10207-025-01018-y.

[4] Ismail, R. Kurnia, F. Widyatama, I. M. Wibawa, Z. A. Brata, Ukasyah, G. A. Nelistiani, and H. Kim, "Enhancing Security Operations Center: Wazuh Security Event Response with Retrieval-Augmented-Generation-Driven Copilot," Sensors, vol. 25, no. 3, Art. no. 870, 2025, doi: 10.3390/s25030870.

[5] R. Singh, S. Tariq, F. Jalalvand, et al., "LLMs in the SOC: An Empirical Study of Human-AI Collaboration in Security Operations Centres," arXiv preprint arXiv:2508.18947, 2025. [Online]. Available: https://arxiv.org/abs/2508.18947.

[6] S. Srinivas, B. Kirk, J. Zendejas, M. Espino, M. Boskovich, A. Bari, K. Dajani, and N. Alzahrani, "AI-Augmented SOC: A Survey of LLMs and Agents for Security Automation," Journal of Cybersecurity and Privacy, vol. 5, no. 4, Art. no. 95, 2025, doi: 10.3390/jcp5040095.

[7] M. Kazi and S. Poinkar, "An Explainable AI Based Alert Triage Framework for Security Operations Centres: Architecture, Evaluation, and Analyst Centered Design," InfoTech Spectrum: Iraqi Journal of Data Science, vol. 3, no. 2, pp. 67–78, 2026, doi: 10.51173/ijds. v3i2.84.

[8] C. V. Garware and S. N. Zisad, "OpenSOC-AI: Democratizing Security Operations with Parameter Efficient LLM Log Analysis," arXiv preprint arXiv:2604.26217, 2026, doi: 10.48550/arXiv.2604.26217.

[9] "A Blockade-Detection-Response (BDR)-Based Security Operations Center Dashboard Design," Computers in Human Behavior Reports, vol. 4, Art. no. 100143, 2021, doi: 10.1016/j.chbr.2021.100143.

[10] H. Zhang, H. Wang, H. Tan, L. Zeng, J. Li, and Z. Gu, "From Ambiguous Queries to Verifiable Insights: A Task-Driven Framework for LLM-Powered SOC Analysis," IET Intelligence Technology, first published May 4, 2026, doi: 10.1049/cit2.70138.

[11] H. Maosa, K. Ouazzane, and M. C. Ghanem, "A Hierarchical Security Event Correlation Model for Real-Time Threat Detection and Response," Network, vol. 4, no. 1, pp. 68–90, 2024, doi: 10.3390/network4010004.