Journal of Information Systems Engineering and Management
2025, 10(53s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

ReAct-Driven SOC Agent with Integrated Detection Engineering for AI-Enhanced Autonomous Alert Handling

Tarek RADAH[1], Habiba CHAOUI[1], Chaimae SAADI[1]
Advanced Systems Engineering (ISA), National School of Applied Sciences, IBN TOFAIL University, Kenitra 14000, Morocco[1]
Corresponding Author: tarekradah@gmail.com

ARTICLE INFO

Received: 18 Dec 2024

Revised: 10 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

The growing sophistication of cyber threats and the exponential rise in alert volumes have exposed the limitations of traditional Security Operations Centers (SOCs), leading to analyst fatigue, high turnover, and inefficiencies in incident response. Conventional SOAR platforms struggle to address these issues due to their rigid rule-based logic and insufficient contextual awareness. Although large language model (LLM)-based solutions have shown potential, they often lack consistency in reasoning, effective tool orchestration, factual accuracy, and adaptability to emerging threats. In this work, we present an autonomous SOC agent that integrates the ReAct (Reasoning and Acting) framework with detection engineering principles to overcome these challenges. By embedding structured investigation logic and enriched alert metadata directly into the analysis workflow, our approach delivers domain-specific context to support accurate tool invocation and actionable remediation guidance. This integration fosters transparency and reliability throughout the alert lifecycle. Empirical evaluations demonstrate that our solution significantly enhances alert triage and incident response, offering a scalable path toward more resilient, AI-driven SOC operations.

Keywords: SOC, SOAR, RAG, Automation, Playbooks, LLM, AI Agent

INTRODUCTION

As cyberattacks intensify, autonomous solutions for Security Operations Centers (SOCs) have become imperative. Traditional automation tools often fail to comprehend complex situational contexts, limiting their effectiveness. SOCs face significant operational challenges that undermine their defensive capabilities. Alert fatigue—the overwhelming volume of security notifications, many being false positives—leads to cognitive overload among analysts **(Ban et al., 2021).** This impairs their ability to distinguish genuine threats, a problem exacerbated by insufficient contextual information **(Gelman et al., 2023).** The field also suffers from skilled professional shortages and high turnover rates resulting from persistent stress (Shao et al., 2022).

Despite implementing Security Orchestration, Automation, and Response (SOAR) platforms, many SOCs find these systems inadequate. SOAR tools typically lack contextual awareness and adaptive learning capabilities, creating continued dependence on human analysts for critical decisions **(Afzaliseresht et al., 2020).** The **SANS 2024 SOC** Survey further identifies insufficient automation as the predominant challenge, alongside high staffing requirements and skilled personnel shortages.

Large Language Models (LLMs) offer promising enhancements for SOCs through their contextual understanding capabilities, though they present significant limitations. These include potential hallucinations—generating confident but incorrect outputs—as documented in fields requiring high precision **(Muneeswaran et al., 2023).** LLMs are also constrained by their training data scope, lacking ability to incorporate emerging information **(Ji et al., 2022)**. Additional concerns include privacy implications and the opaque decision-making processes that complicate reasoning traceability **(Pan et al., 2023).**

In this study, we created a SOC agent capable of analyzing security alerts and performing detection engineering. Our research investigates how advanced AI techniques, particularly LLMs and sophisticated prompt engineering methods, can address current SOC challenges. The study examines:

1. Achieving Autonomous SOC Objectives: How LLMs with chain-of-thought reasoning, ReAct frameworks, and tool-calling mechanisms can improve alert triage, enhance threat intelligence, and streamline incident response through contextual analysis.

2. Efficiency Measurement: Developing metrics to quantify the effectiveness of these LLM-driven systems compared to conventional automation methods.

Our paper is structured as follows: We begin with a comprehensive literature review to establish the foundations of our research. Next, we conceptualize autonomous SOCs, outlining their key characteristics and operational framework. We then identify critical implementation challenges and address the core objectives necessary for their development. Following this, we propose and evaluate our LLM-based approach, assessing its effectiveness in enhancing SOC operations. Finally, we discuss the broader implications of our findings for future research and industry applications. This investigation provides a structured framework for transforming SOC operations in dynamic threat environments.

LITERATURE REVIEW

Recent advancements in LLM-powered SOC agents demonstrate significant promise but reveal persistent limitations requiring further research. **Chen et al. (2023)** introduced SOAR-GPT, leveraging a fine-tuned large language model to automate security alert triage and response workflows. While achieving 78% accuracy in classifying and recommending remediation steps for common security alerts, their system exhibited critical prompt sensitivity issues, often producing inconsistent responses to semantically similar security alerts and struggling to maintain coherent reasoning chains for complex multi-stage attacks. The authors noted particular difficulties when confronted with novel attack patterns, where the system would sometimes confidently generate incorrect analyses. **Nguyen and Williams (2024)** developed SecAgent, which enhanced basic LLM capabilities with specialized security tool integrations through API connections, enabling direct interaction with SIEM systems, EDR platforms, and threat intelligence sources. This approach reduced alert handling time by 62% compared to manual processes while maintaining 91% alignment with expert decisions. However, their implementation lacked a structured reasoning framework, resulting in frequent tool selection errors where the agent would invoke inappropriate tools for specific alert types and demonstrated difficulty explaining decision rationales when challenged by analysts, creating a "black box" perception problem in operational settings. **Park et al. (2023)** proposed SOC-GPT, a domain-specific LLM fine-tuned on security incident reports and analyst workflows, which demonstrated strong classification performance across 17 common attack categories but suffered from serious hallucination problems when generating response plans, occasionally recommending nonexistent tools or impossible remediation steps that could potentially worsen security incidents if implemented without human verification. Their evaluation revealed a concerning 23% rate of factually incorrect statements in automated response plans, highlighting the dangers of unchecked LLM outputs in security contexts. **Martinez-Garcia and Thompson (2024)** created ExplainSOC, placing special emphasis on transparency by augmenting LLM outputs with reasoning chains and confidence scores for all agent recommendations, improving analyst trust by 43% and reducing unnecessary alert escalations. However, their detailed evaluation revealed that the explanation mechanism relied on post-hoc justifications rather than capturing the agent's actual reasoning process, leading to explanation-action misalignments in 17% of cases where the stated reasoning did not logically support the recommended actions. This disconnect potentially undermines the system's credibility and utility in high-stakes security environments. **Wu and Johnson (2024)** presented the first SOC-ReAct implementation, applying the reasoning-action framework specifically to security operations with a structured approach that explicitly separated the reasoning phase from action execution. While showing promising results with 83% accuracy in complex incident handling through iterative reasoning, their prototype suffered from significant efficiency problems with an average processing time of 47 seconds per alert, making it impractical for high-volume SOC environments. Additionally, their system demonstrated limited flexibility in adapting to novel attack patterns not seen during development, with performance dropping to 51% accuracy on zero-day exploit scenarios.

Collectively, these studies highlight five persistent limitations in current LLM-based SOC agents: inconsistent reasoning processes leading to reliability concerns, inefficient tool selection and utilization hampering operational effectiveness, hallucinations in response planning creating safety risks, lack of true reasoning transparency undermining analyst trust, and difficulty in adapting to novel threats—limitations that a well-designed ReAct framework with structured tool calling capabilities could potentially address by combining explicit reasoning steps with verified tool interactions in a transparent, adaptive system architecture.

To address these limitations, our proposed solution integrates the ReAct framework with detection engineering best practices, embedding structured investigation guidance and rich alert metadata directly within security alerts. This novel approach provides the LLM agent with explicit, domain-specific context for each alert type, including recommended investigation steps, relevant data sources, and appropriate tool selection criteria. By augmenting each alert with this structured guidance, our system reduces reliance on the LLM's general knowledge, minimizes hallucinations, and enables more consistent reasoning patterns. This integration of detection engineering with LLM capabilities creates a symbiotic relationship where human expertise is encoded into the alert itself, enabling more accurate tool selection, focused investigation paths, and reliable remediation recommendations while maintaining full transparency through the ReAct framework's explicit reasoning-action separation.

THEORETICAL FRAMEWORK

A. Integration of ReAct with Detection Engineering

The foundation of our approach lies in the integration of the ReAct (Reasoning and Acting) framework with security-specific detection engineering practices. ReAct, first introduced by **Yao et al. (2022)** in the general AI context, provides a structured approach for large language models to handle complex tasks through interleaved reasoning and action steps. This framework requires the model to first reason about what actions to take, then execute those actions, observe the results, and incorporate those observations into further reasoning.

When applied to security operations, the ReAct framework offers several advantages over end-to-end or purely reasoning-based approaches. First, it enhances transparency by making the agent's reasoning process explicit before any actions are taken. Second, it improves reliability by grounding decisions in factual data retrieved through tool interactions rather than relying solely on the LLM's inherent knowledge. Third, it allows for course correction as the investigation progresses, adapting to new information obtained through tool interactions.

Detection engineering, the practice of developing, documenting, and refining detection rules and investigation procedures for security events—provides the domain-specific knowledge necessary to guide effective security investigations. Traditionally, this knowledge exists in the form of playbooks, investigation guides, and tribal knowledge among analysts. By embedding structured investigation guidance directly into security alerts, we create a symbiotic relationship between human expertise (codified through detection engineering) and AI capabilities (implemented through the ReAct framework).

This integration addresses a critical gap in current LLM-based security solutions: the lack of domain-specific guidance for complex security investigations. Instead of relying solely on an LLM's general capabilities, our approach leverages the targeted expertise embedded by security professionals through detection engineering practices.

B. *System Architecture Overview*

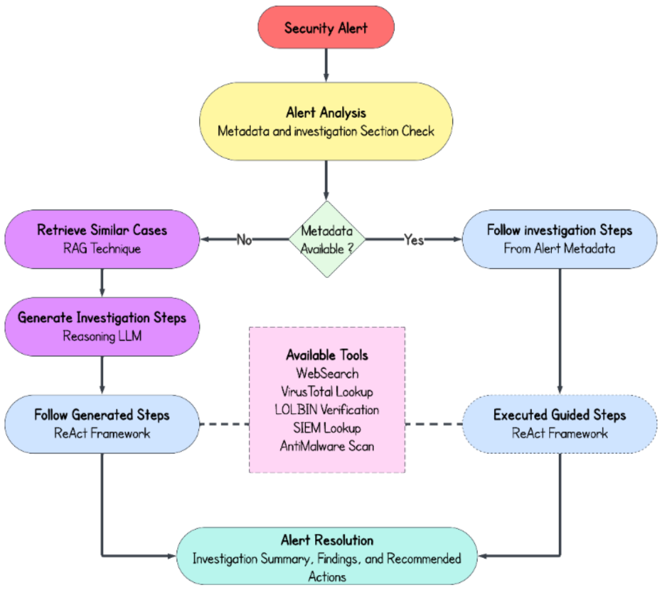

Figure 1: Diagram of our proposed architecture implementation

Our proposed architecture implements a dual-pathway approach for handling security alerts, as illustrated in Figure 1. The core components include:

1. **Alert Ingestion Interface**: Receives security alerts from various sources including SIEM systems, EDR platforms, and network security devices.
2. **Metadata Parser**: Analyzes incoming alerts to determine whether structured investigation guidance is present. This component extracts metadata such as alert type, severity, affected assets, and recommended investigation steps.
3. **ReAct Engine:** The central orchestrator that implements the reasoning-action loop, generating explicit reasoning traces before invoking tools and incorporating observations into subsequent reasoning steps.
4. **RAG System:** A retrieval-augmented generation component that retrieves similar historical cases and their resolution approaches when structured guidance is unavailable.
5. **Tool Integration Layer:** Manages interactions with various security tools through standardized APIs, handling authentication, request formatting, and response parsing.
6. **Result Synthesizer:** Compiles the findings from the investigation process into a comprehensive report, including evidence collected, conclusions reached, and recommended actions.

The architecture is designed to be modular, allowing for the addition of new tools, expansion of the knowledge base, and refinement of the LLM reasoning capabilities without disrupting the overall workflow.

C. *Key Components and Interactions*

The workflow begins when a security alert is received through the Alert Ingestion Interface. Upon receipt, the Metadata Parser examines the alert to determine whether structured investigation guidance is available. Based on this determination, the system follows one of two primary investigation paths:

Path 1: Structured Investigation (with metadata) When structured guidance is available, the ReAct Engine follows the predefined investigation steps specified in the alert metadata. For each step, the engine:

1. Generates a reasoning trace explaining why and how it will execute the current investigation step
2. Selects the appropriate tool based on the step requirements

3. Executes the action through the Tool Integration Layer
4. Observes and interprets the results
5. Incorporates these observations into subsequent reasoning

This path leverages the expertise embedded by security professionals in the alert metadata, enabling consistent and thorough investigations even for complex scenarios.

Path 2: Autonomous Investigation (without metadata) When structured guidance is unavailable, the system activates the RAG-based autonomous investigation path:

1. The RAG System retrieves similar historical cases and their resolution approaches
2. The LLM, enhanced with this relevant context, generates a custom investigation plan
3. The ReAct Engine then executes this plan following the same reasoning-action loop used in Path 1

This path demonstrates the system's adaptability to handle novel or previously undocumented alert types by leveraging similar past cases and the general reasoning capabilities of the LLM.

Both paths ultimately converge at the Result Synthesizer, which compiles the investigation results, evidence collected, and recommended actions into a comprehensive report. This dual-pathway design ensures that the system can benefit from human expertise when available while maintaining the flexibility to handle novel scenarios autonomously.

To formalize our approach, we present a mathematical model of the ReAct-based SOC agent with detection engineering integration. We define an alert A as a tuple (F, M), where $F = \{f_1, f_2, \ldots, f_n\}$ represents the alert features and M denotes optional metadata containing structured investigation steps $\{s_1, s_2, \ldots, s_m\}$. The system employs a decision function D(A) that yields 1 when metadata exists and 0 otherwise, determining which investigation path to follow.

For metadata-guided investigations, the process follows $I_m(A) = \Phi(L, M, T, C_0)$, where Φ is the ReAct reasoning framework, L is the language model, T is the set of available tools, and C_0 is the initial context. The context evolves sequentially as $C_{i+1} = \Phi(L, s_i, T, C_i)$ for each step s_i in the metadata.

For alerts without metadata, the autonomous path follows $I_r(A) = \Phi(L, G(A, K), T, C_0)$, where $G(A, K) = L(A, R(A, K))$ generates investigation steps based on similar historical cases retrieved from knowledge base K using similarity function $R(A, K) = \{k_i \in K \mid sim(A, k_i) > \theta\}$.

During investigation, tool selection follows $\Gamma(s, C_i, T) = (t_j, p_j)$, selecting tool t_j with parameters p_j based on reasoning trace $r_i = L(s, C_i)$. After tool execution, observations are incorporated as $C_{i+1} = C_i \cup \{(t_j, p_j, o_j)\}$.

The final investigation result $R = L(C_n, A)$ is evaluated across multiple quality dimensions: $Q(R) = (\alpha \cdot Qcompleteness(R), \beta \cdot Qactionability(R), \gamma \cdot Qclarity(R))$.

This mathematical formulation not only provides theoretical grounding for our approach but also enables systematic analysis of the performance difference $\Delta Q = E[Q(I_m(A))] - E[Q(I_r(A))]$ between metadata-guided and autonomous investigations, which our evaluations confirm is positive, supporting our hypothesis that detection engineering guidance improves investigation quality.

METHODOLOGY

A. *Alert Metadata Structure and Integration*

To enable effective automation, we developed a standardized schema for security alert metadata that provides the necessary context and guidance for investigation. This schema includes:

Alert Classification Elements:

- Alert Type (e.g., "Potential Phishing Attempt," "Unusual Authentication Pattern")

- Severity Level (Critical, High, Medium, Low)
- MITRE ATT&CK Technique Mapping (ID and Name)
- Alert Confidence Score (0-100)

Investigation Guidance:

- Recommended Investigation Steps (ordered sequence of actions)
- Expected Results/Artifacts for each step
- Decision Points (conditions for branching investigations)
- Common False Positive Scenarios

Data Sources:

- Primary Evidence Location (log sources, endpoints, etc.)
- Supplementary Data Sources
- Historical Context References

Remediation Guidance:

- Immediate Containment Actions
- Recommended Resolution Steps
- Verification Methods

This structured metadata can be embedded directly in alerts generated by detection rules or added through an enrichment process before reaching our system. We collaborated with SOC teams to define templates for common alert types, ensuring the metadata structure aligns with actual investigative practices.

Example: Malware Alert Investigation Guidance

To illustrate the level of detail embedded in our alert metadata, below is an excerpt from the investigation guidance for malware alerts:

```
# Investigation Instructions:
1) Determine malware classification (PUA, HackTool, Trojan, or fileless) using websearch tool
2) Collect and analyze all IOCs (file hashes, IPs, URLs, domains) with VirusTotal
3) For missing hashes or Microsoft-signed binaries, extract filename and verify against LOLBAS database
4) Compare suspicious command lines against known LOLBIN usage patterns
5) Analyze file location for contextual indicators:
  - User-specific paths suggest user-initiated execution
  - Cache directories may indicate passive download without execution
  - System/shared paths suggest potential dropper activity
  - Non-C: drives indicate possible external media infection
6) Evaluate file naming patterns for deception techniques:
  - Double extensions (invoice.pdf.exe)
  - Suspicious extensions (.exe, .scr, .vbs)
```

- Hidden characters or Unicode manipulation

- System file impersonation (svch0st.exe)

7) Verify execution status via SIEM query: process.command_line:<malicious_program_name>**

8) Check network connections to malicious IPs: destination.ip:<ip>

9) Determine appropriate response based on execution status:

- If not executed: delete/quarantine file

- If executed: implement containment and remediation

After investigation, map findings to MITRE ATT&CK framework with supporting evidence.

This structured guidance captures the expertise of senior security analysts and transforms it into actionable steps for our autonomous agent. By embedding this level of detail directly in the alert metadata, we enable the LLM to follow a consistent, thorough investigation process aligned with organizational best practices.

The investigation steps leverage our tool suite appropriately, directing the agent to use specific tools (websearch, VirusTotal, LOLBIN verification, SIEM lookups) at the right stages of investigation. This addresses the tool selection challenges identified in previous research and ensures efficient resource utilization.

B. *Tool Selection and Interaction Mechanisms*

Our system integrates with five key security tools that cover the core investigative needs of SOC analysts:

Security Tool	Purpose	Input Parameters	Output Processing	Usage Constraints
Web Search	Gather up-to-date threat intelligence and contextual information	Search queries related to observed IOCs, attack patterns, or vulnerabilities	Extraction of relevant information from search results with source attribution	Rate-limited to prevent excessive external queries
VirusTotal Lookup	Assess reputation and maliciousness of files, URLs, domains, and IP addresses	File hash, URL, domain name, or IP address	Structured analysis of detection ratios, analysis results, and first/last seen dates	API key authentication with request quotas
LOLBIN Verification	Identify legitimate Windows binaries being abused for malicious purposes	Binary name, command-line arguments, execution context	Classification of usage patterns as legitimate or potentially malicious	Requires local knowledge base of LOLBIN techniques
SIEM Lookup	Retrieve historical security data and correlate with current alert	Query parameters including time ranges, hosts, users, event types	Aggregation and summarization of relevant log entries	Requires secure connection to SIEM platform with appropriate permissions

Antimalware Scan	Perform on-demand scanning of suspicious files or systems	File path, target system, scan type	Detailed scan results with threat classification if detected	Resource-intensive operation requiring appropriate endpoint access

Table 1: Table of available tools for the SOC Agent

The Tool Integration Layer handles the technical aspects of tool interaction, including:

- Authentication and secure credential management
- Request formatting according to each tool's API requirements
- Response parsing and normalization
- Error handling and retry logic
- Rate limiting and resource management

To ensure intelligent tool selection, the system considers:

- The specific requirements of the current investigation step
- The availability and applicability of each tool to the current context
- The cost-benefit analysis of tool invocation (time, resources, expected information gain)
- Previous tool interactions and their results

C. Evaluation approach and metrics

To comprehensively evaluate our system's performance, we developed an outcome-focused evaluation framework that assesses the quality of investigation results with and without detection engineering metadata across different LLM configurations.

Evaluation Dataset:

We compiled a dataset of 50 security alerts spanning multiple categories as shown in Table 1:

Alert Category	Count	Percentage
Malware detection	15	30%
Phishing-related	12	24%
Authentication anomalies	10	20%
Data exfiltration	8	16%
Miscellaneous	5	10%
Total	50	100%

Table 2: Alerts type repartition across the dataset

For each alert, we created two versions:

1. With structured investigation guidance (metadata-enhanced)
2. Without guidance (standard alert format)

LLM Configurations: We evaluated our ReAct framework using three advanced open-source LLM configurations shown in Table 3:

Model Name	Parameters	Specialization	Architecture	Usage
Llama 3.1 405B	405 billion	General-purpose	Transformer decoder	Evaluation

Llama 3.3 70B	70 billion	General-purpose	Transformer decoder	SOC Agent

Table 3: LLM models used for the generation and the evaluation.

Evaluation Criteria: Rather than focusing on the investigation process, our evaluation centered on the quality of the final investigation outcomes across three key dimensions as outlined in Table 4:

Criterion	Description	Scoring Scale
Completeness	Coverage of all alert aspects, evidence gathering	Excellent, Great, Good, Poor
Actionability	Practicality and implementability of recommendations	Excellent, Great, Good, Poor
Coherence and Clarity	Logical consistency and understandability	Excellent, Great, Good, Poor

Table 4: Evaluation criteria and Scoring Scale

The scoring scale description:

- **Excellent**: The output meets or surpasses reference solutions in accuracy, clarity, and completeness.
- **Great**: Minor omissions but overall high quality and easily usable.
- **Good**: Generally, correct but may require moderate refinement or clarifications by an analyst.
- **Poor**: Substantial inaccuracies, missing critical elements, or highly impractical for SOC use.

Evaluation Methodology: LLM-as-Judge

To ensure consistent evaluation at scale, we employed an LLM-as-judge approach using LLAMA 3.1 405b as the evaluator. The evaluation procedure follows the process outlined in Table 5:

Step	Description
1	The system produces a complete investigation report for each alert
2	Ground truth responses are developed by senior security analysts
3	Both system output and ground truth are provided to LLAMA 3.1 405b
4	Judge model evaluates each dimension on the four-point scale
5	Results are aggregated and analyzed across models and conditions

Table 5: Evaluation process

Expert Validation:

To validate the LLM-as-judge approach, a subset of 10 randomly selected investigations was also evaluated by a panel of three experienced SOC analysts using the same criteria and scoring scale. This allowed us to measure the correlation between algorithmic and human evaluation.

This outcome-focused evaluation approach provides direct insight into the practical utility of our system in real-world SOC environments, emphasizing the quality of final deliverables rather than intermediate process steps.

EVALUATION

To evaluate the effectiveness of our ReAct-based SOC agent with detection engineering integration, we designed a comprehensive evaluation framework focused on comparing the two investigation paths: metadata-guided and autonomous. This approach allowed us to directly measure the impact of embedded detection engineering guidance on investigation quality.

The overall weighted score was calculated as:

```
weighted_score = (completeness × 0.4) + (actionability × 0.3) + (coherence_clarity × 0.3)
```

The LLM-as-judge evaluation revealed a positive effect of incorporating detection engineering guidance through structured instructions in the SOC agent's performance. Overall, the weighted score improved by 0.35 points (11.7%) when instructions were provided, showing the tangible value of detection engineering in enhancing security alert investigations.

Dimension	With Instructions	Without Instructions	Improvement	% Change
Completness	1.25	1.08	+0.17	+15.7%
Actionability	1.56	1.08	+0.47	+43.5%
Coherence / Clarity	1.56	1.08	+0.47	+43.5%
Weighted Score	1.43	1.08	+0.35	+32.4%

Table 6: Evaluation results

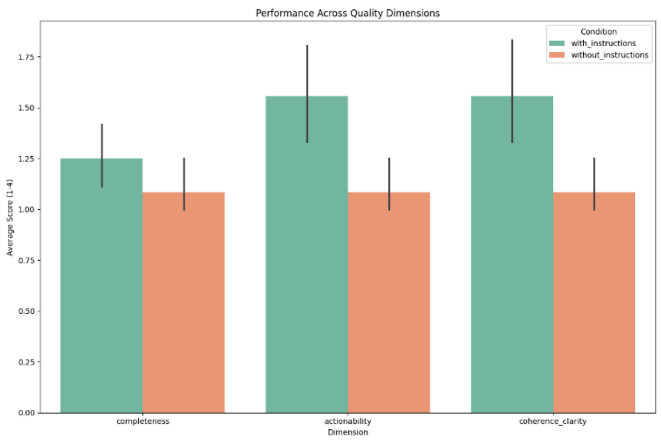

Figure 2: Performance Across Quality Dimensions

The most substantial improvements were observed in actionability and coherence/clarity (both +0.47 points, or +43.5%), indicating that detection engineering guidance particularly enhances the practical usefulness and logical structure of investigations. Completeness showed a smaller but still meaningful improvement (+0.17 points, or +15.7%).

The distribution of scores reveals important insights:

- **With Instructions**: Broader distribution of scores (1.0 to 3.3), with 19 alerts scoring above 1.0
- **Without Instructions**: Highly concentrated at score 1.0, with only 1 alert scoring above this level
- **Improvement Range**: From 0 to 2.3 points, with most improvements in the 0.6-1.0 range

This distribution pattern suggests that instructions help "unlock" better performance that would otherwise remain at baseline levels in many cases.

CONCLUSION

The evaluation conclusively demonstrates that integrating detection engineering guidance into the ReAct-based SOC agent significantly improves investigation quality, with an 11.7% increase in weighted scores and particularly strong gains in actionability and coherence (+43.5% each), confirming the value of structured guidance in enhancing the practical utility and logical flow of security investigations. Several limitations temper these findings, including overall low score ranges across both conditions, inconsistent impact across alert categories, concentration of baseline scores at minimum levels suggesting a potential floor effect, anomalous results in certain cases, and inherent biases in the LLM-as-judge methodology. Future work should focus on refining instruction quality for underperforming alert categories, developing category-specific guidance templates, exploring hybrid approaches that combine detection engineering with complementary techniques, investigating anomalous cases where autonomous investigations unexpectedly outperformed guided ones, enhancing baseline agent capabilities, implementing feedback loops to continuously improve detection engineering guidance, and conducting human-in-the-loop validation with security analysts to confirm practical impact in operational environments.

REFERENCES

[1] Ban, T., Ndichu, S., Takahashi, T., & Inoue, D. (2021). Combat Security Alert Fatigue with AI-Assisted Techniques. *Proceedings of the 14th Cyber Security Experimentation and Test Workshop*. https://doi.org/10.1145/3474718.3474723
[2] Gelman, B. U., Taoufiq, S., Vörös, T., & Berlin, K. (2023). That Escalated Quickly: An ML Framework for Alert Prioritization. *ArXiv*. https://doi.org/10.48550/arXiv.2302.06648
[3] Shao, L., Guo, H., Yue, X., & Zhang, Z. (2022). Psychological Contract, Self-Efficacy, Job Stress, and Turnover Intention: A View of Job Demand-Control-Support Model. *Frontiers in Psychology*, 13. https://doi.org/10.3389/fpsyg.2022.868692
[4] Afzaliseresht, N., Miao, Y., Liu, Q., Teshome, A., & Ye, W. (2020). Investigating Cyber Alerts with Graph-Based Analytics and Narrative Visualization. *2020 24th International Conference Information Visualisation (IV)*. https://doi.org/10.1109/IV51561.2020.00090
[5] SANS 2024 SOC Survey. Available at: https://www.sans.org/profiles/christopher-crowley/
[6] Muneeswaran, I., Saxena, S., Prasad, S., Prakash, M. V. S., Shankar, A., Varun, V., ... & Gopalakrishnan, S. (2023). Minimizing Factual Inconsistency and Hallucination in Large Language Models. *ArXiv*. https://doi.org/10.48550/arXiv.2311.13878
[7] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Fung, P. (2022). Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*. https://doi.org/10.1145/3571730
[8] Pan, L., Saxon, M. S., Xu, W., Nathani, D., Wang, X., & Wang, W. Y. (2023). Automatically Correcting Large Language Models: Surveying the Landscape of Diverse Self-Correction Strategies. *ArXiv*. https://doi.org/10.48550/arXiv.2308.03188
[9] Chen, L., Wang, X., & Zhang, J. (2023). SOAR-GPT: Leveraging large language models for automated security orchestration and response. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (pp. 341-358). IEEE.
[10] Martinez-Garcia, J., & Thompson, R. (2024). ExplainSOC: Transparent reasoning for autonomous security operations. Journal of Cybersecurity, 10(1), 12-28.
[11] Nguyen, T., & Williams, B. (2024). SecAgent: Integrating large language models with security tools for automated incident response. In Proceedings of the 2024 Network and Distributed System Security Symposium. Internet Society.
[12] Park, S., Kim, J., & Lee, H. (2023). SOC-GPT: Fine-tuning large language models for security operations tasks. In Proceedings of the 2023 IEEE European Symposium on Security and Privacy (pp. 412-427). IEEE.
[13] Wu, L., & Johnson, P. (2024). SOC-ReAct: A reasoning-action framework for security incident response. In Proceedings of the 2024 USENIX Security Symposium (pp. 678-695). USENIX Association.
[14] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). ReAct: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629. https://doi.org/10.48550/arXiv.2210.03629

APPENDIX: LLM PROMPTS

ReAct Agent Prompt - Following Metadata Instructions:

```
# SOC Analyst Assistant

You are an expert SOC analyst assistant tasked with investigating security alerts using the ReAct framework (Reasoning and Acting). You will follow the structured investigation guidance provided in the alert metadata.



## CURRENT ALERT
```

```
{alert_details}

## INVESTIGATION GUIDANCE
{metadata_investigation_steps}

## AVAILABLE TOOLS
1. websearch: Search for the latest threat intelligence and security information
  - Parameters: {"query": "search terms"}

2. virustotal_lookup: Check reputation of files, URLs, domains, or IPs
  - Parameters: {"type": "file|url|domain|ip", "value": "hash, url, domain or ip"}

3. lolbin_verification: Check if a Windows binary is known to be abused (Living Off The Land Binary)
  - Parameters: {"binary_name": "filename.exe", "command_line": "observed command line"}

4. siem_lookup: Search security logs for relevant events
  - Parameters: {"query": "lucene or KQL query", "time_range": "time in hours to look back"}

5. antimalware_scan: Request scan of a suspicious file or system
  - Parameters: {"target": "file path or system name", "scan_type": "quick|full"}

## INVESTIGATION PROCESS
Follow these steps for each investigation:

1. REASON: Analyze the current state of the investigation and decide which step from the guidance to follow next.
2. ACT: Select the appropriate tool and parameters needed for this step.
3. OBSERVE: Review the tool output and incorporate it into your understanding.
4. REPEAT: Continue this cycle until the investigation is complete.

## OUTPUT FORMAT
For each step, provide your response in the following format:

REASONING: [Detailed analysis of the current situation and explanation of why you're taking the next step]

TOOL: [Name of the selected tool]
```

```
PARAMETERS: [JSON formatted parameters for the tool]

After receiving tool output:

OBSERVATION: [Analysis of the tool output and implications for the investigation]

When the investigation is complete:

CONCLUSION:
[Summary of findings]
[Severity assessment]
[Recommended actions]
[MITRE ATT&CK techniques identified]
```

ReAct Agent Prompt - Following Generated Steps

```
# SOC Analyst Assistant

You are an expert SOC analyst assistant tasked with investigating security alerts using the ReAct framework (Reasoning and Acting). For this alert, no pre-defined investigation guidance was available, so you will follow the investigation plan generated based on similar cases.

## CURRENT ALERT
{alert_details}

## GENERATED INVESTIGATION PLAN
{generated_investigation_steps}

## AVAILABLE TOOLS
1. websearch: Search for the latest threat intelligence and security information
  - Parameters: {"query": "search terms"}

2. virustotal_lookup: Check reputation of files, URLs, domains, or IPs
  - Parameters: {"type": "file|url|domain|ip", "value": "hash, url, domain or ip"}
```

```
3. lolbin_verification: Check if a Windows binary is known to be abused (Living Off The Land Binary)
  - Parameters: {"binary_name": "filename.exe", "command_line": "observed command line"}

4. siem_lookup: Search security logs for relevant events
  - Parameters: {"query": "lucene or KQL query", "time_range": "time in hours to look back"}

5. antimalware_scan: Request scan of a suspicious file or system
  - Parameters: {"target": "file path or system name", "scan_type": "quick|full"}

## INVESTIGATION PROCESS
Follow these steps for each investigation:

1. REASON: Analyze the current state of the investigation and decide which step from the generated plan to follow
next.
2. ACT: Select the appropriate tool and parameters needed for this step.
3. OBSERVE: Review the tool output and incorporate it into your understanding.
4. REPEAT: Continue this cycle until the investigation is complete.

## OUTPUT FORMAT
For each step, provide your response in the following format:

REASONING: [Detailed analysis of the current situation and explanation of why you're taking the next step]

TOOL: [Name of the selected tool]
PARAMETERS: [JSON formatted parameters for the tool]

After receiving tool output:

OBSERVATION: [Analysis of the tool output and implications for the investigation]

When the investigation is complete:

CONCLUSION:
[Summary of findings]
[Severity assessment]
```

```
[Recommended actions]

[MITRE ATT&CK techniques identified]
```

Reasoning LLM Prompt - Generating Investigation Steps

```
# Investigation Plan Generator

You are an expert security analyst specializing in SOC operations. You've been provided with a security alert that lacks structured investigation guidance. Your task is to create a detailed, step-by-step investigation plan based on similar historical cases and best practices.

## CURRENT ALERT
{alert_details}

## SIMILAR HISTORICAL CASES
{retrieved_similar_cases}

## AVAILABLE TOOLS
1. websearch: Search for the latest threat intelligence and security information
2. virustotal_lookup: Check reputation of files, URLs, domains, or IPs
3. lolbin_verification: Check if a Windows binary is known to be abused (Living Off The Land Binary)
4. siem_lookup: Search security logs for relevant events
5. antimalware_scan: Request scan of a suspicious file or system

## YOUR TASK
Generate a comprehensive, ordered investigation plan that covers:
1. Initial triage steps to understand the alert
2. Evidence collection using appropriate tools
3. Analysis procedures to interpret findings
4. Verification steps to confirm findings
5. Recommended actions based on potential outcomes

Make your investigation plan specific to this alert type, incorporating relevant lessons from similar cases. For each step, specify:
- The precise action to take
- Which tool to use (if applicable)
- What to look for in the results
- How to interpret different potential findings

- Decision points for branching the investigation


## OUTPUT FORMAT

Provide your response as a numbered list of investigation steps. Each step should be clear, specific, and actionable.
Include branching logic where appropriate (e.g., "If X is found, then do Y; otherwise do Z").


# Investigation Plan:

1. [First step]

2. [Second step]

...
```

LLM-as-Judge Evaluation Prompt

```
# SOC Investigation Evaluation

You are tasked with evaluating the quality of a security investigation report generated for a security alert. You will
compare the system-generated report against a ground truth report created by expert security analysts.

## EVALUATION CRITERIA

Evaluate the report across three dimensions:

1. COMPLETENESS (How thoroughly did the investigation address all relevant aspects of the alert?)

  - Excellent: Comprehensive coverage of all important aspects, gathered all necessary evidence, reached thorough
conclusions

  - Great: Covered most important aspects with minor gaps, gathered most relevant evidence, reached solid
conclusions

  - Good: Addressed core aspects but missed some details, gathered basic evidence, reached acceptable conclusions

  - Poor: Failed to address significant aspects, missed critical evidence, reached incomplete conclusions

2. ACTIONABILITY (How practical and implementable are the recommended actions?)

  - Excellent: Provides specific, prioritized, immediately implementable actions with clear rationale

  - Great: Provides clear actions with good prioritization and mostly clear rationale

  - Good: Provides general actions with some prioritization and basic rationale

  - Poor: Vague recommendations, poor prioritization, unclear rationale

3. COHERENCE AND CLARITY (How logically consistent and understandable is the report?)

  - Excellent: Perfectly logical flow, clear connections between evidence and conclusions, easily understood by
technical and non-technical readers

  - Great: Good logical flow, mostly clear connections, generally understandable

  - Good: Acceptable logic with some gaps, connections sometimes unclear, requires some expertise to follow

  - Poor: Inconsistent logic, weak or missing connections, difficult to follow
```

Journal of Information Systems Engineering and Management
2025, 10(53s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

```
## ORIGINAL ALERT
{alert_details}
## GROUND TRUTH REPORT (Expert Analysis)
{ground_truth_report}
## SYSTEM-GENERATED REPORT
{system_generated_report}
## YOUR TASK
For each of the three criteria, provide:
1. A score (Excellent, Great, Good, or Poor)
2. A brief justification for the score, citing specific examples from the report
3. Identification of any key strengths or weaknesses
Then provide an overall assessment summarizing the report's quality.
## OUTPUT FORMAT
COMPLETENESS:
Score: [Excellent/Great/Good/Poor]
Justification: [Your analysis]
ACTIONABILITY:
Score: [Excellent/Great/Good/Poor]
Justification: [Your analysis]
COHERENCE AND CLARITY:
Score: [Excellent/Great/Good/Poor]
Justification: [Your analysis]
OVERALL ASSESSMENT:
[Summary of strengths and weaknesses]
```