Experiences of Using Agentic AI to Fill Tooling Gaps in a Security Operations Center

Kritan Banstola*
University of South Florida
kbanstola@usf.edu

Faayed Al Faisal*
University of South Florida
faayedal@usf.edu

Xinming Ou
University of South Florida
xou@usf.edu

***Abstract*—Large language models (LLMs) are attracting interest from Security Operations Centers (SOCs), but their practical value and limitations remain largely unexplored. In this work, cybersecurity researchers are embedded as entry-level SOC analysts in a university SOC to observe the day-to-day workflows and explore how LLMs can fit into existing SOC practices. We observed that analysts frequently handle large volumes of similar alerts while manually pivoting across heterogeneous and disjoint tools — including SIEMs, OSINT services, and internal security tools. Each tool provides part of the required analysis given a ticket, but the tools cannot easily work together to resolve a ticket without requiring manual effort to integrate the results from the disparate tools. This gap between the tools results in a repetitive and time-consuming workflow that slows down investigations and contributes to analyst burnout. Based on these observations, we designed and implemented an LLM-driven ReAct agent capable of unifying these disparate tools and automating routine triage tasks such as log retrieval, enrichment, analysis, and report generation. We evaluated the system on real SOC tickets and compared the agent's performance against manual analyst workflows. We further experimented with how iterative prompting and additional analyst instructions can refine the agent's reasoning and improve response quality. The results show that our agent effectively reproduces several routine analyst behaviors, reduces manual effort, and demonstrates the potential for human-AI collaboration to streamline alert triage in operational SOC environments.**

I. Introduction

Security Operations Centers (SOCs) are flooded with a high volume of security alerts daily, most of which are false positives [1]. Yet for each of these alerts, in order to close the ticket an analyst must navigates through multiple disjointed tools present in the SOC environment, such as SIEM platforms, OSINT services, and internal security tools, to collect relevant information for triage and documentation. The context switching between different tools increases cognitive load on the analysts, which increases the risks of slow response and missed attacks.

Large Language Models (LLMs) offer a promising opportunity to streamline SOC workflows and reduce analyst workload. In recent years, researchers have explored the use of large language models (LLMs) across various cybersecurity tasks, including penetration testing [2], malware analysis [3], phishing detection [4], and so on. Some explores the use of AI to assist security analysts by automating investigation, triaging, and remediation tasks [5]. However, such general-purpose design may not fit well with specific workflows and toolchains of individual SOCs. Each SOC adopts its own playbooks, workflows, and technology stack – from SIEM platforms to required OSINT tooling – resulting in substantial operational variation across environments. Therefore, it is important to design an AI system that can adapt to the unique context and requirements of each SOC.

In this paper, we present SOC AI Companion, a locally deployed LLM-based assistant that is designed to assist SOC analysts with alert triage. Our system is built on a ReAct-style agent architecture [6], which combines chain of thought reasoning with dynamic tool invocation. When triaging an alert, the agent plans investigative steps, retrieves logs from different data sources, performs enrichment using internal and external tools (e.g., SIEM queries, OSINT lookups), and documents findings into structured reports which are then reviewed by human analysts. The analyst can then further prompt the agent by providing more context or instructions to refine the investigation. The agent is mainly driven by two components: (1) a system prompt that encodes SOC workflows and practices, and (2) a set of tool integrations that allow the agent to interact with SOC tools and data sources. Additionally, the entire system runs locally within the SOC to ensure sensitive data does not leave the organization's network. This design allows the agent to adapt to specific workflows and toolchains of individual SOCs. We believe that this approach of human-AI collaboration can enhance the efficiency and effectiveness of SOC operations while maintaining human oversight and judgment.

To design an effective SOC AI Companion, we needed to understand the workflows, tools, and practices of analysts in a real-world SOC. To this end we embed cybersecurity researchers as entry-level analysts in a university SOC to observe and learn how analysts perform their daily job, by doing the same job themselves. This follows the anthropological approach to study SOCs by participant observation

* Shared first authorship

Workshop on Security Operation Center Operations and Construction (WOSOC) 2026
23 February 2026, San Diego, CA, USA
ISBN 978-1-970672-03-9
https://dx.doi.org/10.14722/wosoc.2026.23016
www.ndss-symposium.org

pioneered by Sundaramurthy [7]. We observed that even routine alert tickets often required significant time and effort to gather evidence across multiple tools, and write structured reports. Based on these observations, we identified specific areas where an LLM-based agent could assist analysts by automating these routine tasks. We evaluated the performance of our SOC AI Companion using real SOC alert tickets, and compared the time taken to triage them with and without the agent's involvement. Additionally, we also explored how prompt iteration – where analysts provide additional context and instructions to the agent – can improve the quality of the agent's responses.

Our contributions are as follows:

- We design and implement an LLM-based "SOC AI Companion" that can assist analysts in triaging alerts, through participant observation in a real SOC.
- We report empirical findings based on actual SOC alert tickets, showing a significant decrease in triage time when using the LLM assistant compared to manual triage.
- We discuss the implications of our findings for future research and development of LLM agents in SOCs.

The remainder of this paper is organized as follows. In Section II, we discuss the organization and workflows in the SOC. In Section III, we present the architecture of our SOC AI Companion. In Section IV, we describe our experimental setup and findings. In Section V, we discuss related works, and Section VII concludes the paper and outlines future research directions.

II. Background: SOC Organization and Workflows

In this section, we describe the operational context of the SOC where we conducted the fieldwork. This includes its organizational structure, roles, tooling ecosystem, and alert-handling workflow. This provides the baseline for understanding where and how an LLM agent can contribute.

A. Research Ethics

At this phase of our research, our participant observation is limited to using the fieldwork to understand how the SOC analysts do their jobs. Our interactions with the analysts are confined to the purpose of learning how to do the job, and we use the researchers' own experience of doing the same job to inform our design of the AI SOC Companion. The experimentation of the resulting AI Companion was all carried out by the researchers themselves.

In the future we plan to invite the SOC analysts to use our AI Companion and gather feedbacks from the SOC analysts as to how effective the AI Companion is. For this future stage of research, we have started the IRB process so that we will be able to use data collected from the analysts in our research. Our findings as presented in this paper does not involve any human subject data and are purely based on the researchers' own experience in the SOC.

B. SOC Structure and Roles

The University SOC employs students as their tier-1 analysts. These students are tasked with real security alerts and incidents, providing them with hands-on experience in cybersecurity operations. The SOC followed a traditional tiered structure, with the goal of efficiently managing alert volume and providing a clear career path for student analysts. Tier-1 (L1) analysts/students served as the initial point of contact for incoming alerts. Their primary responsibilities included reviewing alert metadata, performing basic enrichment, documenting findings, and determining whether further investigation was required. As the first line of defense, L1 analysts handled the alert volume and were expected to make decisions that would decide whether an alert could be closed or needed escalation.

Tier-2 (L2) analysts were responsible for deeper investigations and correlation of events that could not be resolved at the L1 level. Compared to L1 analysts who are students, L2 analysts have more access to the internal systems and tools to perform their investigations. They have a more comprehensive understanding of the organization's network architecture, common threat vectors, and incident response procedures. L2 analysts reviewed escalated alerts, conducted in-depth analysis, and determined the appropriate response actions.

There is a managerial layer overseeing both tiers, responsible for coordinating operations, ensuring adherence to protocols, and facilitating communication between the student SOC and the broader university IT and security teams. They are the initial reviewers of incident reports and are responsible for mentoring and training the student analysts.

Although this structure appears linear, the actual workflow was highly dynamic. Analysts frequently shifted between tasks, collaborated across tiers when necessary, and relied on undocumented operational knowledge developed from experience. This combination of role specialization and informal knowledge-sharing created important challenges for standardizing SOC workflows, especially in high-volume environments.

We focused our investigation on the L1 analyst role, as it represented the most common entry point for alerts and involved a significant amount of routine data gathering. This role also aligns with SOC L1 duties often associated with high levels of burnout.

C. L1 Analyst Work

L1 analysts were responsible for the initial review and triage of incoming security alerts. Their primary tasks included:

- Taking ownership of alerts: L1 analysts monitored the alert queue and claimed alerts for investigation. This involved prioritizing alerts based on severity, potential impact, and available context.
- Reviewing alert metadata: Analysts examined the details of each alert, including source and destination IP addresses, timestamps, event types, and any associated indicators of compromise (IOCs).

- Performing basic enrichment: Analysts gathered additional context about the alert by querying internal databases, threat intelligence feeds, and open-source intelligence (OSINT) resources. This often involved looking up IP reputation, domain information, and related threat actor activity.
- Documenting findings: Analysts recorded their observations, actions taken, and any relevant context in the alert ticketing system. This documentation served as a record for future reference and for escalation to L2 analysts if needed.
- Determining next steps: Based on their review, analysts decided whether the alert could be closed as a false positive or required further investigation by L2 analysts. This decision-making process often relied on heuristics and tacit knowledge developed through experience.

D. Tooling Ecosystem

The SOC utilized a variety of tools to support alert handling and investigation. The primary tool for alert management was a Security Information and Event Management (SIEM) platform, which aggregated logs from various sources and generated alerts based on predefined rules. The SIEM provided a centralized interface for analysts to review alerts, access metadata, and document their findings. In addition to the SIEM, analysts relied on several other tools for enrichment and investigation, including:

- Threat intelligence platforms: These platforms provided access to curated threat intelligence feeds, allowing analysts to look up IOCs and gather context about known threat actors and campaigns.
- OSINT resources: Analysts frequently used open-source tools and websites to enrich alerts with reputation and contextual data on domains, IPs, and infrastructure components.

III. System Architecture

In this section, we discuss the architecture of our SOC AI Companion. We have designed the system to assist SOC analysts in triaging and investigating security alerts by using a ReAct-based AI agent capable of reasoning and performing actions. The system consists of three major components: (1) Chat User Interface, (2) LLM Core, and (3) Tooling and Integration Layer. Figure 1 illustrates these three components and their interactions.

A. Chat User Interface

We have designed a web-based chat interface that allows SOC analysts to interact with the AI companion while working on a security alert ticket. Analysts can load tickets directly into the chat interface from their ticketing software (e.g., Zendesk, ServiceNow). The AI agent will then start investigating the ticket by gathering evidence, performing analysis, and documenting its findings in a structured report format. This structured report will include the final verdict (benign or malicious) for the alert, reasoning and analysis done by the agent, IOCs discovered, evidence collected from various sources, and finally recommendations for the alert. The analyst can review the report draft provided by the agent and validate the findings. If the analyst does not agree with the response, they can provide additional instructions or context to the agent to guide the investigation as needed. This allows the system to automate routine workflows like data gathering, enrichment, etc. and provide analysis and documentation while still keeping the human analyst in the loop for final decision making.

The analyst can also dynamically modify the system prompt directly from the UI and customize the agent's behavior to match specific investigation goals and local SOC specific policies.

B. LLM Core

Security Operations Centers (SOCs) handle highly sensitive data and therefore require that no data leaves the organization's trusted network boundary. This confidentiality requirement stops us from using standard cloud-based LLM APIs from third-party providers. We use `gpt-oss` [8], an open-weight reasoning model released by OpenAI, and deploy it on-premises. This open-source model displayed strong reasoning and agentic capabilities and therefore was an ideal choice for our system based on the constraints. We use the LangChain framework in Python to create a ReAct agent capable of reasoning and dynamic tool invocation. LangChain gives us clean abstractions for prompts, controlled ReAct style reasoning steps, dynamic and structured tool calls, which makes it easier to track the agent's reasoning process and tool invocations.

For any alert triage, the large language model interprets the incoming alert ticket. Then based on the system prompt, the LLM plans investigation steps, and reasons on what tools need to be invoked to gather relevant data and evidence. Then the model makes the necessary tool calls, analyzes the retrieved response from tools and repeats the reasoning and tool invocation step until all necessary data have been retrieved. Then the agent finally outputs a structured triage report, which will be reviewed by the analyst. The complete system prompt used by the AI companion can be found in the appendix.

C. Tooling and Integration Layer

The AI companion needs to make requests to various internal and external tools when investigating an alert. Therefore, we need to integrate these disparate tools into our agentic system. LangChain provides a standardized interface for binding tools to our LLM. For our system, we integrated three classes of tools essential for alert triage: OSINT tools, SIEM tools, and internal security tools. We defined and integrated tools for making requests to Virustotal, WHOIS, and Google search, as part of OSINT services to gather publicly available information on artifacts related to the alert. We integrated APIs for Splunk and Microsoft Sentinel to retrieve relevant logs on network and device events based on the alert. We also

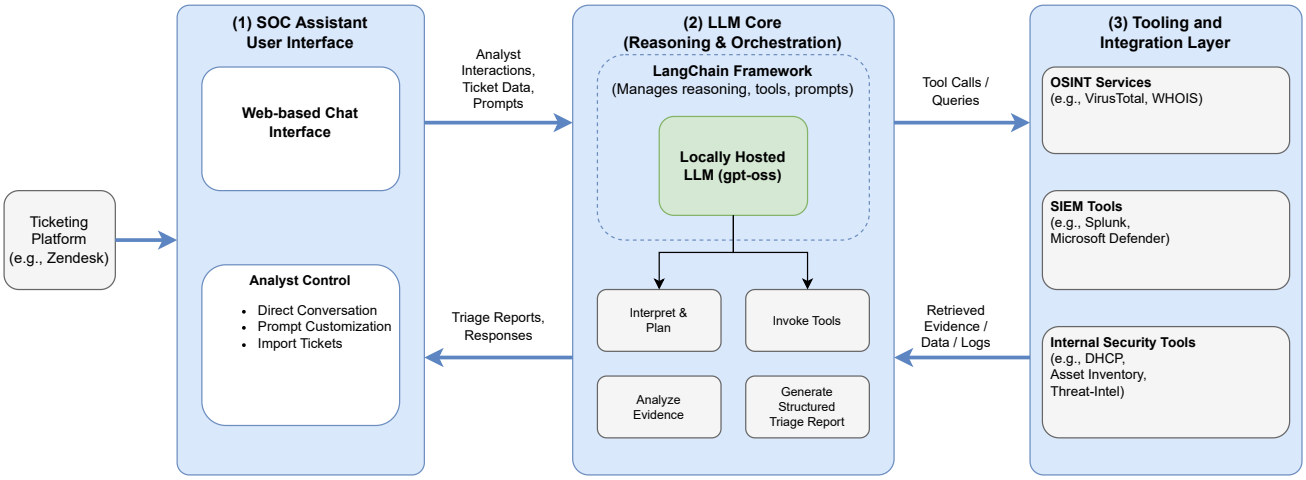

Fig. 1. System Architecture of our SOC AI Companion

integrated internal security tools specific to the SOC environment, including a DHCP lookup tool, an asset inventory lookup, and a threat intel service. Our system logs each tool interaction and it becomes part of the investigation record. This helps the analyst to review the investigation steps taken by the agent. The agent may perform multiple tool calls in parallel or sequential chains, depending on the alert's complexity.

IV. Experiments and Evaluation

The goal of our experimental evaluation is to assess the potential impact of an LLM-based assistant on alert triage tasks commonly performed by entry-level SOC analysts. Our focus is on understanding whether the AI assistant can reduce analyst effort, maintain decision quality, and operate with minimal prompting and intervention. We therefore evaluate the system along four primary dimensions: task completion time, response quality, prompting behavior, and consistency across multiple alert scenarios.

We conducted a series of controlled experiments using representative SOC alert tickets in real-world triage scenarios. For each ticket, we compared the time and effort required to complete triage manually against triage performed with the assistance of the LLM agent. All experiments were performed on actual SOC tickets that have been processed by the researchers deployed as entry-level analysts in the university SOC.

A. Experimental Setup

The experiments were conducted using a set of 8 tickets from the university SOC's alert queue split among the two researchers acting as entry-level analysts. Each ticket was processed twice: once manually without AI assistance and once with the LLM agent. The order of processing was swapped each time to control for learning effects that might arise from familiarity with the ticket content. For manual triage, the researchers followed their standard workflow, using the SOC's SIEM, OSINT tools, and internal resources to investigate and document their findings. For AI-assisted triage, the researchers interacted with the LLM agent via the chat interface described in Section III, providing the ticket details and any necessary instructions.

TABLE I
Time required to complete alert triage with and without LLM assistance, including execution order

Ticket	Order	Manual (min)	LLM-Assisted (min)
Ticket 1	Manual → LLM	18:14	8:02
Ticket 2	LLM → Manual	11:29	9:30
Ticket 3	Manual → LLM	17:35	7:46
Ticket 4	LLM → Manual	9:15	8:51
Ticket 5	Manual → LLM	14:57	6:56
Ticket 6	LLM → Manual	10:21	14:47
Ticket 7	Manual → LLM	12:48	5:01
Ticket 8	LLM → Manual	10:29	6:48

The tickets selected for the experiments were chosen to represent a range of common alert types encountered in the SOC, including alerts that the SOC has encountered frequently in the past. This ensured that the evaluation captured a realistic spectrum of triage challenges faced by entry-level analysts. Due to the nature of the experiments, the researchers were not aware of the specific alert types or the outcomes of their triage efforts until after the experiments were completed. They were only informed of the general type of alert. This setup helped to maintain a level of realism in the experiments by preventing the researchers from tailoring their responses to specific alert types.

Table I shows the time taken to investigate SOC tickets by the analyst when doing the work manually vs when triaging with our AI agent. The results show that, in 7 out of 8 tickets, LLM-assisted triage required less time than manual investigation. On average, AI-assisted triage reduced ticket completion time by approximately 40%. This shows that using AI agents can substantially reduce time and analyst effort for routine alert handling. The most pronounced time reductions were observed in tickets that required extensive cross-tool

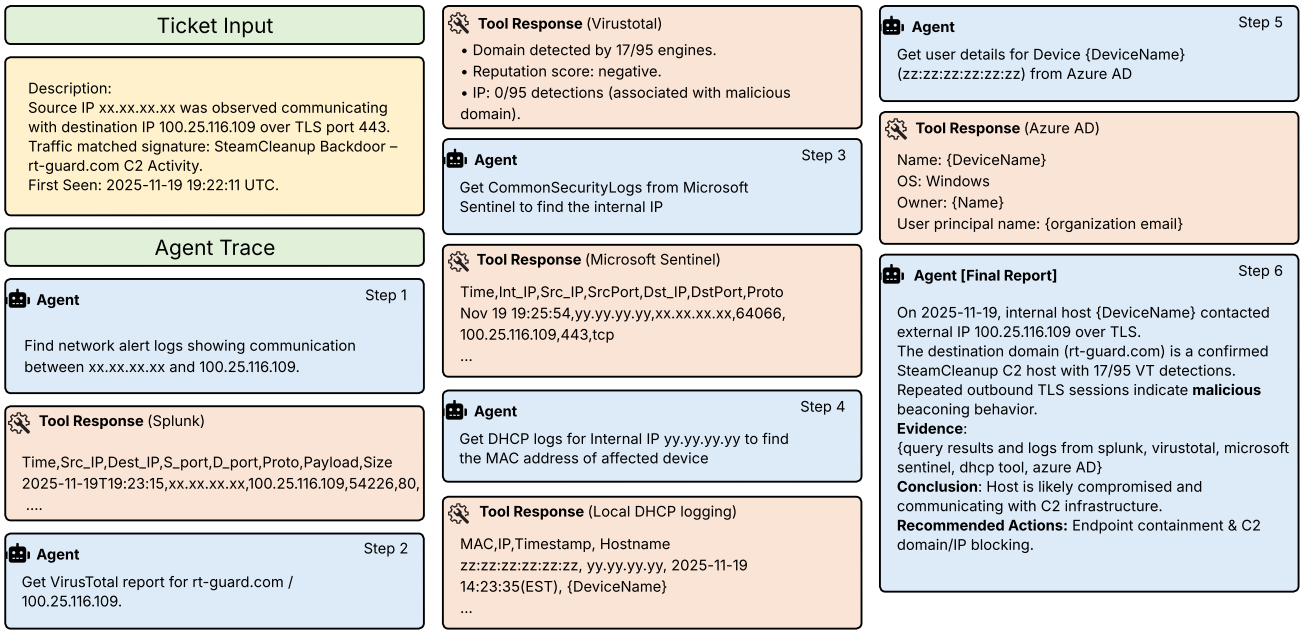

Fig. 2. An example of the agent successfully investigating a real SOC alert

pivoting for logs and evidence gathering. We observed only a slight decrease in triage time in cases where the ticket was first triaged using AI assistance and later handled manually. This is because the analyst was already familiar with the investigation context and evidence from the AI-assisted triage, making the subsequent manual process easier.

We observed an exception in Ticket 6, where LLM-assisted triage took longer than manual investigation. In this ticket, the LLM-assisted triage was performed first, and during the investigation, the payload from network traffic needed to be decoded and interpreted. This required more manual intervention because basic decoders were not able to decode the payload and required the tacit knowledge of the analyst. Later, when the same ticket was investigated manually, the analyst was able to directly decode the payload using established tooling, thanks to the experience from the prior process using the LLM, and thus resulted in faster completion time. This result does not mean that the AI Companion slowed down the analysis, but rather is an artifact from the order of the experiments. Had the manual analysis been performed first, the results would have been swapped. The time spent in this case was dominated by the decoding of the payload which was done manually in both manual and LLM-assisted investigations.

Our findings show that AI assistance reliably cuts down triage time for SOC alerts. Even though the impact varies based on the complexity of the case, the AI gives analysts a head start on their investigations and handles the repetitive parts of the workload, saving both time and effort.

B. Qualitative Case Studies

Figure 2 illustrates how the ReAct agent performs end-to-end alert triage for a real security alert in the SOC. We have anonymized all potentially sensitive identifiers, including IP addresses, MAC addresses, device names, and user information to preserve privacy. The security alert from the IDS is passed to the agent, which reported an internal host communicating with an external destination over TLS (port 443), with detection signature linking the traffic to a known command-and-control (C2) domain. The agent began the investigation by retrieving relevant network logs from Splunk (Step 1) by correctly filtering them based on the IPs in the alert. Then the agent performed threat enrichment (Step 2) by querying VirusTotal for reputation information related to the destination domain and IP. After confirming that the domain is malicious, the agent then pivoted to infrastructure context by querying Microsoft Sentinel logs (Step 3) to find the internal IP address associated with the alert, and in Step 4 the agent queried DHCP records to identify the MAC address and hostname of the affected device, then it queried Azure AD (Step 5) to retrieve device ownership information. These transitions demonstrate the agent's ability to reason across heterogeneous data sources during the investigation. Finally, in Step 6, the agent generated a structured report based on the collected evidence. The report accurately summarized the security alert, presented supporting evidence from multiple data sources, and provided clear analysis and recommendations. The agent concluded that the host was likely compromised and recommended standard response actions, including endpoint

containment and blocking of the associated domain or IP. This case study of the agent investigating a real SOC alert demonstrates that LLM-based agents are capable of handling routine analyst tasks like data gathering by navigating across different tools, and summarizing alerts into a structured report.

In another SOC ticket, we observed that the agent failed to correctly analyze the ticket due to over-reliance on alert metadata and IP reputation, but after the analyst provided additional instructions to guide the investigation, the agent corrected its analysis. In this ticket, the alert flagged "Possible SQL injection obfuscated via REVERSE function" for HTTP traffic from an external source to a web server hosted in the organization's network. Initially, the agent considered the signature match and VirusTotal detections as strong evidence of malicious activity and generated a report that classified the alert as malicious, and recommended isolating the host and blocking the external IP. After reviewing the report, the analyst prompted the agent to validate the actual network traffic between the source and destination. The agent re-examined the payload in the network traffic log and recognized that the request was for a static PNG file named with substring "Reverse(1)" appearing only in the filename. There was no indication of SQL injection in query parameters or request body. Taking this extra context into account, and noticing that the traffic matched typical automated crawler behavior, the agent changed its conclusion to a likely false positive and adjusted its recommended actions to reflect that. This example demonstrates how a human-in-the-loop approach can validate the agent's response and, through additional instructions and prompts, help guide the investigation in the right direction.

V. Related Works

Recent works have explored the use of Large Language Models in various cybersecurity tasks, including penetration testing [2], vulnerability detection [9], malware analysis [3], and phishing detection [4]. These works show that LLMs can assist with a number of security tasks, but their role in real-world SOC operations is less explored. Other works have explored machine learning approaches for automating alert triage in SOCs. AACT [10] learns from historical analysts' decisions to predict benign alerts and suppress them, reducing workload by over 60% in a real SOC environment. These approaches, however, demand frequent model training, and lack interactive, human-in-the-loop capabilities for decision making. Freitas et al. [5] presented Copilot Guided Response (CGR), a cloud-based system that provides automated triage and remediation recommendations. This system uses machine learning models trained on Microsoft Defender XDR telemetry data to classify alerts and suggest remediations. However, CGR is a general-purpose system that may not be suitable for individual SOC environments with unique workflows and dynamic toolchains. Singh et al. [11] conducted a large-scale empirical study of LLM use in operational SOCs. They analyzed over 3000 queries submitted by SOC analysts over a 10-month period and found that analysts used LLMs as cognitive aids for explanation, writing scripts and summarization tasks—rather than for decision making—highlighting the value of human-AI collaboration in SOCs.

VI. Limitations

Our evaluation is limited in both scale and participants. We tested the system on only eight tickets. The evaluation was performed by researchers who built the tool rather than independent SOC analysts. Therefore, there is an inherent bias in the reported efficiency gains. Although we alternated the order of manual vs AI-assisted triage when working on the tickets, researcher familiarity with the system likely introduced bias relative to a novice user. These constraints limit the strength of the quantitative claims, and the case studies should be interpreted as qualitative evidence of feasibility rather than conclusive performance improvements.

VII. Conclusion and Future Work

We presented an LLM-driven ReAct agent that can be embedded into real-world SOC workflows to assist analysts in alert triage. We identified the repetitive, tool-fragmented tasks in an operating SOC by immersing ourselves in the SOC workflows, which led us to design the SOC AI Companion that can unify data sources, automate enrichment, and generate structured reports while keeping analysts in the loop for decision making. Our evaluation on real SOC tickets showed that the AI agent significantly reduced triage time and reproduced routine analyst behaviors. The agent also refined its investigation and made corrections when the analyst provided additional instructions and prompting when necessary. Our findings show that agentic AI has real potential to act as a trusted teammate in the SOC—helping analysts make better decisions, easing the burdens of repetitive work, and improving overall efficiency.

We intend to conduct human subject study to examine the AI companion's utility when used by SOC analysts. Future work also includes enabling the agent to learn from past tickets and investigations. This will allow the system to reuse successful reasoning patterns, enrichment strategies, and report structures across similar alerts. Additionally, incorporating learning prompts and iterative feedback provided by the analyst will allow the system to adapt to investigation preferences and organizational practices, and improve its effectiveness and personalization over time.

Acknowledgement

We would like to thank the SOC for embracing our field-workers into their operations and facilitating this research. Without their assistance this research would not have been possible. This work is supported by the National Science Foundation under award no. 2235102, and the Office of Naval Research under award no. N00014-23-1-2538. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these agencies.

APPENDIX

The following is the system prompt used by the AI Companion. Part of the prompt is redacted to maintain anonymity of the SOC.

> *Please analyze a common type of SOC ticket and produce a report. The ticket usually describes a problematic network traffic observed from a client's network to an external IP that is deemed malicious. There could be problematic traffic in the opposite direction as well but the vast majority is from the client's network to a malicious external IP. After analyzing the ticket, please do the following.*
>
> *STEP I: Risk determination*
> *1. Get additional details about the ticket from Zendesk.*
> *2. Verify the external IP is indeed malicious by running it through VirusTotal. Gather the result from VT and this will need to be included in STEP III;*
> *3. Perform contextual analysis — inspect HTTP payloads, verify legitimate sources, avoid overreliance on VirusTotal, and question whether alert signatures truly match observed behavior to identify false positives. The results from steps 1 and 2 above, together with this contextual assessment, will be analyzed to determine if the traffic mentioned by the ticket is indeed malicious. That assessment will need to be included in STEP III.*
>
> *STEP II: Identify the client's internal device and the associated user. NOTE: you need to wait for the human user to provide the information below before proceeding.*
> *1. Query Microsoft Defender to determine the internal device's IP address. This step needs to be carried out by the human user. Query template is below:*
> *Use the following KQL template to query CommonSecurityLog in Microsoft Defender (replace placeholders as instructed):*
>
> *[Template omitted for brevity]*
>
> *- Replace ORGANIZATION_IP with the IP address in Organization subnet. Get the IP from the ticket that starts with* [redacted]
> *- Replace EXTERNAL_IP with the external IP address mentioned in the ticket.*
> *Output the exact query and wait for the human user to provide the result.*
> *2. Display the IP and a time range for the user to conduct query to* [redacted]*. Wait for the user to provide the MAC address of the device, the device name, and the* [redacted] *associated with the device.*
>
> *STEP III: Report generation.*
> *After you have pieced together all the necessary information above, please produce a report that contains the following content:*
> *1. Executive Summary with conclusion/assessment of the maliciousness of the observed traffic.*
> *2. All supporting evidence as described above. Include all raw log data and tool outputs in separate sections for transparency.*
> *3. Recommendations. If the conclusion is malicious, recommend blocking the external malicious IP and domain(if any). Dont provide more than 2 recommendations.*

REFERENCES

[1] B. A. Alahmadi, L. Axon, and I. Martinovic, "99% false positives: A qualitative study of SOC analysts' perspectives on security alarms," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 2783–2800. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/alahmadi

[2] G. Deng, Y. Liu, V. Mayoral-Vilches, P. Liu, Y. Li, Y. Xu, T. Zhang, Y. Liu, M. Pinzger, and S. Rass, "PentestGPT: Evaluating and harnessing large language models for automated penetration testing," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 847–864. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/deng

[3] W. Kasri, Y. Himeur, H. A. Alkhazaleh, S. Tarapiah, S. Atalla, W. Mansoor, and H. Al-Ahmad, "From vulnerability to defense: The role of large language models in enhancing cybersecurity," *Computation*, vol. 13, no. 2, p. 30, 2025.

[4] T. Koide, H. Nakano, and D. Chiba, "Chatphishdetector: Detecting phishing sites using large language models," *IEEE Access*, 2024.

[5] S. Freitas, J. Kalajdjieski, A. Gharib, and R. McCann, "Ai-driven guided response for security operation centers with microsoft copilot for security," in *Companion Proceedings of the ACM on Web Conference 2025*, 2025, pp. 191–200.

[6] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[7] S. C. Sundaramurthy, A. G. Bardas, J. Case, X. Ou, M. Wesch, J. McHugh, and S. R. Rajagopalan, "A human capital model for mitigating security analyst burnout," in *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, 2015, pp. 347–359.

[8] S. Agarwal, L. Ahmad, J. Ai, S. Altman, A. Applebaum, E. Arbus, R. K. Arora, Y. Bai, B. Baker, H. Bao *et al.*, "gpt-oss-120b & gpt-oss-20b model card," *arXiv preprint arXiv:2508.10925*, 2025.

[9] K. Shashwat, F. Hahn, X. Ou, D. Goldgof, L. Hall, J. Ligatti, S. R. Rajgopalan, and A. Z. Tabari, "A preliminary study on using large language models in software pentesting," in *Workshop on SOC Operations and Construction (WOSOC)*, March 2024.

[10] M. Turcotte, F. Labrèche, and S.-O. Paquette, "Automated alert classification and triage (aact): An intelligent system for the prioritisation of cybersecurity alerts," *arXiv preprint arXiv:2505.09843*, 2025.

[11] R. Singh, S. Tariq, F. Jalalvand, M. B. Chhetri, S. Nepal, C. Paris, and M. Lochner, "Llms in the soc: An empirical study of human-ai collaboration in security operations centres," 2025. [Online]. Available: https://arxiv.org/abs/2508.18947