Supporting SOC Investigations with a Hybrid RAG-KG-assisted LLM System.

Zuhayr A. Mirza*
Eindhoven University of Technology

Abstract

Security Operations Center (SOC) analysts must interpret large volumes of alerts and fragmented evidence from logs, threat intelligence, and previous investigations. Although existing adversary models provide standardized terminology for describing attacker behaviour, they offer limited guidance for interpreting partially observed incidents. This paper presents a SOC AI Investigation Assistant that combines retrieval-augmented generation (RAG), a cybersecurity knowledge graph, and an analyst-facing phase framework. The system retrieves cyber threat intelligence and historical case knowledge, relates it to current case evidence, and generates structured investigative guidance. This prototype is integrated into Security Onion's case-management workflow. The prototype was evaluated through an assessment of generated responses and a preliminary user study with eight SOC analysts investigating a realistic malware scenario. Across the recorded assistant responses, all phase assessments were considered plausible and evidence was grounded. However, only 30 out of 72 investigative actions were directly actionable using available telemetry, indicating that grounded recommendations may not always be applicable. Qualitative findings showed that participants reacted positively to the separation of information and investigative suggestions, but reacted negatively to the Security Onion integration and recommendation specificity. They identified the prototype as a secondary decision-support tool within a SOC workflow. These results indicate the potential of combining retrievable threat and historical knowledge with structured generation for SOC investigation support. However, they also indicate that such systems remain as decision-support tools, and that the analyst is responsible for final critical decisions.

1 Introduction

Security Operations Center (SOC) analysts investigate alerts by combining evidence from detection systems and logs with threat intelligence and knowledge from previous investigations. These sources serve different purposes. Alerts identify potentially suspicious activity, while logs provide technical observations that must be correlated to reconstruct an incident. Cyber threat intelligence (CTI) reports describe known threat behaviours, capabilities, and indicators of compromise (IOCs), but are written as general threat descriptions. Furthermore, SOC analysts accumulate historical case knowledge through repeated case handling, documenting previous decisions, and customer-specific context. However, this knowledge can be difficult to retain and is unevenly distributed across analysts. Therefore, analysts must not only connect fragmented technical evidence, but also determine which historical context is relevant to the current investigation.

During an investigation, analysts are not only determining whether an alert is relevant but must also establish which hypothesis is supported by the current evidence, and what evidence should be examined next. Existing adversary frameworks support only parts of this process. The Cyber Kill Chain provides a high-level progression model for staged intrusion activity, while MITRE ATT&CK provides a detailed vocabulary for describing adversary tactics, techniques, and procedures [10, 21]. Both frameworks are useful for detection engineering, reporting, and standardized communication. However, neither framework is designed to assist in the reconstruction of an incident or the identification of missing evidence. Analysts must perform this interpretation themselves. Automating this analysis with generative AI could reduce analyst effort, but would introduce concerns with both reliability and accountability. Generated investigative assessments can contain unsupported conclusions or overlook relevant evidence. Blindly trusting generated outputs can lead to critical failures within SOC environments.

Large Language Models (LLMs) offer a potential way to support analysts by synthesizing fragmented information into structured investigative guidance. However, directly using an LLM for SOC investigations is risky. A model can produce unsupported claims, overstate confidence, and conflate inferred and observed events [9]. This is significant, since SOC decisions depend on evidence quality and source origin. This challenge is further complicated by the contextual organizational knowledge required during investigation, including the prior handling of similar cases, customer-specific information, and historical investigations. Parts of this organizational knowledge are explicitly recorded in historical cases and reports, while other parts remain tacit knowledge in analysts' experience [24].

To address these challenges, this paper proposes a SOC AI Investigation Assistant that supports analysts in accessing and applying relevant incident knowledge during an incident investigation. The assistant gathers current case evidence, external threat intelligence, and historical case knowledge to provide structured incident guidance. It is intended to support analysts in reconstructing potential incident progression, identifying missing evidence, and determining appropriate investigative actions.

The prototype is integrated into Security Onion, an open-source security monitoring platform with SIEM and case-management functionalities. A dedicated AI tab allows analysts to request investigative support using evidence collected within a case. This deliberate design allows the analyst to retain control over the information submitted to the model and the resulting investigative decisions. A user study examines participant investigation outcomes and their experience of the system including usability, source clarity, actionability, trust, and workflow fit.

This paper makes the following contributions:

- An analyst-facing phase framework that supports reasoning from observed activity towards possible preconditions, missing evidence, and subsequent investigative actions.
- A hybrid RAG-knowledge graph architecture that combines textual CTI and historical case retrieval with structured relationships between threat and investigative concepts.
- A case-integrated investigation assistant that maintains information separation and generates structured investigative support within Security Onion.
- An exploratory empirical evaluation with eight SOC analysts, examining investigation outcomes and qualitative experience with the prototype.

*e-mail: z.a.mirza@student.tue.nl

2 Background

2.1 Security Operations Centers (SOCs)

Security Operations Centers (SOCs) monitor, detect, investigate, and respond to cybersecurity incidents. They collect and analyze telemetry from sources such as Security Information and Event Management (SIEM) systems, Endpoint Detection and Response (EDR), intrusion detection systems, and network sources [25]. SIEM systems often serve as the central aggregation and correlation layer, since they ingest large volumes of security data and generate alerts based on detection rules. These rules may also be mapped to adversary frameworks such as MITRE ATT&CK, meaning that alerts can contain technique or tactic labels to help analysts interpret the detected behaviour.

SOC investigations involve correlating alerts with logs, observables, and other available evidence. Table 1 illustrates what a typical alert, analyst hypothesis, and log source look like during a malware investigation. Analysts commonly organize this material into cases to preserve investigative context, record hypotheses, document findings, and support handover or escalation. Shared case records also allow investigative findings to remain accessible beyond the analyst who originally handled the incident.

SOC analysts are also typically divided into tiered analyst roles, although the exact structure varies across organizations. Tier 1 analysts perform initial alert triage and prioritization, while Tier 2 analysts investigate more complex incidents. Tier 3 analysts handle major incidents, threat hunting, and advanced analysis [25]. This tiered structure is relevant because investigation support tools may be used differently depending on analyst experience. Less experienced analysts may benefit from structured guidance, while more experienced analysts may use such tools to review evidence and validate hypotheses.

Table 1: An example of analyst input during a SOC investigation.

Input	Example

Alert	**Suricata:** ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM
	`src_ip: 203.0.113.45`
	`dest_ip: 10.0.5.20`
	`dest_port: 80`

Analyst context	"Possible SQL injection alert against our web server. Found a matching HTTP request, but not sure whether the attack succeeded."

Log evidence	**zeek.http**
	`method: POST`
	`uri: /login`
	`status_code: 500`
	zeek.conn
	`service: http`
	`conn_state: SF`

2.2 Cybersecurity Models for Cyber Attacks

The Cyber Kill Chain and MITRE ATT&CK are complementary frameworks used to describe and understand adversary activity. The Cyber Kill Chain provides a high-level sequential intrusion model, while MITRE ATT&CK provides a more granular taxonomy of adversary behaviour.

The Cyber Kill Chain is an intrusion-focused adversary model developed by Lockheed Martin to describe the progression of a cyber intrusion campaign [10]. The model organizes attacker activity into seven stages: reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on objectives. Its central idea is that each stage offers analysts a potential point of detection or intervention, giving SOCs a shared structure for communicating where defensive controls may interrupt an attack. It is most useful for explaining intrusion scenarios that follow a relatively clear progression from preparation to payload delivery, execution, command and control, and final objectives.

MITRE ATT&CK is a comprehensive knowledge base and behavioural taxonomy of adversary tactics and techniques based on real-world observations [21]. It organizes adversary behaviour into tactics, techniques, and sub-techniques. Tactics describe the adversary's short-term objective, such as Initial Access or Privilege Escalation. Techniques describe how these objectives may be achieved, while sub-techniques provide a more specific description of a technique variant. For example, a technique may describe credential access behaviour, while a sub-technique may specify the particular method used to obtain these credentials.

MITRE ATT&CK is widely used in SOC environments for detection engineering, threat intelligence analysis, alert enrichment, reporting, and coverage assessment. Detection rules and alert descriptions often include ATT&CK technique mappings, which can help analysts connect observed activity to standardized adversary behaviour. This standardization is valuable because it gives analysts, detection engineers, and threat intelligence teams a common vocabulary for describing attacker actions across various tools and reports. This detailed vocabulary can also complement the higher-level Cyber Kill Chain, since MITRE tactics and techniques can be associated with activities in each phase.

2.3 Large Language Models and Retrieval Augmented Generation

Large language models (LLMs) are transformer-based models trained on large-scale text corpora to generate natural language outputs [4]. In cybersecurity, they have been explored for tasks such as threat intelligence summarization, vulnerability analysis, malware explanation, and incident response support [27]. Their utility in SOC contexts comes from their ability to process heterogeneous information and produce structured explanations from fragmented data.

However, stand-alone LLMs introduce important risks. They may generate unsupported or incorrect claims, a problem commonly referred to as hallucination [9]. They may also lack access to updated domain-specific information, especially in rapidly evolving areas such as threat intelligence. The NIST AI Risk Management Framework identifies validity, reliability, accountability, and transparency as necessary characteristics of trustworthy AI, and highlights the importance of human intervention and oversight [19]. Furthermore, Tilbury and Flowerday [22] note that automation in SOCs can lead to misuse and analysts placing excessive trust in systems. Automation bias can also undermine analysts' situational awareness and lead to complacency in information validation and verification.

Retrieval-Augmented Generation (RAG) addresses part of this problem by combining information retrieval with LLM generation [7]. In a RAG system, external documents are first collected, divided into retrieval units called chunks, embedded into vector representations, and stored in a vector index. At query time, the query is embedded into the same vector space and compared with the indexed document chunks. Relevant chunks are retrieved and appended to the original prompt, which allows the model to generate responses using this evidence, rather than relying only on its training set knowledge.

RAG is particularly relevant for SOC investigation support because the required knowledge is distributed across alerts, previous cases, and CTI reports. A retrieval layer allows the system to incorporate domain-specific knowledge without retraining the underlying model. Nevertheless, RAG does not guarantee correctness. Its usefulness depends on the quality of the indexed documents, the chunking strategy, metadata, and the relevance of the retrieved context [7].

2.4 Knowledge Graphs

A knowledge graph is a structured representation of knowledge in which entities are modelled as nodes and relationships between them are modelled as edges [15]. This makes knowledge graphs suitable for representing information that is highly connected, relational, and difficult to capture explicitly through isolated text fragments alone. Compared with purely tabular representations, graph-based models make relationships between entities explicit and allow multi-hop queries across various connected concepts.

Knowledge graphs are relevant in cybersecurity because security knowledge is naturally relational. Threat actors, malware families, vulnerabilities, techniques, tactics, and IOCs are not independent objects, but are connected through operational and evidential relationships. Cybersecurity knowledge graphs can be used to organize threat intelligence, model attack behaviour, connect observables to possible techniques, and support reasoning over heterogeneous security data [13].

In retrieval-based LLM systems, knowledge graphs can complement vector retrieval. Vector retrieval is useful for finding relevant textual evidence, but it does not explicitly represent how retrieved pieces of information relate to one another. A knowledge graph can provide this missing relational context by linking entities such as threats, behaviours, objectives, preconditions, and investigative actions. This is particularly relevant for SOC investigation support, where the analyst needs not only relevant text, but also an understanding of how a behaviour may relate to an objective, what evidence may indicate it, and which data sources can be used for verification.

However, knowledge graphs also introduce practical challenges. Their usefulness depends on the quality of the schema, the consistency of entity naming, and the accuracy of the relationships inserted into the graph. Manual construction can be time-consuming, while automated construction may introduce errors that require validation.

2.5 Design Requirements

The preceding concepts identify several requirements for case-based SOC investigation support. Table 2 summarizes these concerns and the corresponding design responses.

3 Related Work

3.1 Cognitive and Interpretive Load on SOC Analysts

SOC analysts operate under sustained operational and cognitive pressure. SIEM platforms and rule-based detection systems create high volumes of alerts, of which many are benign or false positives. Repeated exposure to a large quantity of benign or low-priority alerts can contribute to alert fatigue and reduce sensitivity to genuine threats [20]. Beyond alert triage, analysts must correlate alerts, logs, and customer-specific context to reconstruct incidents. Conducting these activities under time pressure also increases the cognitive burden on the analyst.

Cybersecurity human-factors research has linked repeated cognitive overload to stress, burnout, and reduced decision-making performance [14]. These findings indicate that analysts' cognitive burdens are an operational concern for SOC environments, rather than individual issues. They also highlight the importance of reducing unnecessary interpretive effort. Therefore, relevant support must make information more accessible and interpretable, without removing the analyst's responsibility for investigative decisions.

3.2 Organizational Investigative Knowledge

Investigative knowledge is not limited to formal threat intelligence or individual case experience. Analysts also acquire knowledge through repeated investigations and familiarity with their operational environments. Research on organizational knowledge distinguishes knowledge recorded in documents and procedures from tacit knowledge developed through individual experiences [24].

This distinction has important implications for organizational knowledge sharing and retention within SOCs. No individual analyst handles every investigation, and relevant experience may remain unevenly distributed across a SOC. Knowledge that is not adequately documented or transferred may become inaccessible when analysts' change roles or leave the organization [5]. Documentation like historical cases and analysts notes can preserve investigative decisions, evidence, and conclusions. However, undocumented experience and tacit knowledge can be difficult for other analysts to 'access' or reuse [5]. Retrieval-based support can make organizational knowledge more accessible, but it cannot recover tacit knowledge that has not been documented or transferred to other analysts.

3.3 Limitations of Existing Adversary Models in SOC Operations

SOCs use adversary models to structure detection, triage, investigation, and response. Two widely used frameworks are the Cyber Kill Chain and MITRE ATT&CK, which describe adversary behaviour at different levels of abstraction. Both frameworks are valuable for communication, detection engineering, and reporting. However, they provide limited support for reconstructing complex incidents from SOC evidence. In this context, their main limitation is not a lack of descriptive value, but their limited support for reasoning about dependencies, preconditions, and partially observed activity during an investigation.

The Cyber Kill Chain is useful for describing staged intrusion campaigns and identifying potential points of defensive intervention [10]. However, its sequential structure is less suited to incidents that are iterative, identity-driven, or parallel. Modern attacks can skip stages, or move between capability-building and objective-focused actions multiple times. As a result, the Cyber Kill Chain provides limited support when analysts must reconstruct incident progression from disjointed alerts and logs rather than from a complete incident narrative.

MITRE ATT&CK provides analysts with a standardized vocabulary for describing observed behaviours [21]. However, it is not designed as a process-oriented model for incident reconstruction. It does not encode preconditions, dependencies, or temporal ordering between techniques. A technique may also appear under multiple tactics depending on the attacker's objective and the surrounding evidence. Thus, ATT&CK helps analysts name observed behaviour, but it does not by itself explain how that behaviour fits into a broader incident narrative.

These limitations matter because SOC analysts often work from limited evidence. They may have an alert, some relevant logs, and limited threat context, but not a complete view of how the incident began or what conditions enabled it. In such cases, tactic labels, technique mappings, or Kill Chain stages are useful descriptors, but they do not directly answer the operational questions that analysts need to resolve. Such questions include what the attacker is doing, what had to be true for this behaviour to occur, what evidence would confirm or reject the hypothesis, and what should be checked next. This motivates the need for an analyst-facing reasoning structure that connects observed behaviour to possible preconditions, phase context, and investigative actions.

3.4 Use of LLMs in Security Operation Centers

Recent work increasingly examines LLMs as components of SOC workflows, motivated by persistent operational pressure such as high alert volume, fragmented security data, and limited analyst capacity [8, 18]. Existing surveys organize this literature around the core stages of a SOC workflow which are detection, analysis, and response [8]. This organization is useful because it shows that LLM adoption in SOCs is not limited to a single task, but spans the full investigative cycle, from initial alert generation to reporting. More recent surveys indicate that there is a shift from single-shot LLM

Table 2: Design concerns and requirements for SOC investigation support.

Design concern	Requirement	Design response
Fragmented case evidence	Structure observations and identify missing evidence.	Analyst-facing phase framework.
Distributed recorded knowledge	Retrieve relevant CTI and historical investigations.	Semantic retrieval using RAG.
Implicit relationships in threat knowledge	Represent connections between threats, behaviours, and objectives.	Knowledge graph.
Confusion between evidence and context	Preserve source attribution and evidence value.	Structured prompt and source presentation.
Unsupported or generic guidance	Provide justified assessments and actionable evidence checks.	Structured investigation output.
Fragmented analyst workflow	Integrate support into case management while retaining analyst control.	Security Onion integration.

assistance towards agentic and more integrated tool deployments, in which LLMs are embedded as reasoning components within a pipeline, rather than a stand-alone chat interface [18, 27]. This shift is relevant to the present paper, since the proposed system is similarly structured as a reasoning layer embedded within an existing SOC platform, rather than a general-purpose assistant.

Within the analysis and triage stage specifically, several systems automate the classification and prioritization of alerts. Non-LLM approaches such as Automated Alert Classification and Triage (AACT) learn from analyst triage actions on cybersecurity alerts to automatically close benign alerts and prioritize critical ones. AACT has reduced the number of alerts shown to analysts by 61% over six months, with a low false positive rate of 1.36% [23]. Other LLM-based systems extend this line of work through multi-agent designs. CORTEX uses specialized agents for activity inspection, evidence-gathering, and reasoning [26] while Audit-LLM uses a similar multi-agent approach to detect insider threats [17]. LLMs have also been applied to security log analysis, since logs are large and difficult to interpret manually. LLMLogAnalyzer is a clustering-based log analysis chatbot which helps analysts query and interpret large log volumes [3]. These studies demonstrate that LLMs can automate or assist with specific SOC activities.

Operational studies focus on how analysts experience and use LLMs during real investigations, rather than measuring their task-level accuracy. Singh et al. conducted an empirical study of 3090 analyst queries from 45 SOC analysts over 10 months and reported that analysts use LLMs as assistive tools for sense-making, rather than decision-making [16]. Most of the queries were related to interpreting security telemetry. This is directly relevant to the evaluation part of the paper, which also focuses on analyst experience, trust, and perceived usefulness rather than automated performance alone. This finding also emphasizes the importance of understanding how generated support is used within an existing investigative workflow.

Research on structured investigation processes explores a related but separate concern. Kersten et al. report that Tier 1 analysts who follow a four-stage threat analysis process are 2.5 times more likely to produce an accurate assessment than those who do not [11]. This result suggests that a structured approach for an investigation process, independent of AI assistance, can improve analyst performance. A follow-up tool operationalizes this structure within a SOC platform, supporting Tier 1 analysts in analyzing and understanding network security events [12]. Kersten et al.'s work makes a unique methodological point of isolating the effect of investigative structure from AI assistance. These findings establish a basis for considering a defined investigation structure when conducting an incident investigation.

This prior work demonstrates that LLM-based systems can support specific SOC activities such as alert triage, log interpretation, and event analysis, while structured investigation processes can improve analyst assessment accuracy. However, the investigation problem considered in this paper extends beyond single tasks or information sources. Analysts may need to combine case evidence with external threat intelligence and organization-specific historical case knowledge, while preserving the distinction between what is observed in the current case, and what has been provided as supporting context. Existing LLM-based approaches support specific analysis tasks, while structured investigation approaches provide reasoning guidance. This leaves an open question in terms of how retrieval-grounded LLM support and an analyst-facing reasoning structure can be combined to make relevant incident knowledge accessible and translate it into actionable investigative guidance while preserving source origin and uncertainty. This gap motivates the system design and evaluation presented in this paper.

4 Problem Statement

Investigative knowledge within a SOC is distributed across various sources, including external threat intelligence, historical case records, and the experience of individual analysts. Although previous investigations may document relevant evidence, investigative decisions, and conclusions, this knowledge may be difficult to access and reuse during future investigations. Relevant customer context may be documented in previous cases, but it may also depend on familiarity accumulated by analysts who have previously worked with that environment. At the same time, investigative knowledge can remain unevenly distributed across analysts. Therefore, personnel turnover or incomplete knowledge transfer may make historically relevant context difficult to recover during later investigations.

Access to relevant knowledge alone does not resolve the investigative problem. Analysts must interpret fragmented evidence from the current case and determine which explanations are supported by the available observations. CTI and historical cases provide contextual knowledge about potential threat behaviour and previous incidents, but do not establish that the same activity occurred in the current case. As a result, analysts must distinguish observed facts from contextual information, identify remaining uncertainties, and determine which evidence should be examined next.

The resulting problem is one of both knowledge accessibility and investigative interpretation. Relevant information may exist across CTI, historical investigations, current case evidence, and analyst experience, but it is not practically available to the analyst in a form that can assist the current investigation. Therefore, SOC analysts need support that can retrieve relevant recorded investigative knowledge, relate it to the current evidence, and preserve the distinction between what is observed, what is inferred, and what remains uncertain. It must also allow the analyst to retain control over investigative decisions.

This paper investigates how a hybrid RAG-KG-assisted LLM assistant can support evidence-aware SOC case investigation. The main research question is:

1. **Main Research Question:** How can SOC analysts be supported in retrieving accessible and operational incident knowl-

edge during an alert or case investigation?

- **RQ1.1:** To what extent does the proposed SOC AI Investigation Assistant generate grounded and operationally applicable responses with plausible phase assessments and investigative steps?

- **RQ1.2:** How do SOC analyst participants experience the SOC AI Investigation Assistant during a realistic case investigation, particularly in terms of usability, source clarity, trust, actionability, confidence of suggestions, and workflow fit?

RQ1.1 is addressed through a descriptive assessment of the recorded assistant responses to examine whether a plausible phase assessment was performed, the response was grounded, and if the investigative actions are applicable to the evaluation environment. RQ1.2 is addressed through a user study in which SOC analysts use the Security Onion prototype during a realistic case investigation, think aloud during tool use, complete a written report, and participate in a semi-structured post-session discussion.

5 Methodology

This paper uses a mixed, prototype-based methodology to investigate how SOC analysts can be supported in accessing and utilizing relevant incident knowledge. The proposed system is developed according to the requirements identified in Table 2 and is evaluated through a descriptive assessment of generated investigative guidance and a preliminary user study examining analysts' interaction with the prototype.

The expectation is that combining accessible incident knowledge with structured investigative support can assist analysts in interpreting fragmented evidence and verifying their conclusions. However, the quality of generated responses and their usefulness within an operational workflow are separate concerns. As a result, the two research questions are explored using different sources of evidence.

5.1 Descriptive Assessment of Generated Investigation Support

RQ1.1 examines whether the assistant provides operationally useful investigative support. To assess this, all recorded assistant responses generated during the eight evaluation sessions were reviewed. The number of requests varied between participants, but all responses were selected, which came to a total of 24.

The assessment uses the information available to the model at the time of the request. This is important because evidence identified later in the investigation should not be used to judge an earlier response. The information that is considered for each assessed response includes the current case evidence consisting of the alert, analyst context, and log evidence.

Two properties are assessed at the response level which are the phase assessment and the grounding of the response. The phase assessment is considered **plausible** when the assigned phase and its justification are compatible with the evidence available at the time of the generation request. This criterion evaluates plausibility rather than assuming that a partially observed incident can only have one phase classification.

A response is considered to be **grounded** when it does not provide a case-specific claim that is not supported by the current evidence. For example, the model cannot claim that data was exchanged between the victim and the attacker if there are no logs showing that a connection was established.

Another assessed aspect is the applicability of the investigation actions provided. Each generated response contains three structured investigation rows, and a row is considered to be applicable when the proposed action is relevant to an unsolved aspect of the investigation, and it can be conducted using the available telemetry. Recommendations that depend on unavailable telemetry are classified as not applicable, even if the response is technically feasible. This assessment is conducted by the researcher and is interpreted as a descriptive evaluation of the recorded outputs. It is also only evaluated for one scenario and cannot establish reliability across different incidents.

5.2 Evaluation of Analyst Experience

RQ1.2 investigates how SOC analysts experience the prototype during a realistic case investigation. Accessible contextual knowledge and structured guidance can assist investigative reasoning, but practical value depends on whether information is clear, useful, and trustworthy. These dimensions require explicit participant feedback rather than a technical assessment.

A user study involving SOC analysts is conducted using a common malware investigation scenario in Security Onion. The study combines quantitative investigation measures with qualitative observations to examine both investigation outcomes and analysts' experience with the prototype. Quantitative evidence includes initial and final hypothesis confidence, and the final hypothesis conclusion. Confidence measurements describe changes in participants' reported certainty but are not treated as a measure of investigative accuracy.

Qualitative evidence is collected through think-aloud observation and semi-structured post-session discussions. Observations are categorized using seven predefined themes derived from the evaluation objectives. Their evaluative value and explicit suggestions are recorded separately. The researcher examines both the frequency of coded observations and their distribution across participants to avoid interpreting repeated comments from one participant as agreement across the group.

The different sources of evidence are compared to identify where reported experiences, observed interaction, and investigation outcomes are consistent or diverge. The study uses a consistent scenario and system configuration to support comparison between participants. However, its small sample size, single scenario, researcher-led assessment, and lack of a control group limit generalizability, and prevent causal claims about improvements in investigative performance.

6 System Design

This chapter describes the design and implementation of the SOC AI Investigation Assistant. The system combines an analyst-facing phase framework, a hybrid knowledge base consisting of a vector store and knowledge graph, a structured LLM investigation pipeline, and an integration within the Security Onion case-management interface. The following sections describe the architecture of these components, how investigative context is represented and retrieved, how the retrieved information is assembled for generation, and how the resulting investigation support is presented to the analyst.

6.1 System Overview

Figure 1 shows the various components of the system and how they interact with each other. The runtime workflow begins with case input provided by the analyst, including an alert, contextual notes or a working hypothesis, and relevant log evidence. An example of this input is shown in Table 1. This input is transformed into retrieval queries for both the vector store and the knowledge graph. The retrieved vector chunks and graph relationships are kept separate so that their evidential roles remain transparent. Current case evidence is treated as the primary basis for reasoning, while retrieved CTI, historical case context, and graph relations are treated as supporting context.

The final output is a structured investigation response containing a narrative summary, a phase assessment, possible preconditions, evidence to check, relevant data sources, next investigative steps, and a confidence assessment. The analyst can then use this output to

validate or challenge the initial hypothesis, identify missing evidence, and continue the investigation. In the prototype, this workflow is integrated into Security Onion through the case-management interface, which is shown in Figure C.

6.2 Analyst-Facing Phase Framework

The analyst-facing phase framework provides an investigative abstraction between the broad sequential stages of the Cyber Kill Chain and the granular behavioural taxonomy of MITRE ATT&CK. Crime script analysis provides additional inspiration by organizing activity around enabling conditions, actions, and outcomes [6]. Direct adoption of crime script stages is unsuitable for the intended SOC workflow, since its terminology introduces additional complexity that does not map onto common investigation language. Therefore, the framework uses a smaller set of analyst-oriented phases adapted to a SOC context.

It consists of six phases, described in Table 3. They include: Preconditions, Entry, Establishing Foothold, Operational Execution, Impact/Benefit, and Residual Presence. Preconditions, actions, and outcomes reflect the structure of crime script analysis and the distinction between progressive attacker activities reflects the sequential nature of the Cyber Kill Chain. ATT&CK provides the behavioural vocabulary used to interpret activities within each phase. Impact/Benefit includes both the consequences of the incident for the victim and the objectives or benefits realized by the attacker, such as monetary gain, credential/information theft, or disruption of services. Residual Presence additionally refers to artifacts and compromised resources that remain after the main intrusion activity. This framework is illustrated in Figure 2.

The framework does not impose a strict sequence on activity occurring during the active incident. Behaviours such as discovery, persistence, command and control, and lateral movement may overlap, recur, or occur in different orders. Establishing Foothold and Operational Execution are grouped within a broader Action Thread. This structure allows observed activity to be placed within an incident narrative while supporting backward reasoning toward possible preconditions, missing evidence, and subsequent investigative steps. It also provides the LLM with a consistent structure for organizing its response.

6.3 Hybrid Knowledge Base

The system uses a hybrid knowledge base to retrieve both relevant knowledge and explicit relationships between investigative concepts. Semantic retrieval identifies relevant sections from CTI reports and historical cases, providing detailed descriptions of threat behaviour and previous investigative outcomes. However, the semantic similarity of text alone does not represent how threats, behaviours, objectives, and investigative actions are connected.

A knowledge graph complements semantic retrieval by representing these relationships explicitly. It allows the investigation pipeline to retrieved related concepts and investigative steps without relying solely on the LLM to infer these relationships from the retrieved text. However, the graph does not contain a narrative or case-specific context contained in original documents. As a result, both components are designed to provide complementary investigative knowledge.

The vector stores and knowledge graph are constructed independently and retain their respective source information. They are queried together during an investigation request and their outputs are provided to the investigation pipeline as separate contextual sources.

6.3.1 Vector Store Construction

The vector store contains selected CTI reports and historical SOC cases. CTI sources were collected from established public institutions and contain information including threat behaviours, indicators of compromise, mitigation guidance, and other threat-specific context. Historical SOC cases contain previously recorded investigation material, including observed evidence, analyst findings, and investigation conclusions. Semantic retrieval allows relevant material to be identified even when the terminology in the case differs from that used in the source documents.

Before indexing, the documents are parsed and normalized into a common retrieval format. CTI reports are processed using Docling to preserve document structure, while SOC documents are parsed with pdfplumber due to their simpler format. Non-investigative material including preambles, acknowledgements, and reference sections are removed where applicable. The remaining content is enriched with metadata such as the source document, threat or malware family, section type, and ATT&CK related labels. CTI material additionally retains source-quality metadata used during retrieval. CTI reports and historical cases are indexed in their own vector stores so that each can be separately queried.

Retrieval units (also known as chunks) are created according to the structure of each source type rather than using fixed-size chunking alone. CTI reports are parsed and cleaned before their content is divided into sections covering topics such as threat behaviour, indicators of compromise (known domains and hashes), mitigation guidance, and ATT&CK mappings. Section boundaries are identified using the document's structure and content, and are also manually reviewed to prevent parsing and logic errors. A list of commonly used headings is used to conduct a first pass of chunking, after which missed headings or incomplete sections are manually edited and fixed. Each resulting chunk contains the relevant section text and metadata identifying its source, threat, and section type. Table 4 illustrates the structure of a CTI retrieval unit.

Historical cases are stored as large retrieval units to retain the relationship between observed evidence, investigative actions, and resulting conclusions. Essentially, each historical case is stored as its own chunk. However, alerts and logs are not indexed into the vector store, since they can dominate semantic similarity. Instead, the analyst selects current-case evidence when submitting an investigation request, and relevant log information is processed and supplied directly to the investigation pipeline. This distinction keeps evidence from the current incident separate from the external and historical knowledge to support the interpretation.

Each retrieval unit is embedded using Qwen3-Embedding-4B and stored in Chroma together with its metadata. The resulting vector store supports semantic similarity search and metadata-based filtering, including filtering by attributes such as threat name, section type, or quality. Retrieved chunks retain their source metadata so that the CTI report or historical case remains visible to the analyst alongside the generated investigation output.

6.3.2 Knowledge Graph Schema

The relational component of the hybrid knowledge base is implemented as a Neo4j knowledge graph. The graph schema represents the entities and relationships required by the investigation pipeline, including threats, behaviours, objectives, alerts, preconditions, investigative steps, data sources and analyst-facing phases.

The graph distinguishes between knowledge originating from CTI and observations recorded in historical SOC cases. CTI reports represent general threat knowledge and are connected to entities describing threats, associated behaviours, objectives, and ATT&CK mappings. Historical cases represent incident-specific observations and connect cases to alerts, behaviours, indicators, and data sources observed during those investigations. Source entities are retained within the schema so that the origin of graph relationships can be preserved during retrieval.

The schema is intentionally scoped to relationships that are relevant to the investigation workflow rather than attempting to model a complete cybersecurity ontology. Behaviour entities can be associated with threats and objectives, mapped to ATT&CK techniques, connected to relevant preconditions, and linked to investigative steps

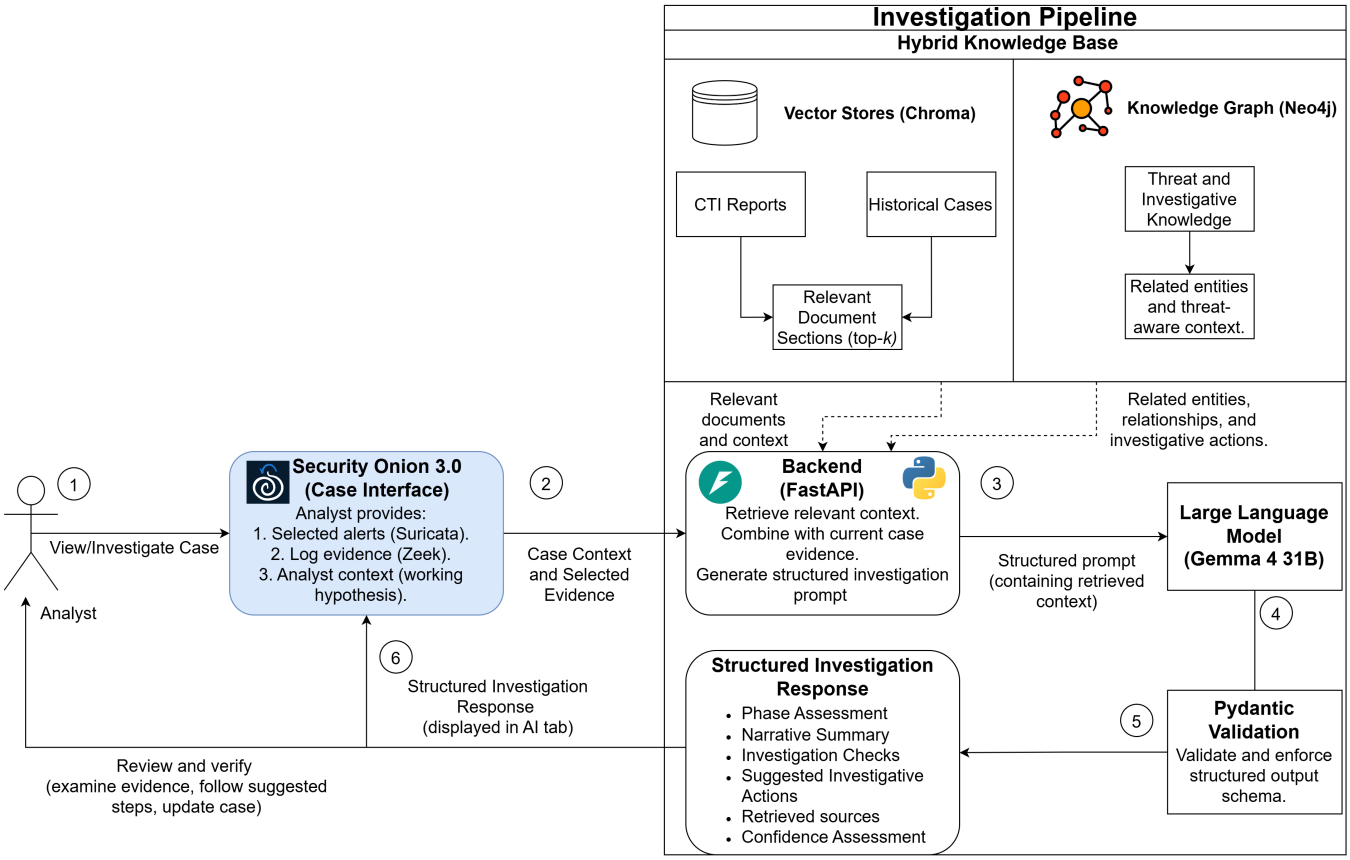


Figure 1: Overall architecture of the proposed SOC AI Investigation Assistant. Offline processing converts CTI reports and historical SOC cases into a hybrid knowledge base consisting of a Chroma vector store and a Neo4j knowledge graph. At runtime, case-specific input from Security Onion is used for hybrid retrieval, structured prompt assembly, LLM generation, and return of a structured investigation response together with supporting source context.

Table 3: Operational and SOC interpretation of each phase within the framework.

Phase	Operational definition	SOC investigation interpretation
Preconditions	Environmental states that make the intrusion possible.	Vulnerabilities, exposed services, misconfigurations, or other enabling conditions that existed before observable attacker activity.
Entry	Initial point and method of access into the environment.	Phishing, credential abuse, exploitation of public-facing services, or other initial access mechanisms affecting a specific asset or identity.
Establishing Foothold	Actions that strengthen the attacker's position in the environment.	Persistence, discovery, credential access, tool staging, or preparation of compromised systems for later operations.
Operational Execution	Actions that advance the attacker's objective.	Command and control, lateral expansion, collection, propagation, exfiltration, or other behaviour directed toward operational goals.
Impact/Benefit	Realized outcome of the intrusion.	Disruption, unauthorized access, data theft, extortion outcome, service degradation, or other forms of operational or strategic harm.
Residual Presence	Artifacts left after the intrusion.	Persistence mechanisms, compromised identities, modified systems, remaining infrastructure, or evidence of cleanup and disengagement.

and data sources. Behaviours can additionally be associated with the analyst-facing phase framework, allowing relational knowledge to be connected to the structure used in the generated investigation response. Table 5 summarizes the main schema groups and representative relationships used in the graph.

The schema, entity types, and permitted relationship types

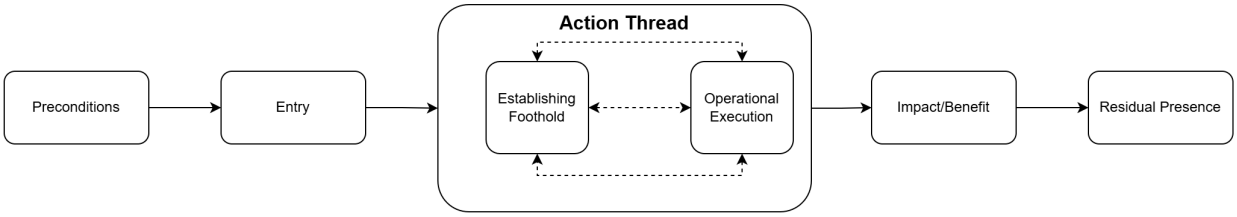


Figure 2: Analyst-facing phase framework for SOC investigation support. The framework separates Preconditions, Entry, Action Thread, Impact/Benefit, and Residual Presence. Establishing Foothold and Operational Execution are grouped within a broader Action Thread to reflect the overlapping nature of activity during modern incidents.

Table 4: Example of a CTI retrieval unit produced during document ingestion.

Field	Example
Source	Threat Actors Deploy LummaC2 Malware to Exfiltrate Sensitive Data from Organizations.
Section	Overview
Content	LummaC2 malware first appeared for sale on multiple Russian-language speaking cybercriminal forums in 2022...
Metadata	*Threat*: LummaC2; *sectiontype*: overview; *source*: lummac2.pdf

were defined manually. Graph population was performed semi-automatically, with an LLM generating Cypher statements from the source material, after which the generated statements were manually reviewed before execution in Neo4j. This review step was used to maintain consistency with the predefined schema and prevent unsupported entities or relationships from being introduced. Both CTI-derived and case-derived observations remain attributable to their originating source entities within the graph. As a result, graph retrieval can return the source from which the relationships were derived, and this information is retained when the graph context is passed to the investigation pipeline.

6.4 Investigation Pipeline

The investigation pipeline converts case context into a structured investigation response through five stages: query preparation, hybrid retrieval, structured prompt assembly, LLM generation, and validation. This process has been illustrated in Figure 3. The pipeline receives the alert, analyst context or working hypothesis, and optional log evidence from the active case. From this input, the system derives a focused retrieval query and a set of relevant entities such as threat names, malware families, and behaviours. The retrieval query is used for semantic search in the vector store. Where suitable metadata is available, filters such as threat name or section type further restrict the search space. Identified entities are also used to query matching relationships in the knowledge graph.

Vector and graph retrieval are executed as independent retrieval paths. The vector stores returns CTI and historical case chunks together with their source metadata. The knowledge graph returns relationships connected to the entities identified during retrieval preparation. The retrieved results are inserted into separate sections of the generation prompt. Current case evidence, CTI context, historical case context, and knowledge graph context remain distinct. The full prompt can be found in Appendix B.

The prompt also includes the analyst-facing phase framework and the required Pydantic output JSON schema. The LLM returns a structured investigation response rather than unrestricted free text. The response contains a narrative summary, phase assessment, source-based justification, possible preconditions, evidence checks, relevant data sources, recommended investigation actions, and a confidence assessment. The backend parses the generated response into the expected output structure using Pydantic before returning it to the analyst interface. Retrieved source information is retained alongside the generated investigation so that the analyst can inspect the CTI report, historical cases, and graph context associated with the response.

The pipeline preserves the traceability of sources through generation. Retrieved CTI, historical cases, and graph context remain available alongside the generated response, allowing the analyst to inspect the material supplied to the LLM. They can also independently verify relevant sources. Therefore, the pipeline provides a traceable path between case evidence, retrieved context, generated interpretation, and analyst verification. However, it does not guarantee that every generated claim is correct. The analyst remains responsible for verifying and validating both information and suggested actions.

6.5 Integration into Analyst Workflow

The prototype is integrated into the Security Onion case-management interface through a dedicated AI tab shown in Figure 4. The tab allows the analyst to provide case-specific context, configure the available retrieval (Figure 5) and generation settings, and submit an investigation request. The selected case information is sent to the investigation pipeline described in the previous section. The case provides a natural boundary for an investigation since alerts, observables, analyst notes, and supporting evidence for an incident can be accumulated over time in a shared record. As a result, the analyst controls which case evidence is submitted to the LLM. The prototype does not search the SIEM or select additional telemetry.

The resulting structured response is returned to the same case interface together with the supporting CTI, historical case, and knowledge-graph context. This is shown in Figure 6. As a result, analysts can inspect the generated assessment and its supporting sources without leaving the active case. They also have access to both a current investigation analysis and a historical case analysis. The current investigation analysis focuses on the evidence surrounding the incident, providing relevant information like a narrative summary and the next steps to do. The historical case analysis focuses on whether this case has been seen before, and if so, what steps were taken, and what the analyst should be aware of. Figure 7 shows another panel which contains precondition checks, which are essentially conditions which had to be true based on the current evidence. This can allow the analyst to reason backwards and check relevant

Table 5: Summary of the main schema groups in the knowledge graph.

Schema group	Main entities	Representative relationships
Intelligence sources	CTIReport, Case	MENTIONS, INVOLVES THREAT, OBSERVED behaviour
Detection evidence	Alert	HAS ALERT, MAY INDICATE, MAY MAP TO
Threat model	Threat, Objective, Behaviour	COMMONLY EXHIBITS, PURSUES OBJECTIVE, FURTHERS OBJECTIVE
ATT&CK grounding	Tactic, Technique, Sub-technique	MAPS TO TECHNIQUE, PART OF TACTIC, COMPOSED OF
Investigation logic	Precondition, Investigative Step, Data Source	REQUIRES PRECONDITION, VERIFY WITH, USES DATA SOURCE
Analyst abstraction	Narrative Phase	CAN OCCUR IN PHASE

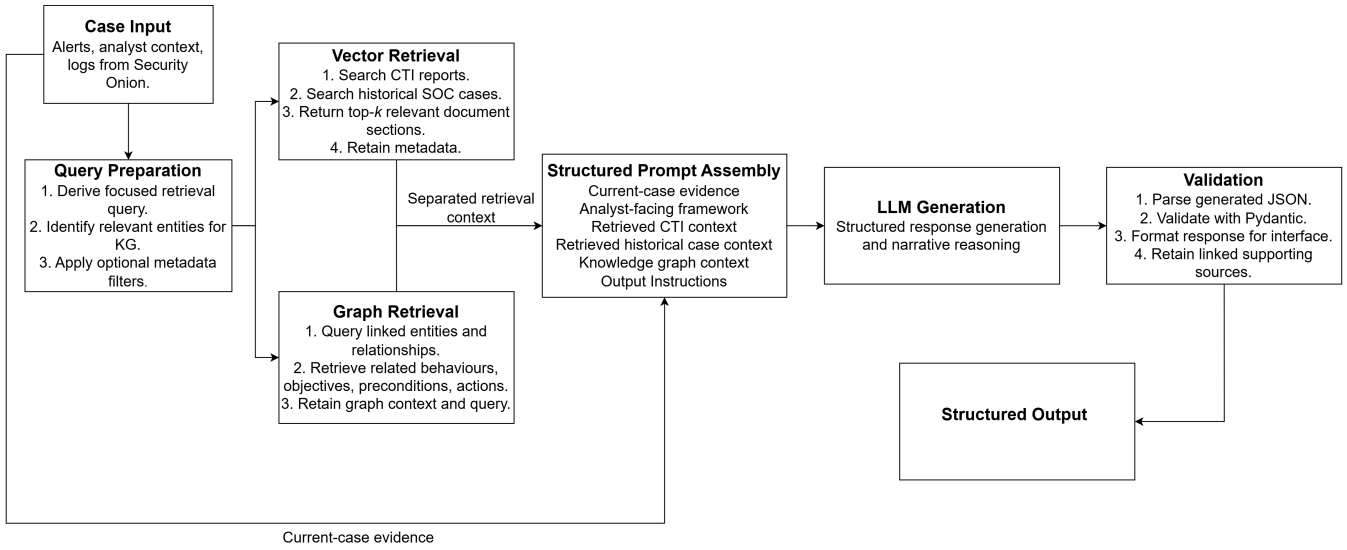


Figure 3: Investigation pipeline that takes the case input from Security and conducts hybrid retrieval using a focused retrieval query. The retrieved CTI historical cases are appended into the structured prompt alongside current-case evidence for LLM generation. This JSON response is validated using Pydantic and formatted for the interface.

telemetry. Finally, Figure 8 shows a summary panel that contains both external threat sources and historical case sources. Analysts can click on each result to view them in more detail and verify them as needed. As the investigation develops and new evidence becomes available, the assistant can be queried again. Generated responses can also be stored as case comments, allowing the output to remain part of the case record for later review, handover, or escalation.

This design prioritizes analyst control over evidence collection. Since the prototype does not search the wider SIEM telemetry, relevant information may be absent from the model context if it has not been identified or added to the case. In return, the analyst retains control over the evidence used for generation and can review the resulting response within the same case workflow. The user study examines the practical implications of this design for usability, trust, and workflow fit.

6.6 Default System Configuration

Table 6 summarizes the default runtime configuration used in the prototype during the evaluation. These values were selected during prototype testing and are kept constant during the evaluation.

A generation temperature of 0.2 was selected to reduce output variability and prioritize consistent responses over creative generation. The maximum output length was set to 1800 tokens to maintain a balance between sufficient detail for the analyst to digest, and adhering to the Pydantic schema. During prototype testing, lower output token limits resulted in generic responses and higher output token limits resulted in verbose responses and generation problems. These generation problems were specifically issues with new-line generation when the LLM tries adhering to the output schema.

Vector retrieval uses a top-k of three results from each vector store (CTI and historical cases). This value was chosen to provide several potentially relevant retrieval units while limiting the amount of contextual information provided to the LLM. Increasing the top-k value introduces additional input tokens and can also add less relevant material which can mislead the LLM. These values represent specific choices made during prototype development, rather than a random selection of parameters.

7 User Study

The user study investigates how SOC analysts experience the SOC AI Investigation Assistant when using it to evaluate a suspected compromise within an existing Security Onion interface. It addresses RQ1.2 by examining the usability of the integrated interface, the clarity of supporting information, the actionability of generated recommendations, participant trust in the assistant, and its fit within an investigation workflow.

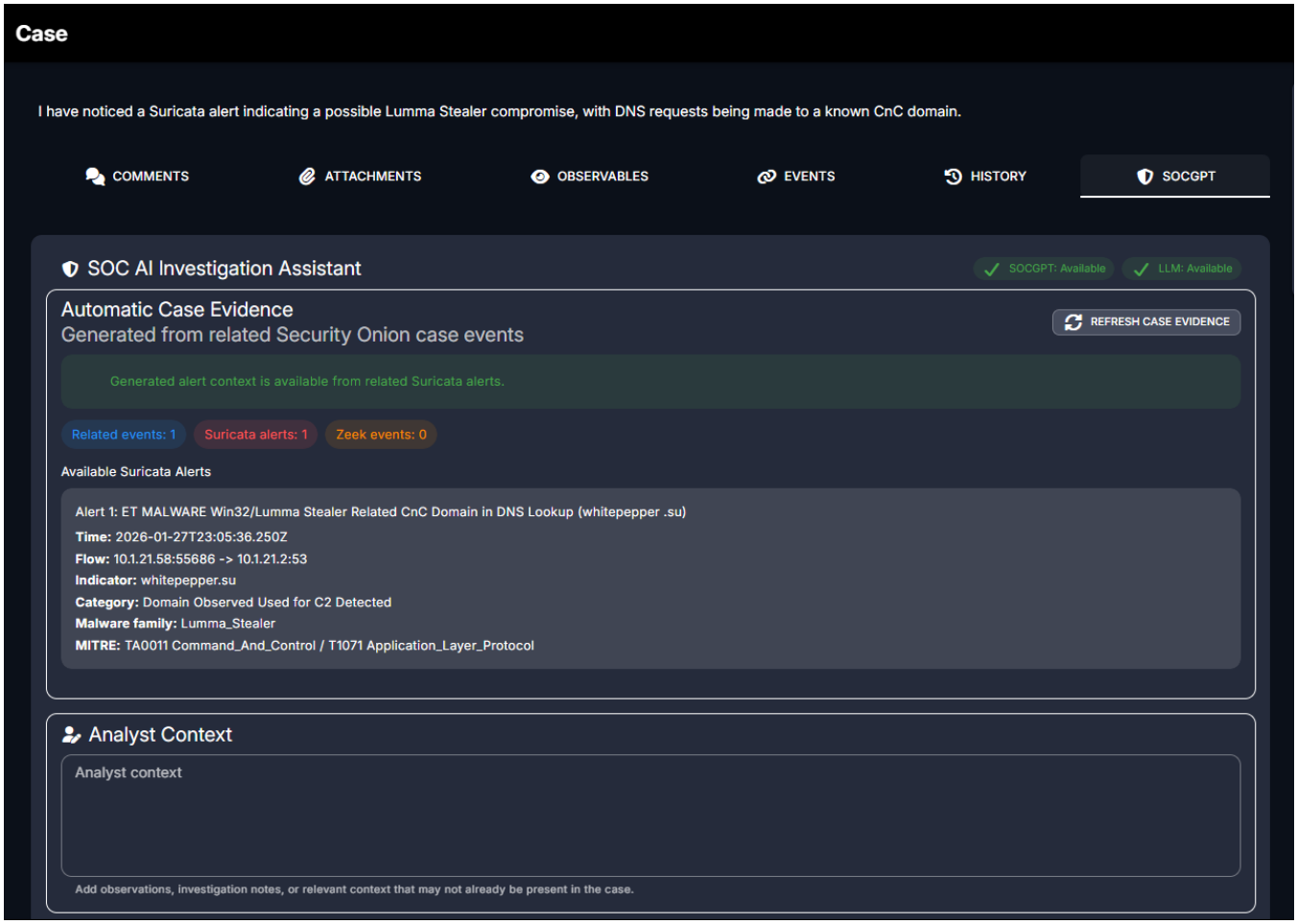


Figure 4: The analyst can add case evidence and refresh to check what has been added in the automatic case evidence box. Analyst context can be added in the appropriate box which includes observations and investigation notes that are relevant to the incident.

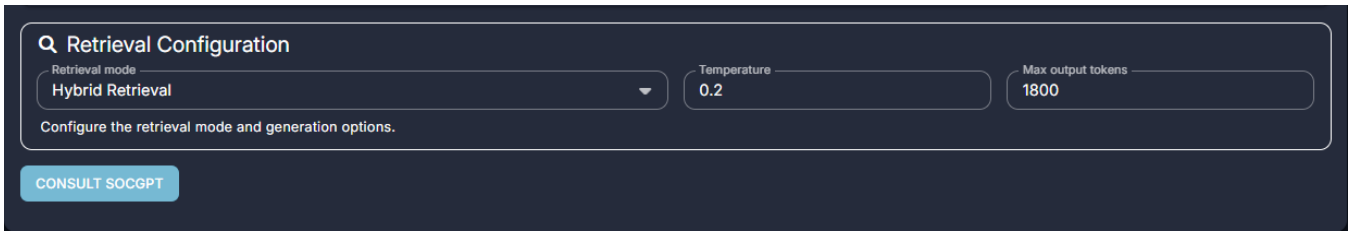


Figure 5: The retrieval configuration can be adjusted in terms of what retrieval mode to select, the temperature of the LLM, and the maximum number of output tokens. The following retrieval modes are available: No Retrieval, Vector Retrieval, Hybrid Retrieval and Knowledge Graph Retrieval.

Participants perform a realistic malware investigation and receive an initial alert and a working hypothesis at the start of the investigation. The hypothesis concerns a potential compromised Windows host. Their task is to examine the available telemetry, evaluate the hypothesis, and produce a written report documenting their conclusion and supporting evidence. Each participant investigates the same predefined scenario.

The study uses a mixed-method design with investigation reports and initial and final confidence ratings providing descriptive information about investigation outcomes. Think-aloud observations and semi-structured interviews provide insight into how participants interact with the assistant, interpret AI recommendations, and identify both practical strengths and limitations of the prototype. The study does not include a non-assistant comparison condition and does not aim to measure the assistant's causal effect on investigative performance. The study procedure, environment, and system configuration remain consistent across participants.

There was no fixed time limit, so participants were allowed to end the investigation once they considered that they had gathered sufficient evidence to reach a conclusion. During the scenario, participants could ask questions about operating the prototype or Security Onion interface, but the researcher did not provide investigative guidance, interpret evidence, or answer questions concerning the hypothesis.

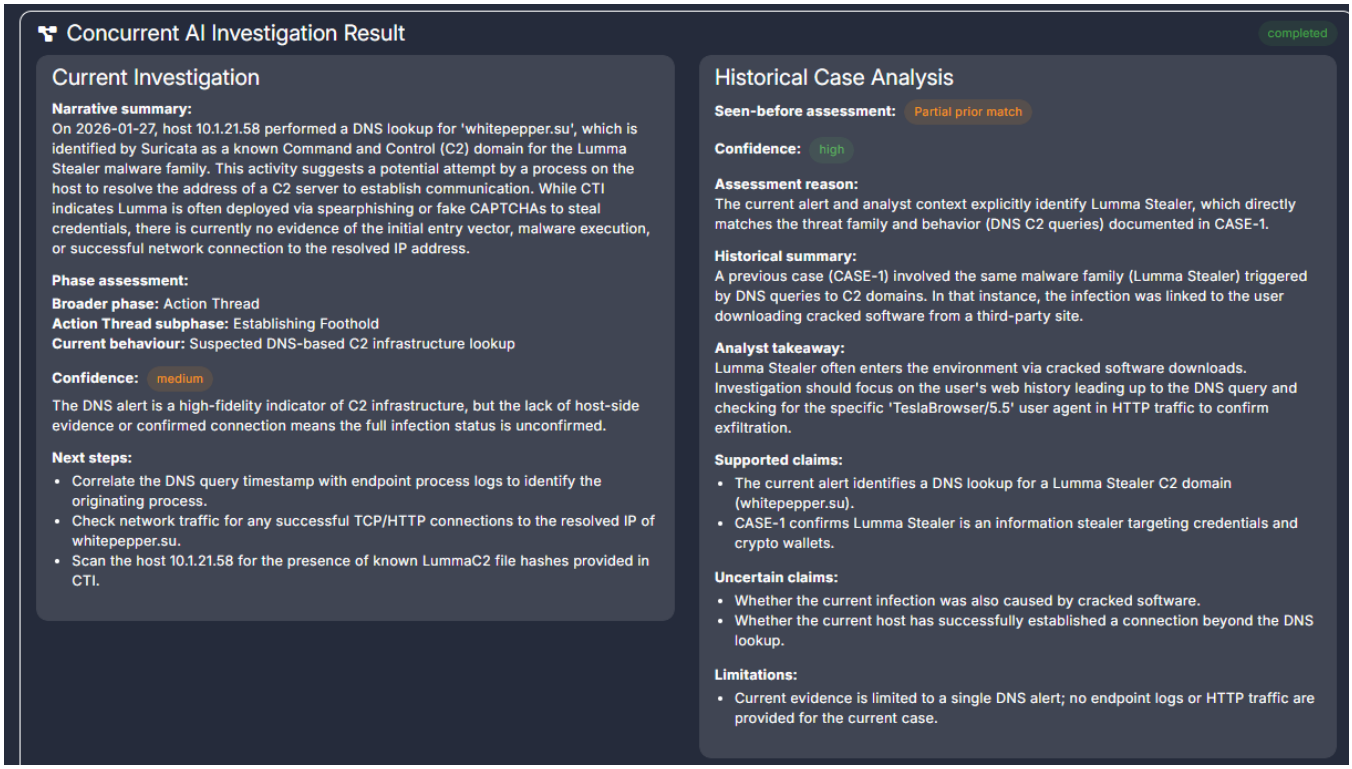


Figure 6: The result of the LLM pipeline contains a current investigation panel and a historical case analysis panel. The first panel provides a narrative summary, a phase assessment, the confidence, level, and the relevant next steps. The historical case analysis indicates whether this threat has been seen before, a summary of the previous case, the analyst take-away, and important claims.

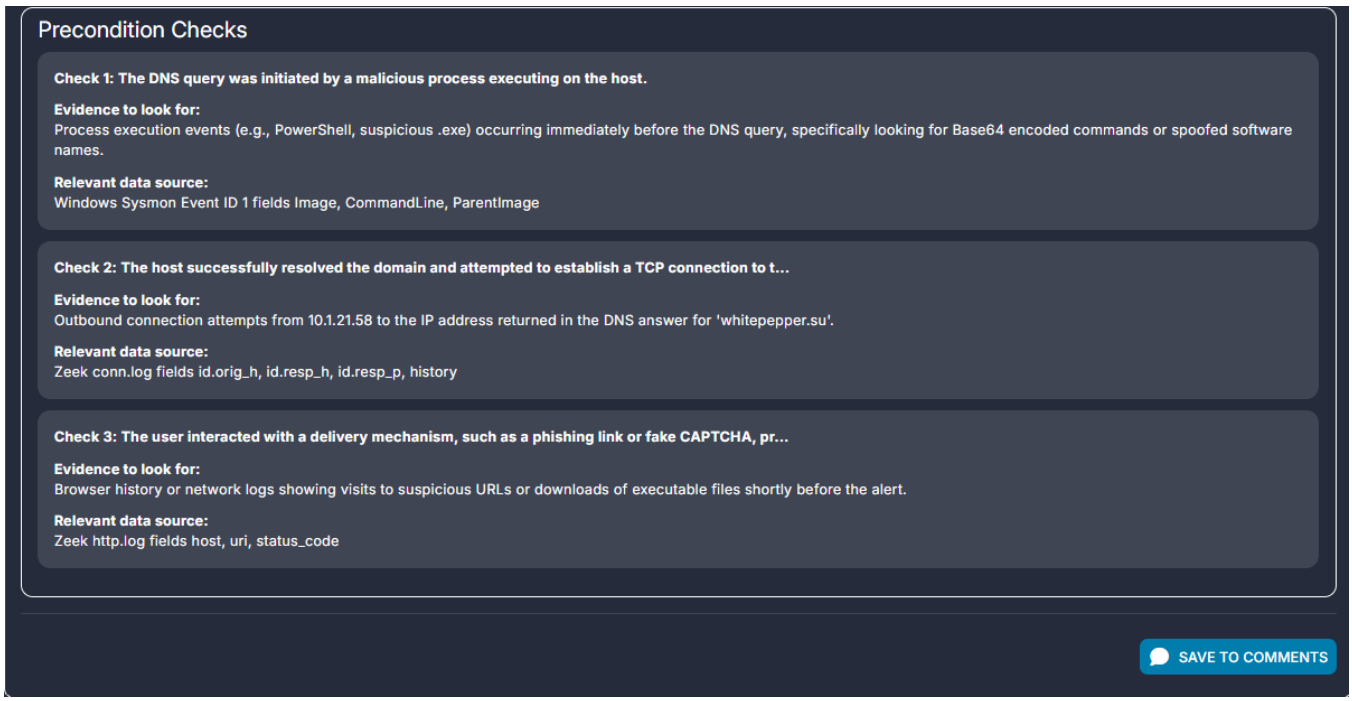


Figure 7: The precondition checks panel indicates which conditions had to be true to be at this stage of the investigation. It indicates what logs and events the analyst should look for to determine how the incident unfolded and how to develop the investigation further.

7.1 Participants

Eight SOC analysts from the Eindhoven Security Hub participated in this study, comprising of six Tier 1 and two Tier 2 analysts. All

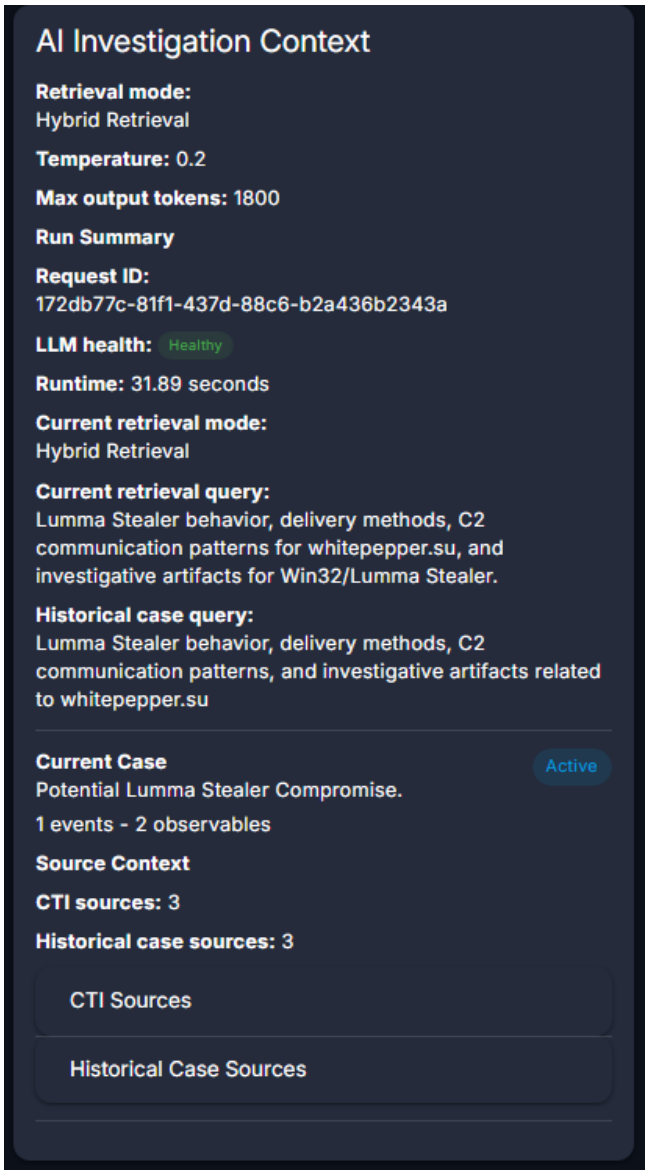


Figure 8: The information panel containing both the external threat sources and the historical case sources. Analysts can click on each result for a drop-down to view each source individually.

participants have completed Tier 1 analyst training and have prior experience with Security Onion. Seven participants report some prior familiarity with LLM-based tools. A complete illustration of participant characteristics can be found in Table 7. Seniority and self-reported LLM familiarity are recorded as descriptive participant characteristics to interpret individual study behaviour and responses.

7.2 Study Environment

The study uses Security Onion 3.0 with the AI investigation interface described in Figure C. The prototype provides access to the case evidence, the analyst-context input, retrieval configuration, generated investigative guidance, and supporting information panels.

The runtime configuration remains fixed across sessions, as specified in Table 6. Hybrid retrieval was enabled, generation temperature was set to 0.2, and vector retrieval returned up to three chunks from each indexed source. Participants are not allowed to modify these settings during the study. Keeping these parameters constant retains control across the environment, ensuring that each participant is interacting with the same prototype configuration.

Table 6: Default system configuration used in the prototype.

Component	Configuration
Generative model	Gemma 4 31B
Embedding model	Qwen3-Embedding-4B
Vector store	Chroma
Graph database	Neo4j
Retrieval mode	Hybrid (vector and knowledge graph retrieval)
Default top-k	3 retrieved CTI and historical case chunks each
Temperature	0.2
Maximum output tokens	1800
Output format	Structured JSON parsed with Pydantic
Security platform	Security Onion 3.0

Table 7: Participant characteristics.

Participant	SOC tier	LLM familiarity
P1	1	4
P2	1	4
P3	1	3
P4	1	3.5
P5	2	4
P6	1	2
P7	2	1
P8	1	4

A clean Security Onion environment is prepared for each participant. Each session uses a new anonymous account and a fresh case containing only the evaluation material. Before each session, previous case comments and investigation artifacts are removed. Underlying scenario logs and alerts are also removed and re-imported before each session to prevent duplicate or previously escalated events from appearing in the environment. Therefore, each participant begins from the same initial case state and is not impacted by actions performed by previous participants.

The study used separate scenarios with different purposes. A **sparse-evidence demonstration case** was used before the evaluation scenario to introduce participants to the AI interface and to show how the assistants responses differ when only limited evidence was available. The evaluation scenario contained the malware investigation that participants were asked to complete. Extra information on both scenarios can be found in A.1 and A.2 respectively.

7.3 Investigation Scenario and Task

The evaluation scenario was based on network traffic associated with Lumma Stealer from Malware-Traffic-Analysis.net [1].At the start of the task, each participant received a Security Onion case containing an initial Suricata alert and a predefined hypothesis. The task was to determine whether the available evidence supported that hypothesis. The alert and hypothesis are as follows:

> **Alert**: ET MALWARE Win32/Lumma Stealer Related CnC Domain in DNS Lookup (whitepepper .su)
>
> **Hypothesis**: A Windows host is compromised by Lumma Stealer and is communicating with Lumma-associated infrastructure.

The initial alert provided a starting point but was insufficient on its own to establish the full extent of the suspected compromise.

The Security Onion environment contained additional telemetry consisting of both relevant and unrelated events. Participants needed to investigate beyond the initial alert and look for additional evidence while distinguishing unrelated events.

Participants were instructed to critically test the hypothesis rather than searching for evidence to confirm it. They could examine the available Security Onion telemetry, append relevant alerts and logs as case evidence, provide contextual notes to the assistant, and then inspect the generated results. The assistant only provided investigative suggestions instead of concrete findings, and participants remained responsible for deciding which conclusions to accept or reject.

The final deliverable was a written investigation report that documented the participant's assessment of the hypothesis, supporting evidence, any additional findings, their final confidence, and suggested next steps. There was no fixed time limit imposed, so participants could end the investigation when they considered the available evidence to be sufficient enough to complete the report.

7.4 Study Material and Procedure

Two materials support the user study which are an investigation checklist and an investigation report template. The checklist specified the tasks that participants were expected to complete. The report template provided a format for documenting final conclusions, supporting evidence, additional findings, final confidence, and recommended actions.

Each session follows the same procedure:

1. **System Demonstration**
   The researcher introduced the assistant using the sparse-evidence case. The demonstration covered the interface, submission of an investigation request, interpretation of the response, and the purpose of supporting information panels. Participants were informed that generated responses could be incomplete or incorrect, and that they remain responsible for verification.

2. **Initial Case Assessment**
   Participants received the evaluation case and its predefined hypothesis. Before using the assistant, they record their initial confidence in the hypothesis on a five-point scale, where 1 indicates low confidence and 5 indicates high confidence. Participants are also instructed to critically test the hypothesis rather than attempting to solely confirm it.

3. **Independent Investigation**
   Participants received the investigation checklist and began examining the evaluation case. they could investigate Security Onion telemetry, submit investigation requests to the assistant, and inspect its generated guidance, and continue investigating. They were encouraged to verbalize their observations and decisions throughout the task.

4. **Investigation Report**
   When participants believed that they had gathered sufficient evidence, they completed the report template. They recorded their final hypothesis assessment, supporting evidence, additional findings, final confidence, and recommended next steps.

5. **Post-session Interview**
   After completing the investigation, participants took part in a semi-structured discussion covering interface clarity, information separation, generate investigative actions, uncertainty, trust, and workflow fit. Follow-up questions were used to clarify individual responses.

The researcher could answer questions about operating Security Onion or the prototype but did not provide investigative guidance, interpret evidence, or indicate whether a participant's reasoning was correct. Participants were informed that the study evaluated the prototype rather than their individual analytical ability. Sessions were recorded for subsequent transcription and observational notes were taken throughout the task.

Table 8: Qualitative coding themes used in the user study.

Theme	Focus
Security Onion Integration	Understanding and use of the integrated AI functionality.
Information Panels	Clarity and usefulness of supporting information presented by the interface.
Investigation Actions	Relevance, grounding, and actionability of suggested investigation steps.
Information Separation	Distinction between current case evidence and CTI or historical case context.
Confidence	Whether system uncertainty is communicated consistently with the available evidence.
Trust	Factors affecting participant willingness to rely on or question system output.
Workflow	Fit with existing SOC investigation practices and barriers to operational use.

7.5 Data Collection and Coding

Quantitative measures include initial and final hypothesis confidence. These measures are analysed descriptively across participants.

Qualitative material consists of think-aloud observations, researcher notes, and the post-session discussion. Session recordings are reviewed and relevant observations are coded using a structured thematic coding schema derived from the study evaluation dimensions. The coding was done by a single researcher.

Each coding observation has four attributes: theme, type, evaluative value, and suggestion status. The theme identifies the part of the system or investigation experience under discussion. The type classifies the observation as UI, AI output, evidence/data, or process/workflow. Evaluative value is coded as positive, negative, or neutral. Suggestion status records whether the participant proposes an explicit change to the system.

Positive observations describe aspects of the prototype considered to be useful or effective, while negative observations identify areas of improvement or limitations. Neutral observations are descriptive without expressing a clear positive or negative judgement. Suggestion status is coded independently from evaluative value.

The seven coding themes are defined in Table 8. They are applied consistently across participants to support descriptive comparison in the Results section.

8 Results

This chapter reports the findings of the user study. The analysis is descriptive and focuses on participant investigation outcomes and qualitative experience with the prototype. Quantitative results include changes in hypothesis confidence and the final assessment reached by each participant. Qualitative results summarize the coded observations collected during think-aloud sessions and post-session discussions. Due to the small sample size, statistical generalization should not be done based on these results.

Table 9: Summary of the assessment of recorded assistant responses.

Assessment	Result	Rate
Plausible phase assessment	24/24 responses	100%
Evidence grounded	24/24 responses	100%
Applicable investigation rows	30/72 rows	41.7%
Non-applicable investigation rows	42/72 rows	58.3%

8.1 Generated Investigation Support

The eight evaluation sessions produced 24 recorded assistant responses, with the number of requests varying per participant. All recorded responses were included in the assessment.

As shown in Table 9, all 24 responses contained a phase assessment that was considered plausible given the evidence available at the time of generation. All 24 responses also satisfied the grounding criterion, since no claim was marked as supported, unless it was referenced by the provided case evidence.

The main limitation was observed in the applicability of the generated investigation rows (actions). Across the responses, the system generated 72 investigation rows, of which 30 (41.7%) were not applicable within the available environment. The remaining 42 rows (58.3%) were not applicable within the available environment. A recurring reason was reliance on telemetry that was not available to participants, specifically endpoint, device, and firewall logs. In some instances, the recommendation was feasible, but could not be performed due to to lack of sources available in the scenario. Other actions were infeasible, such as conducting a memory wipe of the affected asset.

8.2 Participant Investigation Outcomes

Figure 9 shows the change in participant confidence at the start and end of the investigation. The mean confidence increases from 2.88 to 3.75, with five participants reporting a higher final confidence, two reporting no change, and one reporting lower confidence. The largest increase is observed for P5, whose confidence rises from 2 to 5. However without a control group, the use of the assistant cannot be the reason for this increase. It is also possible that by conducting the investigation, the participant felt more confident.

Figure 10 shows the final participant assessment of the investigation hypothesis. Five of the eight participants conclude that the hypothesis is supported, which matches the expected scenario conclusion. Two participants classify the hypothesis as partially supported, while one concludes that the available evidence is insufficient.

8.3 Participant Contributions and Suggestions

Figure 11 shows the distribution of explicit suggestion observations across participants. Of the 104 coded observations, 26 are marked as suggestions. All eight participants contribute at least one suggestion, although the frequency varies. P1 contributes six suggestions and P2 contributes five, accounting for 11 of the 26 suggestion observations. P4 contributes the fewest, with one suggestion.

8.4 Qualitative User Experience

Figure 12 shows the evaluative distribution of coded observations across the seven qualitative themes. Information Separation receives the highest proportion of positive observations (78%), followed by Confidence (60%) and Investigative Actions (56%). Security Onion Integration contains the highest proportion of negative observations (55%), while Information Panels is more evenly divided between positive (43%) and negative (46%) feedback. Workflow observations are predominantly neutral (50%), with 30% positive and 20% of negative observations.

The positive evaluation of Information Separation is also reflected in participant comments. P8 describes the separation as "quite clear" and "clearly labelled", while P6 describes the Current Investigation panel as "super useful" because it provides both a self-check and a summary. Feedback on the wider information panels is more mixed. P5 notes that CTI information is "a bit hidden" and suggests presenting the underlying source URL rather than an identifier such as CTI-1. P1 similarly considers the case summary to be less useful since it mostly repeats information that is already available elsewhere.

Feedback on Investigative Actions is mixed. P8 states that the suggested next steps are "quite useful" because they provide a "possible list of indicators, more than you might be able to come up with on the spot". P2 also considers the suggestions useful, but notes that they become repetitive over time. P1 reports a related limitation, describing some checks as "hyper-focused" and suggesting that precondition checks adapt to the available log evidence. Participant's perceptions of confidence also vary. P2 describes most responses as "hedged with medium confidence", while P3 considers the model to be "a bit overconfident" when it cannot directly inspect the PCAP.

Security Onion Integration receives the highest proportion of negative coded observations. P2 identifies the separation between Cases and Hunt as a barrier, stating that switching between the two interrupts the investigation workflow. P4 also mentions that repeated page refreshing is confusing during an active investigation. However, P6 considers the interface to be user-friendly and does not report any navigation issues.

Trust and Workflow observations show that participants mostly describe the assistant as a supporting tool for an investigation. P5 describes the tool as "an intermediary, not a replacement", while also noting that it can be "a bit too agreeable". P8 describes a similar workflow in which the "the analyst investigates first, uses the assistant to confirm or question findings, and manually verifies the result" because "AI should not have the final say". participants also identify separate points at which the tool may be most useful. P3 considers the tool to be more appropriate when an analyst is stuck, while P7 describes it as a way to "suggest additional checks and potentially speed up investigations".

9 Discussion

The findings provide an initial view of both the generated investigation support and analysts' experience of using the prototype. The response assessment shows consistent evidence grounding and plausible phase attribution, while identifying issues in the operational use of investigative suggestions. The user study shows that participants generally valued information separation and investigative suggestions, but found limitations with some information panels and the integration within Security Onion.

9.1 Generated Investigation Support

The generated response assessment showed that across 24 recorded assistant responses, all phase assessments were considered to be plausible and that evidence was properly grounded. These findings are limited to the evaluated scenario and are not evidence of the models reliability. However, they indicate that the evaluated responses maintained the intended distinction between observed case evidence and supporting contextual information.

The same consistency was not observed for the applicability of investigative actions. Of the 72 generated rows, 30 (41.7%) could be checked using the available logs in the environment. The remaining rows frequently depend on data sources that were unavailable to participants, including endpoint, firewall, and device logs. In several instances, the recommendation was technically feasible. This finding raises the point that the current pipeline reasons about which evidence could theoretically validate a hypothesis but it does not contain a representation of the specific data sources that are available

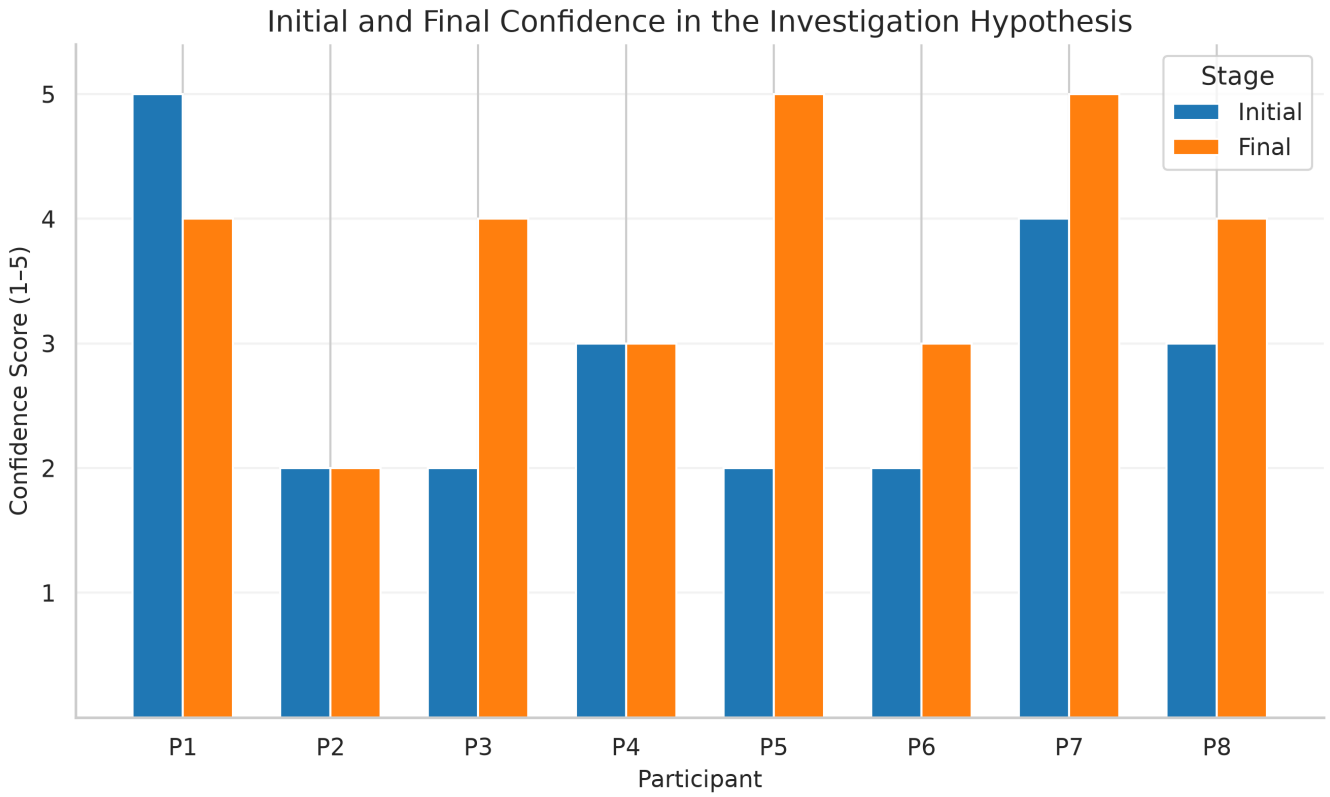


Figure 9: Initial and final participant confidence in the investigation hypothesis, measured on a five-point scale.

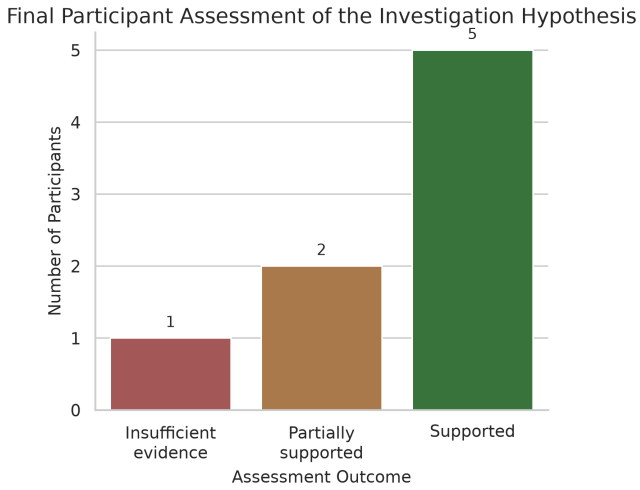


Figure 10: Final participant assessment of the investigation hypothesis. The expected conclusion of the scenario is **supported**.

to the analyst. As a result, the system can generate valid investigative suggestions that cannot be performed in the environment.

This assessment also aligns well with several participant observations. Participants reacted positively to the investigative suggestions but some described them as overly specific, repetitive, and incompatible with the available evidence. For example, P1 suggested that precondition checks should adapt to the available logs instead of being generic. This suggests that the assistant should account for the analyst's available telemetry. This could be done by providing the investigation pipeline with a detailed list of available sources and requiring investigative actions to specifically use these sources, or emphasize that extra unavailable evidence is required.

9.2 Changes in Analyst Confidence

Participant confidence generally increases over the course of the investigation. Five of the eight analysts report higher final confidence, while two remain unchanged and one reports lower confidence. The mean increases from 2.88 to 3.75, suggesting that participants finish the investigation with a clearer position on the initial hypothesis. Between the measurements, participants analyse the case evidence, develop their hypotheses, and interact with the assistant. As a result, the effects of these factors cannot be separated within the study.

The final assessment results provide additional context for this confidence change. Five participants conclude that the hypothesis is supported, which matches the expected conclusion for the scenario. The remaining three participants reach more cautious assessments. Two participants classify the hypothesis are partially supported, and one considers that the available evidence is insufficient. A review of participant reports and assistant responses was conducted to check whether evidence cited by participants have been suggested by the assistant. In several cases, the assistant had recommended data sources that were reflected in the participant's report. However, the exact sequence is inconclusive and it is not possible to determine whether participants investigated this evidence because of the assistant or whether the assistant confirmed their reasoning.

These findings are encouraging for the intended decision-support role of the system, but they do not establish an improvement in analyst accuracy. The study does not include a control condition in which the same investigation is completed without the assistant. As a result, it is not possible to determine whether the observed

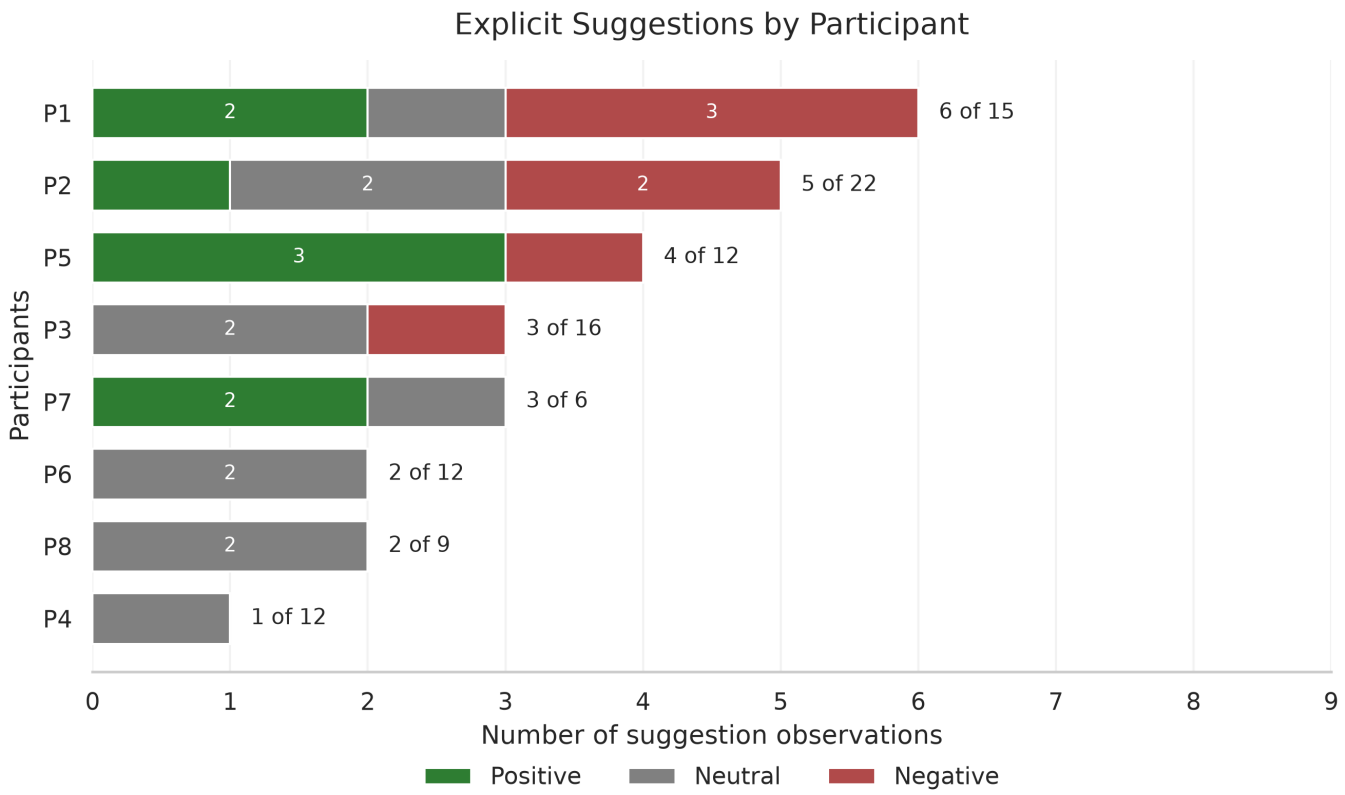


Figure 11: Explicit suggestion observations by participant, grouped by positive, neutral, and negative evaluative value. Labels at the end of each bar indicate the number of suggestion observations relative to the participant's total coded observations.

confidence changes or whether final conclusion differ from what analysts would have achieved using Security Onion alone.

9.3 User Experience and Workflow Fit

The qualitative results show that participant experience differs considerably across the various system components. Information Separation receives the strongest positive distribution, with 78% of coded observations classified as positive. This aligns with the central design objective of the prototype, to keep current case evidence distinguishable from CTI and historical case context. This result suggests that participants generally recognize and value this separation during investigation. Investigative Actions also receives more positive than negative observations, indicating that the generated follow-up steps are usually perceived as useful or relevant to the investigation by the analyst.

In contrast, Security Onion Integration contains the largest proportion of negative observations at 55%. This result is important since it suggests that positive views of the underlying investigative support do not automatically translate into a frictionless workflow. A system may provide useful information while creating integration problems within the environment analysts must use it. A repeated theme across participants was the friction in switching between Hunt and Cases, and refreshing the tab often for new information to load. However, this is a limitation of the software and difficult to bypass.

Information Panels receives a nearly balanced distribution of positive and negative observations. This indicates that the panels provide value for some aspects of the investigation but also remain a source of criticism for other aspects. Figure 8 received frequent criticism due to its location, and many participants did not use it. Participants have suggested to rework it to be an overlay showing the alert or having it scroll down with the user to be more useful. Workflow observations are mostly neutral since participants describe how the prototype would fit into existing investigative processes, rather than expressing a positive or negative judgement.

Trust and Confidence contain fewer coded observations than other themes, with only six and five observations respectively. Therefore, their percentages should be interpreted with caution. The balanced Trust distribution does not establish either strong trust or distrust of the prototype. Similarly, a large proportion of positive Confidence observations provide limited evidence due to the small number of coded statements.

9.4 Information Separation and Source Presentation

The participants' response to information separation supports one of the central design choices of the prototype. The system deliberately distinguishes current case evidence from CTI, historical cases, and other retrieved context, so that background information is not conflated as evidence that an event occurred in the current incident. Participants generally recognize this distinction in the interface, suggesting that the separation implemented within the investigation pipeline remains visible at the analyst-facing level.

The mixed response to the information panels shows that the source origin and separation alone is not sufficient. Analysts also need supporting information to be easily visible and verifiable. The specific request to provide the original CTI URL rather than an internal identifier emphasizes this difference. The system preserves the source origin internally but makes verification difficult for the analyst. Therefore, the future interface design should prioritize direct visibility and access to the underlying source and reduce panels that repeat information that is already displayed elsewhere in the case.

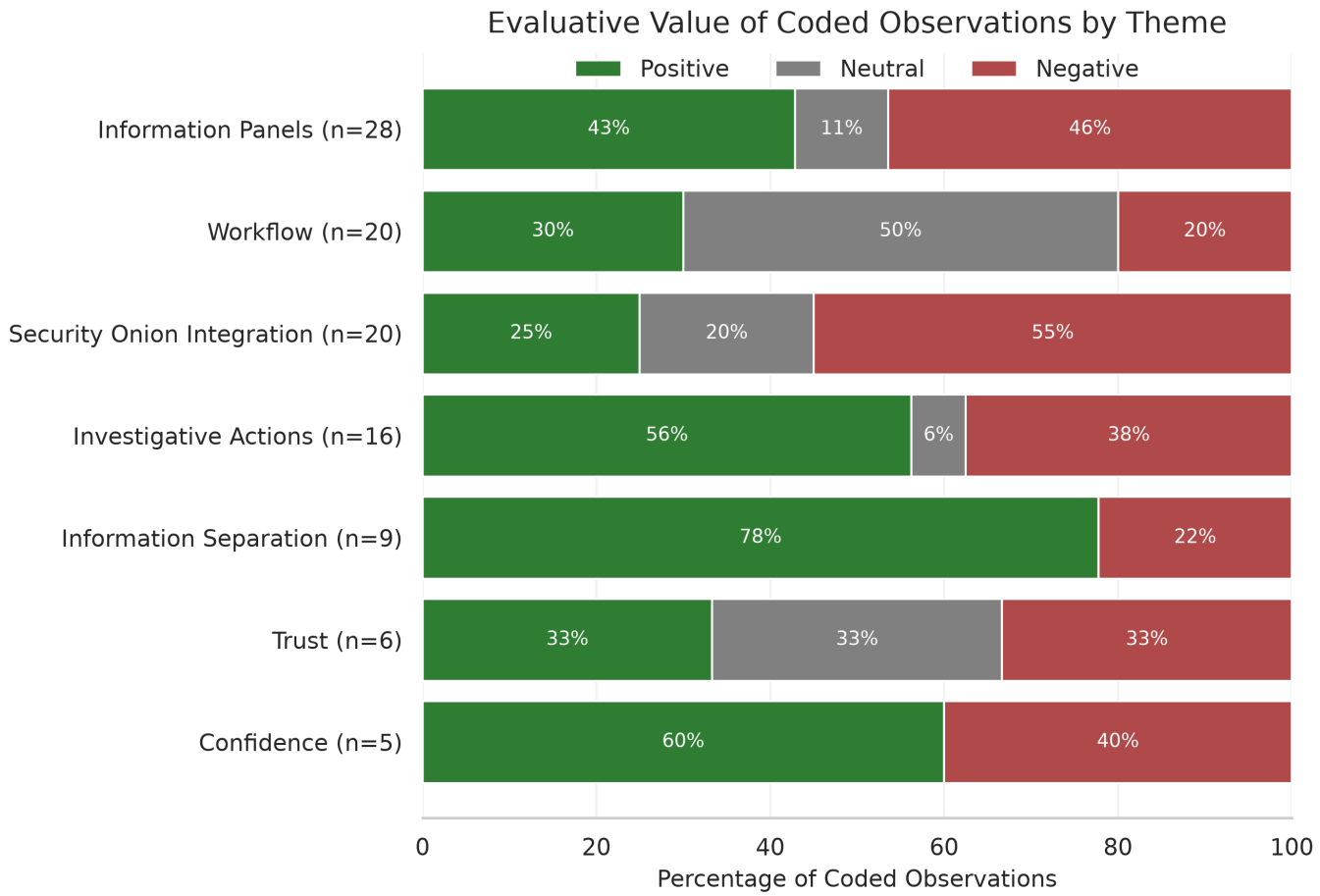


Figure 12: Evaluative value of coded participant observations across the seven qualitative themes. Percentages show the distribution of positive, neutral, and negative observations within each theme. n denotes the total number of coded observations for that theme.

9.5 Investigative Actions and State

Participant feedback on investigative actions suggests that usefulness depends on more than the plausibility of a step. Participants value the assistant when it exposes checks, indicators, or evidence sources that they have not considered independently. This supports the intended role of the system as an investigation aid that converts retrieved knowledge into operational next steps.

The negative observations imply that recommendations should also reflect the current state of the investigation. Repetitive and overly specific checks provide little additional value and can increase the amount of information the analyst must process. This was also observed in the generated response assessment, where not all suggestions are applicable to the environment. This limitation is consistent with the current prototype, which does not maintain a persistent session memory of previous investigation requests. A session-aware version of the pipeline could retain which evidence has already been examined and use that information to prioritize unresolved checks. As a result, actionability should not only focus on the correctness of an investigation step, but also its relevance to what the analyst has already established in the investigation.

9.6 System Integration and Workflow Fit

The integration results show that useful investigative output does not produce a smooth analyst workflow. Several participants identify the navigation between Cases and Hunt, page refreshing, and context loss between requests as sources of friction. These issues are not inherently related to the quality of the generated investigation response, but they affect how a case can be developed and conducted.

This distinction is important for the decision to integrate the assistant into Security Onion rather than deploying it as a separate application. Embedding the application within the case workflow of a SIEM reduces having to use multiple tools, but the prototype requires analysts to move between the case management interface and the telemetry interface. As a result, there should be a deeper integration in preserving the investigation state through session memory and reducing the navigation between both case management and telemetry interfaces. It may be possible to experiment with having an interface be an overlay on top of another, although this would require further research.

Participants also differ in terms of where they would use the assistant during an investigation. Some participants mention that tool is most useful when an investigation becomes complex or a second opinion is required. Others see value in using it earlier during an investigation to generate as many possible checks to investigate. This suggests that the prototype does not have a single fixed point within the SOC workflow, but rather its value depends on each individual analyst. Based on analyst experience and case complexity, the tool can be used at different stages of an investigation, and it is possible that the tool may not be deeply used, if an analyst decides to escalate the case to a Tier 2 analyst.

9.7 Trust

Participants indicate a preference for using the prototype as a confirmation mechanism and a second perspective, rather than a re-

placement for their own investigation. This aligns well with the methodological choice to keep the analyst solely responsible for selecting case evidence, validating generated claims, and making operational decisions during the course of the investigation.

However, maintaining analyst control does not automatically produce an appropriate level of trust in the prototype. Participants identify both overconfidence and affability in generated responses. It has also been noted that the prototype tends to re-affirm and be agreeable in the investigation. These observations show that trust also depends on whether the system can clearly communicate uncertainty based on provided evidence. This is particularly important if there is not enough evidence for strong conclusions. In such situations, the assistant should be clear about the limited evidence and hedge its interpretations, instead of confidently emphasizing them.

Therefore, the findings emphasize that a decision-support role is best for the assistant. This role involves helping analyst formulate and challenge hypotheses while leaving key decisions such as evidence collection, validation, and judgement with the analyst. This role is also consistent with participants' preference to use the system as a sparring partner or secondary opinion rather than an autonomous decision-making tool.

10 Limitations and Future Work

This paper evaluates a bounded prototype under controlled conditions. Its main limitations revolve around the scope and construction of the knowledge base, dependence on the fixed model and retrieval choices, analyst selection of case evidence, and the validity of the user study. These limitations constrain the degree to which the results can be generalized beyond the evaluated system and study. The following sections identify the validation steps required for future work.

10.1 Knowledge Base Construction and Scope

The knowledge graph schema and entity mappings are manually defined, while LLM-generated Cypher statements are manually reviewed before execution. This provides control over relation quality but limits the scale of the graph. The vector store is also similarly bounded by the CTI reports and historical cases that have been ingested. Consequently, the system can only retrieve knowledge represented within the current knowledge base.

Future work should evaluate the automated extraction of entities and graph-population methods while retaining source provenance and validation. During prototype development, GLiNER was explored as a possible approach for identifying cybersecurity entities from unstructured text and showed promising results. Such a model could support automatic entity extraction and graph population, but would require validation. Automated graph outputs should be compared against manually reviewed graph content using measures such as relation accuracy, knowledge coverage, and schema violations.

The current knowledge base also represents only a limited set of threats and historical investigations. A larger deployment requires a continuous ingestion of new CTI and validation SOC cases as threat knowledge and organizational experience evolve. This introduces additional requirements for deduplication, source quality, and removal of outdated knowledge.

10.2 Model, Retrieval, and Prompt Dependence

The prototype uses a single language model, Gemma 4 31B, and a single embedding model, Qwen3-Embedding-4B. No controlled model comparison is performed. Therefore, the evaluation cannot separate behaviour caused by the proposed architecture from behaviour specific to these model choices. Future work should keep the investigation pipeline and evaluation cases constant while comparing alternative locally deployable language and embedding models.

Retrieval and generation also use fixed design choices including the CTI and case chunking strategies, retrieval depth (top-k), hybrid retrieval configuration (temperature and output tokens), and the structured prompt template. These settings are not changed in the current evaluation. The results demonstrate the behaviour of the implemented configuration rather than suggesting that these configuration settings are optimal. Controlled experiments should compare various chunking and retrieval configurations, alongside retrieval strategies. Different structured prompting strategies should also be evaluated against the same cases to determine if there is any deviation in terms of grounding and unsupported claims.

10.3 Analyst-Controlled Case Context

The assistant only receives evidence that has been explicitly included within the investigation request. It does not autonomously search the wider Security Onion environment for additional and related telemetry. This preserves analyst control over the information supplied to the model, but it also means that it is possible that relevant evidence is missing from the investigation request, if the analyst has not yet identified it. As a result, output quality is partially dependent on the completeness of the case context supplied by the analyst.

The response assessment also showed that reasonable recommendations were not always valid due to a lack of available telemetry in the environment. Future versions should make available data sources explicit to the generation pipeline, so that the assistant can perform immediate checks, instead of recommending unavailable data sources. This will also force the assistant to prioritize available data sources.

Another future extension should investigate mechanisms for identify potentially relevant evidence from the active case or wider SIEM data, while retaining analyst approval over what is supplied in the investigation request. Such an approach could suggest additional telemetry for inclusion, rather than giving the assistant unrestricted access to the operational environment.

10.4 Evaluation Validity

The user study includes eight analysts from a single commercial SOC, comprising of six Tier 1 and two Tier 2 analysts. All participants have prior experience with Security Onion. This provides relevant feedback from users familiar with the target workflow, but limits generalization to analysts from other SOCs, experience levels, and technical environments. A broader evaluation should include participants from multiple organizations and SOC environments.

The study also evaluates the prototype using a single malware investigation scenario. Therefore, the results show how analysts interact with the system for this specific case rather than demonstrating whether there is equivalent behaviour for various incident types. The use of a ready-made hypothesis also introduces a risk of confirmation bias, since participants may focus on validating the hypothesis they are given, rather an independently testing it. Future evaluations should include cases with varying levels of evidence availability and different threat categories, including cases where the initial hypothesis is incorrect, or where available evidence remains sparse and inconclusive.

Qualitative coding was also performed by a single researcher using the predefined coding framework. This provides consistency in applying the schema but does not test agreement between independent coders. Future work could use a second coder for all of a sample of the experimental material to assess coding consistency and resolve any ambiguous classifications.

The study does not include a non-assistant control condition. With only eight participants, dividing the sample space into assistant and non-assistant groups would have produced very small groups. Within these groups, it would have been difficult to differentiate between an assistant effect versus individual variances. A within-subject evaluation using the same scenario would also be

inconclusive because participants would already know the hypothesis and outcome during the second condition. Finally, constructing independent and equivalent investigation cases is also challenging since they can differ substantially in difficulty. The study can characterize analyst experience, report outcomes, confidence changes, and interactions with the prototype, but it cannot establish that the system improves investigation accuracy or the workload relative to the existing workload. A controlled and larger follow-up study is required to compare matched investigations with and without the assistant while accounting for learning effects between these changing experimental conditions.

10.5 Operational Validation

The study evaluates short-term use of the prototype under controlled conditions and does not establish sustained operational value within a live SOC. A long-term deployment is required to examine how the system affects real investigations over repeated use. This type of study could measure investigation time relative to an appropriate baseline, analyst workload, repeated-use behaviour, adoption, and the types of cases for which analysts choose to use or ignore the assistant. This would extend the present user study toward the type of longer-term usage analysis reported by Singh et al. [16].

This type of operational deployment would also introduce engineering requirements that are outside the scope of the current prototype, including robust failure handling, asynchronous processing, access control, controlled knowledge-base updates, and session memory storage so that the LLM can maintain a history of the case. These upgrades should be developed as deployment requirements rather than contributions from this study.

11 Conclusion

This paper investigated how SOC analysts can be supported in accessing and operationalizing relevant incident knowledge during alert and case investigations. To address this problem, the proposed SOC AI Investigation Assistant combines semantic retrieval of CTI and historical SOC cases, a cybersecurity knowledge graph, and an analyst-facing phase framework. Its integration into Security Onion allows analysts to request structured investigative guidance within an existing case while keeping current evidence separate from supporting context through the investigation pipeline.

The response assessment findings indicate that responses contained plausible phase assessments and that evidence was properly grounded. It also showed that the usefulness of investigative actions was less consistent, since many of the recommendations depended on unavailable telemetry. These results show that further improvements are required to the pipeline to provide both grounded and useful suggestions to analysts.

The qualitative results provide insights into how analysts experienced the prototype. Participants appreciated the separation of information and identified value in the suggested investigative actions. They also described the assistant to be most appropriate as a secondary perspective or confirmation mechanism within a SOC workflow. However, they also reported that some actions were overly specific and repetitive. There was also a negative reaction to the prototype integration within Security Onion, although this is specific to the limitations of the software.

Taken together, these results suggest that SOC analysts can be supported by making external and historical incident knowledge accessible within an active investigation. This reusable knowledge can be further converted into structured investigative guidance. The hybrid knowledge base provides a mechanism for reusing external threat intelligence and documented historical investigation knowledge, while the investigation pipeline separates contextual information from current information. The user study indicates potential for this approach, but does not establish that the assistant improves investigation accuracy, workload, or investigation time. A long-term study with more participants and a control group is required to research these effects.

Overall, the results support the use of the proposed system as a human-in-the-loop decision-support approach for SOCs instead of an analyst replacement. The prototype demonstrates a practical way to retrieve and reuse incident knowledge while keeping the analyst responsible evidence selection, verification, and final incident decisions.

12 AI Statement

During the development of this thesis, Generative AI was used for code generation for knowledge graph traversal, Cypher queries, and managing edits to the native Security Onion Go code. Select unit tests and documentation for backend functions was generated, while output schemas and prompts were all created manually. When using AI to generate code, the author provided specific requirements and pseudocode, examined the generated output, and adapted their functions as needed. Select LaTeX formatting code was also generated with AI assistance. In addition, structural brainstorming was done by bouncing ideas off Generative AI. All diagrams have been manually constructed using draw.io. The author has independently verified all generated outputs and authored the final text. The author assumes full responsibility for all content and conclusions presented in this work.

Acknowledgments

The author would like to thank Dr. Luca Allodi for his supervision and guidance.

References

[1] Bradley Duncan. *Malware-Traffic-Analysis.net - 2026-01-31 - Traffic analysis exercise: Lumma in the room-ah*. https://www.malware-traffic-analysis.net/2026/01/31/index.html. Accessed: 2026-07-12. 2026.

[2] Bradley Duncan. *Malware-Traffic-Analysis.net - 2026-02-28 - Traffic analysis exercise: Easy as 123*. https://www.malware-traffic-analysis.net/2026/02/28/index.html. Accessed: 2026-07-12. 2026.

[3] Peng Cai, Reza Ryan, and Nickson M. Karie. *LLMLogAnalyzer: A Clustering-Based Log Analysis Chatbot using Large Language Models*. 2025. arXiv: 2510.24031 [cs.AI]. URL: https://arxiv.org/abs/2510.24031.

[4] Yupeng Chang et al. "A survey on evaluation of large language models". In: *ACM transactions on intelligent systems and technology* 15.3 (2024), pp. 1–45.

[5] Selina Y. Cho, Jassim Happa, and Sadie Creese. "Capturing Tacit Knowledge in Security Operation Centers". In: *IEEE Access* 8 (2020), pp. 42021–42041. DOI: 10.1109/ACCESS.2020.2976076.

[6] Derek B. Cornish. "The Procedural Analysis of Offending and Its Relevance for Situational Prevention". In: 1994, pp. 151–196. URL: https://popcenter.asu.edu/sites/g/files/litvpz3631/files/problems/stolen_goods/PDFs/Cornish1994.pdf.

[7] Yunfan Gao et al. "Retrieval-augmented generation for large language models: A survey". In: *arXiv preprint arXiv:2312.10997* 2.1 (2023), p. 32.

[8] Ali Habibzadeh, Farid Feyzi, and Reza Ebrahimi Atani. "Large Language Models for Security Operations Centers: A Comprehensive Survey". In: *arXiv preprint arXiv:2509.10858* (2025).

[9] Lei Huang et al. "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions". In: *ACM Transactions on Information Systems* 43.2 (2025), pp. 1–55.

[10] Eric M Hutchins, Michael J Cloppert, Rohan M Amin, et al. "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains". In: *Leading Issues in Information Warfare & Security Research* 1.1 (2011), p. 80.

[11] Leon Kersten et al. "'Give Me Structure': Synthesis and Evaluation of a (Network) Threat Analysis Process Supporting Tier 1 Investigations in a Security Operation Center". English. In: *Proceedings of the 19th Symposium on Usable Privacy and Security, SOUPS 2023*. 19th Symposium on Usable Privacy and Security, SOUPS 2023, SOUPS 2023 ; Conference date: 07-08-2023 Through 08-08-2023. Usenix Association, Aug. 2023, pp. 97–111.

[12] Leon Kersten et al. "A Security Alert Investigation Tool Supporting Tier 1 Analysts in Contextualizing and Understanding Network Security Events". In: *2024 Annual Computer Security Applications Conference (ACSAC)*. 2024, pp. 890–905. DOI: 10.1109/ACSAC63791.2024.00076.

[13] Kai Liu et al. "Recent progress of using knowledge graph for cybersecurity". In: *Electronics* 11.15 (2022), p. 2287.

[14] Calvin Nobles. "Stress, burnout, and security fatigue in cybersecurity: A human factors problem". In: *HOLISTICA–Journal of Business and Public Administration* 13.1 (2022), pp. 49–72.

[15] Ciyuan Peng et al. "Knowledge graphs: Opportunities and challenges". In: *Artificial intelligence review* 56.11 (2023), pp. 13071–13102.

[16] Ronal Singh et al. *LLMs in the SOC: An Empirical Study of Human-AI Collaboration in Security Operations Centres*. 2025. arXiv: 2508.18947 [cs.CR]. URL: https://arxiv.org/abs/2508.18947.

[17] Chengyu Song et al. *Audit-LLM: Multi-Agent Collaboration for Log-based Insider Threat Detection*. 2024. arXiv: 2408.08902 [cs.CR]. URL: https://arxiv.org/abs/2408.08902.

[18] Siddhant Srinivas et al. "AI-Augmented SOC: A Survey of LLMs and Agents for Security Automation". In: *Journal of Cybersecurity and Privacy* 5.4 (2025). ISSN: 2624-800X. DOI: 10.3390/jcp5040095. URL: https://www.mdpi.com/2624-800X/5/4/95.

[19] Elham Tabassi. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. en. 2023-01-26 05:01:00 2023. DOI: https://doi.org/10.6028/NIST.AI.100-1. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=936225.

[20] Shahroz Tariq et al. "Alert fatigue in security operations centres: Research challenges and opportunities". In: *ACM Computing Surveys* 57.9 (2025), pp. 1–38.

[21] The MITRE Corporation. *MITRE ATT&CK: Design and Philosophy*. https://attack.mitre.org. Accessed: 2026-06-12. 2026.

[22] Jack Tilbury and Stephen Flowerday. "Humans and Automation: Augmenting Security Operation Centers". In: *Journal of Cybersecurity and Privacy* 4.3 (2024), pp. 388–409. ISSN: 2624-800X. DOI: 10.3390/jcp4030020. URL: https://www.mdpi.com/2624-800X/4/3/20.

[23] Melissa Turcotte, François Labrèche, and Serge-Olivier Paquette. *Automated Alert Classification and Triage (AACT): An Intelligent System for the Prioritisation of Cybersecurity Alerts*. 2025. arXiv: 2505.09843 [cs.CR]. URL: https://arxiv.org/abs/2505.09843.

[24] Manfred Vielberth et al. "Security Operations Center: A Systematic Study and Open Challenges". In: *IEEE Access* 8 (Dec. 2020), pp. 227756–227779. DOI: 10.1109/ACCESS.2020.3045514.

[25] Manfred Vielberth et al. "Security Operations Center: A Systematic Study and Open Challenges". In: *IEEE Access* 8 (2020), pp. 227756–227779. DOI: 10.1109/ACCESS.2020.3045514.

[26] Bowen Wei et al. *CORTEX: Collaborative LLM Agents for High-Stakes Alert Triage*. 2025. arXiv: 2510.00311 [cs.CL]. URL: https://arxiv.org/abs/2510.00311.

[27] Jie Zhang et al. "When llms meet cybersecurity: A systematic literature review". In: *Cybersecurity* 8.1 (2025), p. 55.

A User Study Scenarios

A.1 Sparse-Evidence Demonstration Scenario

The demonstration scenario used the packet capture from the Malware-Traffic-Analysis.net exercise *Easy as 123*, published on the 28th of February 2026 [2]. The original exercise contains traffic related to a NetSupportManager RAT.

The packet capture was imported into Security Onion and used as a separate demonstration case. Its purpose was to introduce the SOC AI Investigation Assistant and demonstrate how the generated response behaves when the available case evidence was limited. It was not included in any evaluation.

A.2 Evaluation Scenario

The evaluation scenario used the packet capture from the Malware-Traffic-Analysis.net exercise *Lumma in the Room-ah*, published on the 31st of January 2026 [1]. The original exercise contains network traffic associated with the Lumma Stealer infostealer.

This packet capture was imported into Security Onion for the user study and presented as the case to investigate. Instead of answering the original exercise questions, particpants received an initial alert and a predefined hypothesis. The selected alert and hypothesis are listed below.

> **Alert**: ET MALWARE Win32/Lumma Stealer Related CnC Domain in DNS Lookup (whitepepper .su)
>
> **Hypothesis**: A Windows host is compromised by Lumma Stealer and is communicating with Lumma-associated infrastructure.

B Structured Prompt

The following prompt was used by the investigation pipeline during the evaluation. Runtime placeholders were replaced with the current case evidence, analyst context, retrieved CTI, and knowledge-graph context.

```
## Role
Act as a senior SOC analyst performing evidence-based
↪ alert triage and investigation planning. Use
↪ conservative reasoning. Separate observed case
↪ evidence from interpretation and from background
↪ threat intelligence. Do not fill evidence gaps with
↪ assumptions.

## Task
Produce the best-supported current-behavior hypothesis
↪ for the case. Use the alert, analyst context, and
↪ log evidence as primary evidence. Use retrieved
↪ CTI, KG, or historical context only as background
↪ support.

Reason backwards from the current behavior to identify
↪ three concrete preconditions or preceding
↪ conditions that should be verified. For each
↪ precondition, specify the exact evidence to look
↪ for and the exact data source or telemetry needed.

The output should help an analyst decide what to check
↪ next, not declare the incident fully understood.

## Evidence Hierarchy
Use this evidence priority order:
1. Log evidence from the current case.
2. Analyst context from the current case.
3. Alert name and alert metadata.
4. Retrieved CTI, KG, or historical context.

Alert names and detection-rule names are not proof of
↪ the full malware lifecycle. A rule name can suggest
↪ a behavior, but the behavior is only confirmed if
↪ supported by case evidence.

Malware capability in CTI is not evidence that the
↪ capability occurred in this case.

## Attack Narrative
Use the following phases to orient the observed
↪ behavior:

Preconditions:
Environmental or attacker-side conditions that had to
↪ exist before the observed activity could occur,
↪ such as exposed services, weak controls, user
↪ interaction, prior malware execution, successful
↪ DNS resolution, or unblocked outbound access.

Entry:
How the attacker or malware first gained access, if
↪ supported by evidence. Do not infer the entry
↪ vector from CTI alone.

Action Thread:
Observed attacker or malware activity after entry. This
↪ has two subphases:

- Establishing Foothold:
  Activity consistent with establishing or preparing
  ↪ control, such as first C2 lookup, first C2
  ↪ connection, agent registration, beaconing,
  ↪ persistence setup, discovery, credential-access
  ↪ preparation, or tool staging.

- Operational Execution:
  Activity directed toward the objective, such as
  ↪ confirmed collection, credential theft,
  ↪ exfiltration, lateral movement, propagation,
  ↪ destructive action, or attacker tasking over an
  ↪ established channel.

Impact:
Confirmed damage, theft, access, disruption, or
↪ business effect.

Residual Presence:
Persistence, compromised identities, cleanup traces, or
↪ remaining attacker access.

For DNS-only or weak network evidence involving known
↪ C2 infrastructure, prefer "Action Thread" and
↪ "Establishing Foothold" only if the evidence
↪ supports suspected initial C2 lookup, first
↪ contact, or beaconing. Do not classify collection,
↪ exfiltration, persistence, or impact unless
↪ directly supported.

## Case Inputs

### Alert
{alert}

### Analyst Context
{analyst_context}

### Log Evidence
{log_context}

### Retrieved Context
{context_block}
```

```
## Source Basis Rules
The source_basis field must contain only short origin items that are actually available in the prompt.

Allowed primary source labels:
- "alert"
- "analyst_context"
- "log_evidence" only if log evidence is provided

For retrieved context, only cite labels that appear literally in the Retrieved Context section, such as:
- "[CTI-1] <short description>"
- "[KG-1] <short description>"
- "[CASE-1] <short description>"

Do not invent source labels.
Do not write "KG - ..." unless the retrieved context contains an explicit KG-labelled item.
Do not write "CTI - ..." unless the retrieved context contains an explicit CTI-labelled item.
If retrieved context is present but unlabeled, cite it only as "retrieved_context".

## Reasoning Rules
Use cautious language.

Use "observed" only for facts directly present in alert, analyst context, or logs.
Use "consistent with" or "suggests" for interpretations supported by retrieved context.
Use "unconfirmed" for anything not directly supported.

Do not claim that a behavior, phase, impact, persistence mechanism, lateral movement, compromise, malware execution, credential theft, or exfiltration occurred unless it is supported by alert, analyst context, or log evidence.

Do not say:
- "successful compromise" unless execution, installation, authentication success, or confirmed C2 session evidence exists
- "data exfiltration" unless outbound transfer, exfil destination, payload, volume pattern, or exfil-specific alert evidence exists
- "persistence" unless registry, service, scheduled task, startup folder, implant reinstall, or equivalent evidence exists
- "spearphishing" unless email, URL-delivery, attachment, user-click, or message evidence exists
- "lateral movement" unless internal authentication, remote execution, SMB, RDP, WinRM, SSH, or similar evidence exists

If evidence is limited to DNS, HTTP, TLS, firewall, or IDS records, focus on network-observable behavior and recommend the exact additional evidence needed to confirm host compromise or malware execution.

If the logs show attempted but unsuccessful activity, such as NXDOMAIN responses, HTTP 404 responses, blocked connections, incomplete TCP handshakes, reset connections, or denied firewall actions, reflect this uncertainty in the phase assessment and confidence.

## Investigation Rows
Return exactly 3 investigation_rows.

Each row must contain:
- precondition: a concrete preceding condition or requirement that would make the current behavior plausible
- evidence_to_look_for: specific evidence that would confirm or reject that precondition
- relevant_data_source: the exact telemetry, log type, field, device record, or service log to inspect

The three rows must not be generic. Avoid broad phrases such as "check SIEM logs", "review EDR", or "analyze network logs". Name the specific log or fields.

Prefer data sources appropriate to the evidence in the case. If only network evidence is present, prefer Zeek DNS, Zeek conn, Zeek SSL, Zeek HTTP, Suricata alert metadata, proxy logs, firewall logs, DNS resolver logs, and DHCP logs. Use Windows endpoint telemetry only when the asset type or case context supports it.

Good relevant_data_sources examples:
- Zeek dns.log fields query, rcode_name, answers, id.orig_h
- Zeek conn.log fields id.orig_h, id.resp_h, id.resp_p, history, orig_bytes, resp_bytes
- Zeek ssl.log fields server_name, established, ja3, ja3s
- Zeek http.log fields method, host, uri, status_code, request_body_len, response_body_len
- Suricata alert fields rule.name, source.ip, destination.ip, network.community_id
- Proxy or firewall logs with destination.domain, destination.ip, action, bytes_out, bytes_in
- Windows Sysmon Event ID 1 fields Image, CommandLine, ParentImage, Hashes

## Current Behavior Rules
The current_behavior field must be a conservative behavior label. Prefer "Suspected ..." when the evidence is limited.

For example:
- "Suspected DNS-based C2 infrastructure lookup"
- "Suspected initial C2 contact"
- "Confirmed HTTP agent registration to suspected C2 endpoint"
- "Confirmed fingerprinting data submission to suspicious infrastructure"

Do not use broad labels such as "malware infection", "credential theft", "data exfiltration", or "successful compromise" unless the case evidence directly supports them.

## Confidence Assessment
Assign one overall confidence level.

Use:
- high: The current behavior and investigation rows are directly supported by alert, analyst context, logs, and retrieved context, with no major unsupported assumptions.
- medium: The current behavior is supported by case evidence, but some preconditions or interpretations rely on reasonable inference from CTI or retrieved context.
- low: The response relies mostly on alert wording or background CTI, with weak or missing support from logs or analyst context.
```

```
Case evidence has higher weight than retrieved context.
 ↪  CTI and KG context can support interpretation, but
 ↪  they do not prove the behavior occurred in the
 ↪  current case.

In supported_claims, list only claims directly
 ↪  supported by alert, analyst context, logs, or
 ↪  retrieved context.
In uncertain_claims, list important assumptions that
 ↪  still need confirmation.

## Output Requirements
Return compact JSON only.
Do not include markdown.
Do not include keys outside the expected schema.
Return exactly 3 investigation_rows.
Return exactly 3 next_steps.
Keep narrative_summary between 80 and 120 words.
Keep justification concise.
If broader_phase is not "Action Thread", set
 ↪  action_thread_subphase to "N/A".
```

C Security Onion Prototype Interface

Figure 13 provides the complete AI Investigation Assistant interface. Cropped views of the individual interface components are presented in the main System Design section.

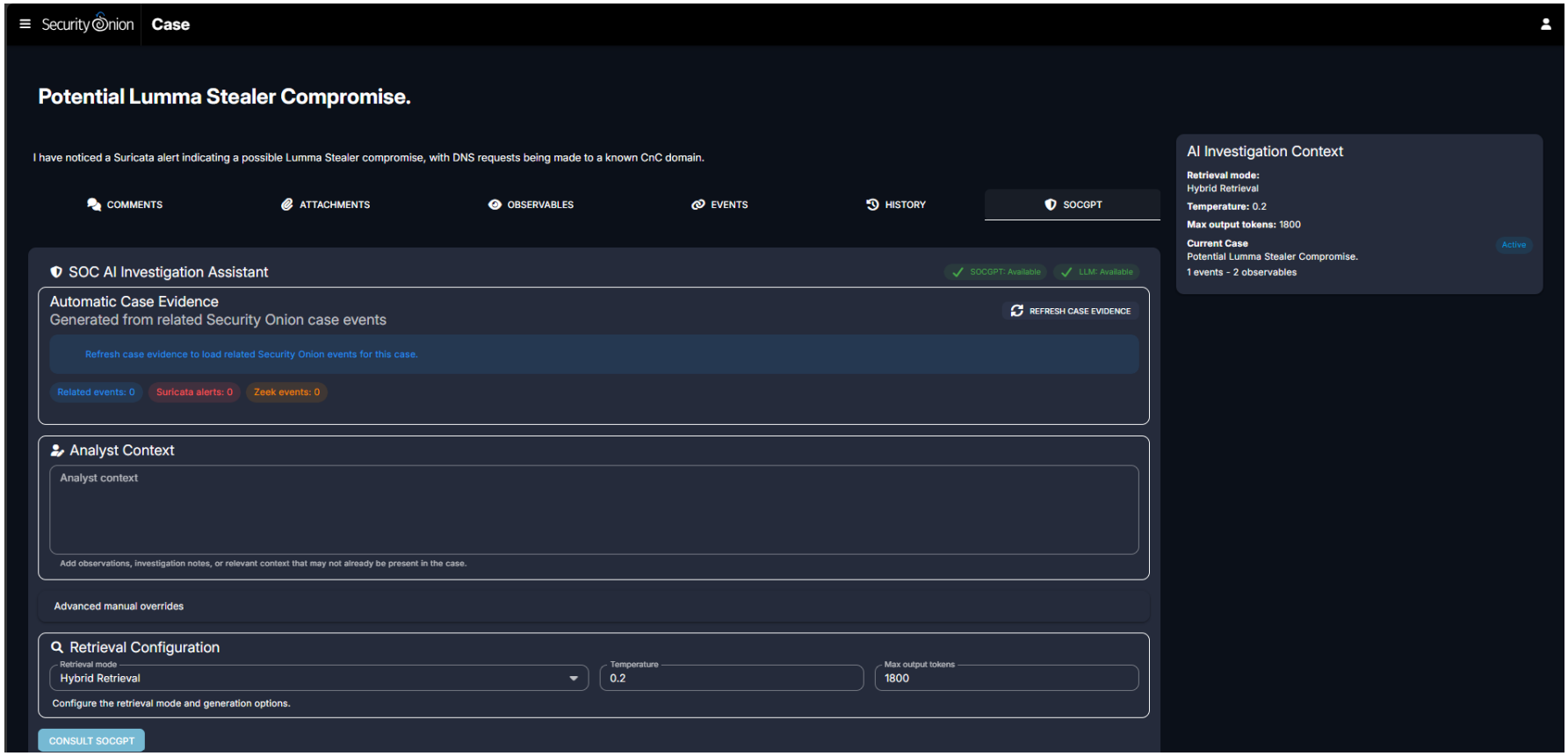


Figure 13: Complete Security Onion AI Investigation Assistant interface, including case evidence, analyst input, configuration controls, generated investigation support, and supporting information panels.