

Evaluation of LLM Agents for the SOC Tier 1 Analyst Triage Process

UNIVERSITY OF TURKU
Department of Computing
Master of Science (Tech) Thesis
Department of Computing, Faculty of Technology
June 2024
Openime Oniagbi

Supervisors:
Antti Hakkala
Ismayil Hasanov

UNIVERSITY OF TURKU
Department of Computing

OPENIME ONIAGBI: Evaluation of LLM Agents for the SOC Tier 1 Analyst Triage Process

Master of Science (Tech) Thesis, 62 p., 3 app. p.
Department of Computing, Faculty of Technology
June 2024

Cyber attacks are continuously growing in volume and complexity, which requires developing efficient and effective solutions that enable Security Operations Center (SOC) environments to detect and mitigate these threats. This thesis focuses on designing and evaluating the application of agents powered by Large Language Models (LLM) in the triage process of Tier 1 SOC analysts. This research evaluates 5 LLMs: OpenAI's GPT-4o and GPT-3.5, Meta Llama 3, Mixtral 8x22B, and OpenHermes 2.5 Mistral 7B. The study investigates LLM agents' ability to label different alerts as either 'interesting' or 'not-interesting.'

The research proposes a framework consisting of an alert generation and processing module, the LLM agent, and a reporting module. An experiment uses an open-source SIEM, Wazuh, with three endpoints connected to test this framework. 'Not-interesting' alerts were generated from normal user operations, with 'interesting' alerts from adversary emulation using CALDERA. The results show that LLM agents can significantly improve the speed of initial triage and reduce the cognitive load on human analysts by automating routine tasks and providing natural language explanations for alerts. However, challenges like occasional hallucinations, integration complexity, and privacy risks are identified. The research concludes that while LLM agents show promise for use in the triage process in Tier 1 SOC operations, a hybrid approach that uses the LLM agents as co-pilots is essential to maximize benefits and mitigate risks.

This research contributes to the growing body of knowledge on artificial intelligence in cybersecurity, providing practical insights for organizations considering integrating LLMs into their SOC workflows. Recommendations for future work include exploring testing and training LLMs on a large dataset of alerts and investigating prompt optimization techniques to get the best results from LLM agents.

Keywords: Cybersecurity, Large Language Model (LLM), Security Operation Center (SOC), Analysis and Triage, Security Information and Event Management (SIEM), ChatGPT, Mistral AI, OpenHermes, Llama 3

Contents

1	Introduction	1
1.1	Problem Statement	4
1.2	Aim and Objectives	5
1.3	Research Questions	6
1.4	Thesis Structure	6
2	The Security Operation Center	7
2.1	Description of the Security Operation Center	7
2.1.1	SOC Architecture	8
2.1.2	SOC Structure	8
2.2	Security Information and Event Management	14
2.2.1	SIEM Types	16
2.3	The SOC Tier 1 Analyst	17
2.3.1	Triage Process	19
2.3.2	Related Work	21
3	Large Language Models in Cybersecurity	24
3.1	Evolution of Large Language Models	25
3.2	Prompt Engineering	26
3.2.1	Zero-shot Learning	27
3.2.2	Chain-of-thought Prompting	27

3.2.3	LLM Public APIs	28
3.3	LLMs and Threat Detection	29
3.3.1	Other Cybersecurity applications of LLMs	29
3.3.2	LLM Limitations	30
3.3.3	LLMs and the SOC Tier 1 Analyst Triage Process	32
4	Framework and Experiment Design	33
4.1	Triage System	33
4.1.1	Alert Generation and Processing Component (SIEM)	33
4.1.2	LLM Agent	34
4.1.3	Reporting Component	34
4.2	Experiment Design	35
4.2.1	Dataset Preparation	35
4.2.2	LLM Agent Alert Triage	37
4.2.3	Evaluation Metrics	43
5	Results and Discussion	46
5.1	Description of Alert Dataset	46
5.1.1	Windows Alerts	46
5.1.2	Linux Alerts	47
5.1.3	macOS Alerts	47
5.2	Numerical Results	50
5.3	Qualitative Analysis	52
5.3.1	Testing Understanding of Alert Context	54
5.4	Cost Analysis	56
5.5	Research Implications	56
6	Conclusion	58
6.1	Limitations of the Study	60

6.2 Future Work	61
References	63
Appendices	
A Sample Wazuh Alert	A-1

List of Figures

2.1	SOC structure and architecture	9
2.2	SOC alert and incident management	11
2.3	SIEM components	14
2.4	Sample Wazuh dashboard	18
2.5	Tier 1 analyst workflow [33].	20
4.1	Framework for automated alert triage	35
4.2	Log collection architecture	36
4.3	Wazuh agent dashboard	36
5.1	Alert classification overall and for each operating system	47

List of Tables

4.1	Alert classification	37
5.1	Windows alerts	48
5.2	Linux alerts	49
5.3	macOS alerts	50
5.4	Performance comparison of LLM agents	52

List of Snippets

3.1	Python code to create an OpenAI assistant math tutor	28
4.1	Create OpenAI Assistant	39
4.2	Upload alert file to OpenAI assistant	39
4.3	Create and send message to OpenAI assistant	41
4.4	Together AI prompt	42
5.1	Output from LLM agents	52
5.2	LLM agent referencing alert severity	53
5.3	GPT-4o context analysis	55
A.1	Sample Wazuh alert	A-1

1 Introduction

Cybersecurity has become critical in today's modern digital world due to the increasing dependence on interconnected technologies. The widespread use of electronic devices, cloud services, and digitalization of assets has introduced easier ways to communicate and conduct business. Technology is more important than ever before, with several critical infrastructures and sectors depending on it in one way or another. However, the pervasiveness of technology worldwide also introduces different security challenges and threats, with cyber-attacks continuously evolving with new techniques and targets. Cyber attacks can take various forms but ultimately cause financial and reputational loss for individuals and corporations when they are successful. Cyber attacks can also lead to loss of lives when they are successful against critical infrastructure like healthcare.

To combat cyber threats, nations and organizations continue to develop and use various cyber security tools, processes, and strategies to prevent cyber incidents and reduce their impact when they occur. One of such developments is the Security Operations Center (SOC). For organizations, the SOC is the central hub of all alerts generated by different security tools. The primary purpose of the SOC is to detect, respond to, and report cyber-attacks promptly in monitored environments [1]. Also, Mutemwa et al. highlight that SOCs are critical to the cyber security strategy of organizations [2]. Therefore, any attempt to optimize SOC processes will usually lead to an improvement in the cyber security posture of an organization.

There are several tools used in a SOC to function properly and achieve its goals of protecting the organization, and one such tool is the Security Information and Event Management (SIEM) system [2]. The SIEM is an integral SOC tool that collects, correlates, and analyzes events and alerts from security logs generated by network devices, servers, applications, and users [3]. The SIEM also connects to cyber threat intelligence (CTI) platforms, which provide context to the generated alerts [4]. Research on SIEM technology usually focuses on enhancing the capabilities of SIEM solutions to detect threats more effectively from the logs it receives [5, 6].

However, the incident detection process in the SOC includes a human component. This is because human analysts are used in the SOC to review and triage the alerts generated by the SIEM system to determine the presence of threats and incidents within the organization. A conventional SOC uses a system of analysts that operate in a tiered system [7]. From the ground up, the Tier 1 analysts or SOC operators are the first to review and triage alerts. Only the relevant alerts are forwarded to the higher tier analysts, Tier 2 and Tier 3, who are more experienced in conducting in-depth investigations and remediation of identified incidents.

SIEM solutions usually generate many alerts using redefined rules with a high frequency of false positives because they collect security logs from several devices and applications. False positives are alerts that wrongly indicate the occurrence of an incident in the monitored environment. Besides false positives, most SIEM rules and the alerts they generate are usually informational and require no further steps in the incident detection process. This leads to a situation where Tier 1 analysts must review a lot of alerts within a limited time. This amount of workload leads to the widely known issue of alert fatigue. Alert fatigue is a state of desensitization experienced by SOC operators after continuous exposure to SIEM alerts [8]. Alert fatigue can lead to incidents not being identified promptly or, worse, not being identified.

A contributing factor to alert fatigue is the high frequency of false positives in SIEM alerts, and Ban et al. state that 31.9% of security analysts reported that they do not pay attention to alerts anymore due to alert fatigue [8]. Another contributing factor to alert fatigue is the monotonous work performed by SOC operators [9].

Several systems have been designed to address the alert fatigue issue by optimizing or automating the Tier 1 alert triage process. For example, Kersten et al. provide a process that improves the Tier 1 analyst's decision-making quality. The method outlines the sequence of tasks required for a Tier 1 analyst to make a correct decision regarding determining the importance of an SIEM alert [7]. In their study, Kersten et al. showed that analysts who used this outlined process made better decisions than analysts who relied on experience and knowledge [7]. Meanwhile, Ban et al. focused on automating the Tier 1 analyst triage process entirely using Artificial Intelligence (AI) [8]. The results showed a low false positive rate and a high rate of alerts correctly identified as potential incidents. This study also demonstrates the potential of AI in cyber security operations.

Artificial Intelligence has grown in popularity recently due to the developments and general availability of Large Language Models (LLMs) in intelligent chatbots. LLMs are neural networks that use data in text format to process language-related tasks. The latest breakthrough in LLMs comes from using Transformer-based models in Natural Language Processing (NLP). This has resulted in developing effective LLM models like GPT (Generative pre-trained Transformer), PaLM, and Llama [10]. These models serve as the foundation for intelligent chatbots; for example, ChatGPT uses various versions of GPT, including the latest GPT-4, which is multimodal and can accept image and text inputs [11].

Although LLM-based chatbots are most widely used to generate content (images, text, and videos), they can also perform powerful functions such as multilingual machine translation, code writing, and debugging [11]. Compared to other disciplines,

the study and use of LLMs in cyber security are underexplored [3]. For example, in education, LLMs have been shown to help generate questions for educators and provide assessment and feedback to students [10]. LLMs have also been used to automate and deliver services in customer care, marketing, and healthcare [10]. In cyber security, Motlagh et al. identifies various uses of LLMs in defensive and offensive security [10]. Specifically, in the detection phase of defensive security, LLMs can be used to detect anomalies in logs and vulnerabilities in software. Therefore, this research explores the effectiveness of applying LLMs to the alert triage of Tier 1 analysts in a SOC to automate the process, reducing human workload and alert fatigue.

1.1 Problem Statement

The recent growth in the popularity of AI, particularly LLMs, has resulted in several research opportunities in different fields, even though it has not directly resulted in widespread adoption. Consequently, there has been an increase in the number of papers related to LLMs. However, Motlagh et al. notes that cybersecurity is traditionally slow to adopt data-driven and machine-learning solutions [10]. Nevertheless, cybersecurity research has continued to show the benefits and threats that AI presents in cybersecurity. For example, Zeadally et al. thoroughly describes AI's opportunities in cyber security, particularly in the domain of incident detection [12]. However, a study by Gartner Inc. in 2023 showed that only a third of the respondents were using or implementing AI application security tools, and slightly over half considered it [13]. Compared to the potential shown by AI in security research, AI-backed solutions are not heavily adopted in the industry.

One barrier to adopting AI solutions is usually job security, as noted by a systematic literature review on the adoption of AI [14]. However, when the available workforce cannot meet the industry's demands, AI augmentation should be accept-

able even to employees who feel that their job security is threatened. It is widely accepted that there is a global skill shortage in cybersecurity. Although the estimates vary, Furnell reports that there are between 1.8 million and 4 million unfilled cybersecurity positions worldwide [15]. This skill shortage also affects entry-level positions, such as Tier 1 analysts in the SOC. This issue is further exacerbated by the fact that organizations find filling positions with staff with the required skills challenging.

Alert fatigue is thus a product of the cybersecurity skill shortage when the available staff must analyze more events than they can handle [15]. Sundaramurthy et al. also notes that a high turnover of analysts plagues SOCs due to burnout [16]. Moreover, burnout can result in analysts making poor decisions, which can lead to huge financial losses in a critical field like cybersecurity. Since SOC analysts usually perform the same tasks repeatedly, it is easy to see how automating some of these tasks can mitigate burnout and improve the creativity of analysts in the SOC [16]. Therefore, this study will focus on designing and determining the efficiency of a system that utilizes openly available LLMs to automate the Tier 1 analyst alert triage process. Such a system provides much-needed support for analysts in the SOC and will also provide an idea of how useful LLMs can be in cybersecurity.

1.2 Aim and Objectives

This study aims to design and test the effectiveness of an automated Tier 1 analyst alert triage process performed by an LLM system.

The objective is to determine the reliability of using an LLM-backed system in a SOC for Tier 1 analyst alert triage processes. If such a system is proven to be reliable, it will help reduce alert fatigue and improve operational efficiency in SOCs, especially those employed by SMEs.

1.3 Research Questions

This thesis aims to answer the following research questions:

- RQ1: How can an LLM agent perform the SOC Tier 1 analyst's alert triage process?
- RQ2: What is the accuracy of LLMs when they are used to automate the Tier 1 analyst's alert triage process?

1.4 Thesis Structure

The rest of the thesis consists of 6 chapters, and a brief description of the chapters is provided below.

Chapters 2 and 3 focus on the relevant theory and helpful terminology to the rest of the thesis. Chapter 2 specifically describes SOCs, the team structure, incident detection processes, SOC tools, and the SIEM. Chapter 3 describes and analyses LLMs, how they work, and the application of LLMs in cybersecurity and the SOC. Chapter 4 introduces the different components of a system that integrates LLMs into the SOC Tier 1 analyst workflow and the experiment design. Chapter 5 is a discussion of the results of the research. Chapter 6 summarizes the findings for the research questions of the studies and provides recommendations.

2 The Security Operation Center

This chapter will address the fundamental theory relevant to answering the research questions posed in the previous chapter. This study discusses the relevant literature concerning automating the workflow for Tier 1 security analysts in a SOC, SOC operations and processes, and SIEM alerts analysis and triage. This chapter will describe a Security Operations Center, its structure, and the tools used in conventional SOC environments. The chapter also explains the SOC analyst workflow, emphasizing Tier 1 analysts in the SOC.

2.1 Description of the Security Operation Center

A SOC is a centralized hub that operates at the heart of an organization's network and security architecture and monitors an organization's assets to detect, analyze, mitigate, and report security incidents [2, 4]. A unit within the organization can provide these SOC services, or a third-party managed security solution provider who monitors the organization for a fee [7]. A SOC can also operate in a hybrid model that incorporates a dedicated internal SOC with the services of an outsourced service provider [4]. Regardless of the mode of implementation, a SOC is usually heavily integrated into the organization's information technology infrastructure [2] and serves as the centralized hub where all security events are directed to be analyzed. One of the key features of any successful SOC is continuous monitoring [17], which enables the SOC to detect issues in near real-time and carry out mitigating

actions to prevent damage from cyber-attacks. Mughal asserts that the SOC is essential to any organization's security strategy [17]. Furthermore, Mutemwa et al. state that the SOC is critical in measuring the maturity of an organization's security posture [2].

2.1.1 SOC Architecture

A SOC's overall architecture can be centralized, distributed, or decentralized [4]. A SOC with a centralized architecture requires that all the logs sent from different devices and locations are processed in one central SOC. Log processing and analysis are spread across several subsidiaries in a distributed SOC. The decentralized SOC combines centralized and distributed SOC design into an architecture comprising SOC subsidiaries reporting to a central SOC to eliminate any single point of failure.

Technological Architecture

The building blocks of any SOC require several technologies that perform specific functions. On a basic level, conventional SOC environments have technologies that perform the functions of event generation, event collection, event databases, analysis engines, case management, and response management [4]. Recently, this architecture has been improved to include cyber threat intelligence to provide additional context to internal events and incidents [18, 19]. As seen in Figure 2.1 [4, 20], the various technologies in the SOC are integrated with the SIEM, which provides a holistic view of the environment that the SOC is monitoring.

2.1.2 SOC Structure

To discuss the structure of a SOC, Vielberth et al. propose the PPTGC framework, which is an abbreviation of People, Processes, Technology, and Governance and Compliance [4]. This framework provides a lens through which we can understand

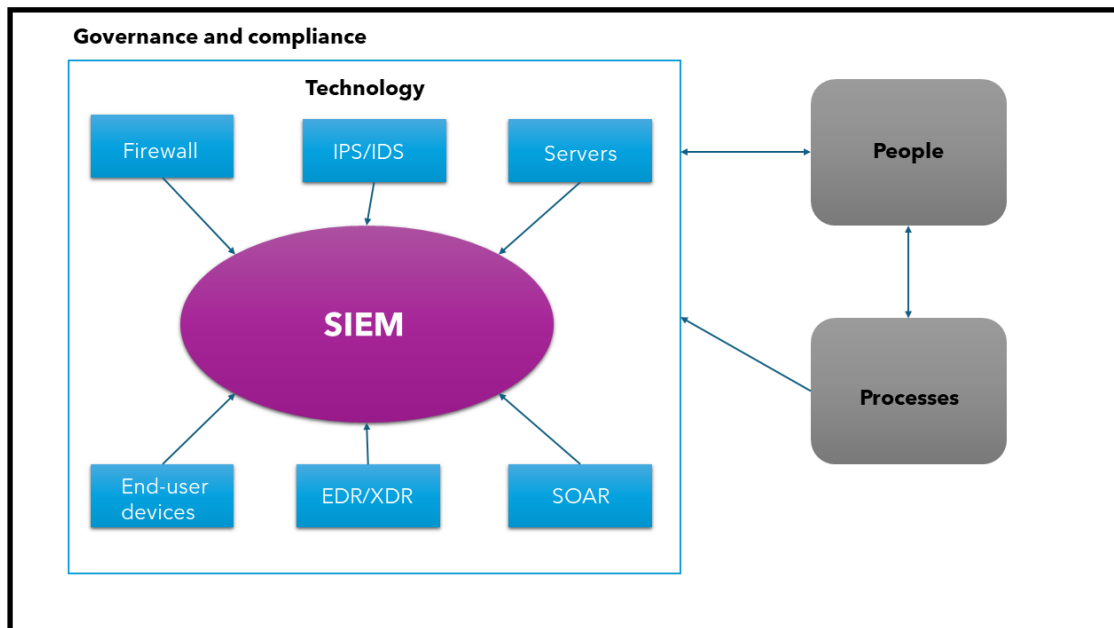


Figure 2.1: SOC structure and architecture

the various elements of the SOC. It is an improvement of the PPT (People, Process, and Technology) framework that is usually used to describe information technology systems.

People

People have several roles and responsibilities within a SOC. Although there are different classification systems, it is generally recognized that a SOC should have Tier 1 analysts who handle triage, Tier 2 and Tier 3 analysts who are incident responders and threat hunters, and a SOC manager [4, 7]. Apart from this tiered system, Mughal describes a SOC that uses a responsibility-based classification system that focuses on the roles carried out by members of the SOC team. In this system, the additional roles include threat intelligence analysts, vulnerability management specialists, penetration testers, forensic analysts, and user and entity behavior analysts [17].

Process

The process followed by the SOC for handling alerts is usually the same as conventional incident response guidelines like the NIST Computer Security Incident Handling Guide [21]. This is because the core functions of the SOC are to prevent, detect, and respond to cyber security incidents. In the guide, NIST recommends four steps, which are “preparation,” “detection and analysis,” “containment, eradication, and recovery,” and “post-incident activity” [21]. Figure 2.2 shows how each of these steps relate to each other to create a cyclical process where each incident leads to the improvement of the incident response process.

In a SOC, preparation activities include designing and implementing systems that collect, process, and log event data. The processing of events includes normalization, filtering, reduction, log rotation, synchronizing time across log sources, aggregation, and correlation [4]. Correlation especially allows the SOC to track related activities from different log sources.

The detection and analysis phase in the SOC comprises the generation of alerts that indicate suspicious activities using different tools and techniques and the human analysts’ triage and prioritization of these alerts by the SOC [4]. The success of these phases depends on the quality of the detection mechanisms implemented and the skill of the analysts involved [7].

The containment phase depends on the SOC’s operational model. In an outsourced SOC, it is possible that the SOC is not responsible for any of the processes after detecting, analyzing, and reporting the incident to the organization. This phase can include using Security Orchestration, Automation, and Response (SOAR) tools to automate response actions to identified incidents. This phase aims to ensure that the incident does not spread across the organization’s assets and that affected assets are recovered successfully [21]. These assets can include endpoints, servers, network devices, and user accounts.

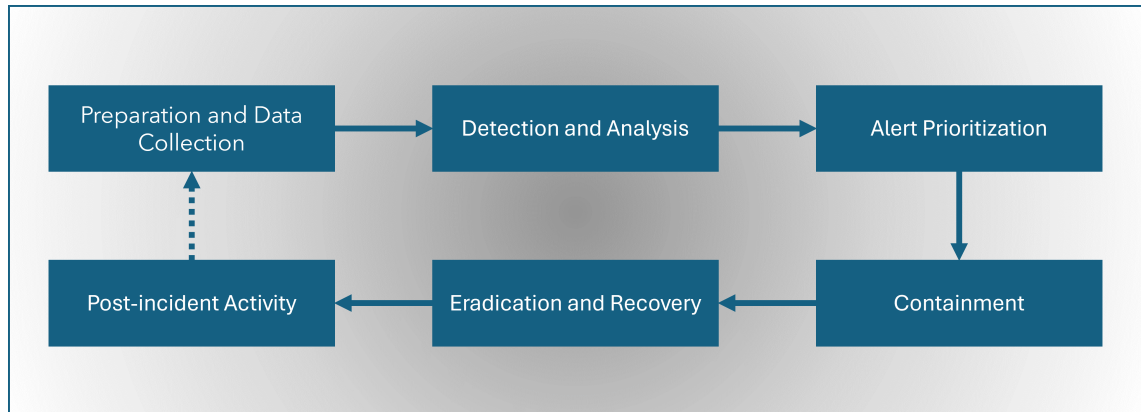


Figure 2.2: SOC alert and incident management

Post-incident activity in the SOC involves enhancing its tools and detection capabilities using the information gathered during the forensic analysis of an incident. The activities performed in this phase ensure that future attacks using similar techniques are detected and mitigated early before any damage is done.

Technology

The technology used in a SOC has been categorized into data collection tools, analysis and detection tools, and visualization tools [4]. A SOC is usually filled with several screens with dashboards showing changes in the state of devices in the organization's network. These dashboards help analysts to identify anomalies and respond appropriately.

Data collection in the SOC requires collecting data from several sources. The data sources include security software, like firewalls, anti-malware software, and vulnerability scanners. Other sources include network assets, virtualization environments, operational technology, physical security assets like cameras, threat intelligence, employees, and external users. Additionally, logs are collected from operating systems and applications and provide information about activities on those assets [4]. Usually, all the events from these different data sources are forwarded to

a SIEM solution for further analysis [2]. SIEM solutions will be discussed in depth in the second part of this chapter. However, it should be noted that the SIEM is a part of each block in the SOC technology stack described above. SIEM solutions usually have a threat detection engine and dashboards for investigating and visualizing security data. Other technologies in the SOC include log storage systems, endpoint detection and response (EDR) solutions, and SOAR solutions. EDR and SOAR solutions can automatically respond to detected incidents, which can help reduce the impact and effectiveness of cyber-attacks [17].

Although technically not a technology, Vielberth et al. note that a conventional SOC uses playbooks to automate frequent use cases [4]. These playbooks can also be implemented in the SOAR platform.

Governance and Compliance

Governance and compliance describe activities that can be a service provided to the SOC or by the SOC. They can be used to measure the performance of the SOC against industry standards and regulations. Although there are no recognized holistic SOC standards [4], it is vital for every SOC to have metrics and key performance indicators to measure its performance [17]. The purpose of measuring SOC performance is to identify subpar performance and implement policies and procedures to improve the SOC. As mentioned above, the SOC can also ensure that the organization it monitors meets specific compliance regulations by conducting regular audits, documenting its findings, and providing recommendations. However, the rest of this section will discuss some metrics that can be used to measure SOC performance.

One of the metrics used to measure SOC performance is coverage [4]. This measures the number of monitored assets about an organization's total number of assets. This is important to measure because a SOC can only monitor a limited number of assets due to resource constraints.

Another metric used to measure SOC performance is false positive rate [4]. This metric measures the percentage of alerts generated by the SOC that are not incidents. Ideally, the SOC should have a very low rate of false positives [22]. However, as mentioned in the previous chapter, that is hardly the case.

To measure the performance of analysts in the SOC, the average analysis time is used to measure how much time analysts spend on alerts on average [4].

Mean Time to Detect (MTTD) is one of the key SOC metrics [17]. This metric measures the time the SOC takes to detect a security incident. An efficient SOC team will have a low MTTD value, indicating that its analysts quickly review alerts and identify incidents.

Another key SOC metric is Mean Time to Respond (MTTR) [17]. This is the time the SOC takes to respond to a security incident. This metric is only relevant to SOC teams responsible for incident response activities.

Compliance metrics [4] are used to measure the SOC team's adherence to regulatory and industry standards. Examples of such standards include ISO/IEC 27001, a general cyber security standard that describes requirements for information security management systems. A SOC can measure its performance against such a standard by measuring the number of policy violations in the SOC environment [22].

Assessing SOC Performance with Adversary Emulation

The metrics described in the previous section are useful for evaluating the performance of a SOC against real alerts and incidents. However, adversary emulation is another technique for evaluating a SOC and identifying gaps in its coverage and technologies. Adversary emulation uses the tactics, techniques, and procedures (TTPs) known threat actors use to test an organization's security implementation [23]. The TTPs of the threat actors is firstly gathered from threat intelligence and mapped to a threat model like the ATT&CK framework created by The MITRE Corporation

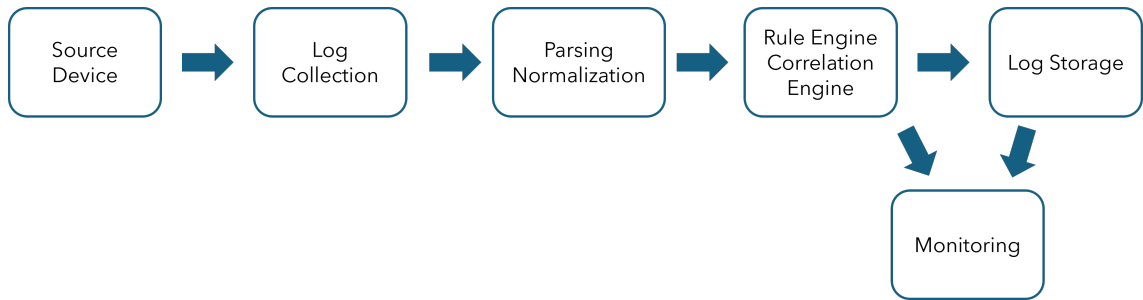


Figure 2.3: SIEM components

[24]. This is beneficial to a SOC because it identifies gaps in the coverage of the SOC and can also be used to measure the performance of the SOC analysts and the SIEM rule engine. For example, SAIBERSOC is an adversary emulation tool that can measure SOC performance and identify areas in the SOC that can be improved [25].

2.2 Security Information and Event Management

As mentioned above, the SIEM is one of the tools used in a SOC. Furthermore, SIEM systems have been described as the central or core technology in a SOC [26]. Bhatt et al. note that SIEM solutions are critical in SOC processes [27]. The functions performed using the SIEM in the SOC are widely known and generally the same across several research papers. These functions include collecting logs from several sources, normalizing the logs from these disparate sources into a standard format, aggregating, correlating, and storing these logs before using the built-in and custom detection mechanisms to detect and report anomalies, and visualizing the same anomalies using dashboards and other reporting mechanisms [26, 28]. The functions of a SIEM solution also directly correlate to its essential components, as shown in Figure 2.3 [26]. It is important to understand the SIEM because it is an important component of the Tier 1 analyst workflow.

Log Collection

SIEM systems were developed as a response to the growth in the number of different security tools used by organizations and the need to collect data from these tools into one platform [27]. Therefore, log collection is the first function of a SIEM, and it is usually performed by a component called a log collector. Although the component's name differs depending on the SIEM solution, the function is usually the same; it provides a mechanism by which the log source can send logs. Logs can be sent using protocols like Syslog or by installing an agent on the log source, which will collect and send the logs to a central SIEM server [28]. The log sources for a SIEM can include firewalls, intrusion detection and prevention systems, anti-malware systems, application systems, network devices, and operating systems.

Normalization

Normalization is necessary because the SIEM collects logs from different sources whose logs come in different formats. The SIEM converts the logs into a standard format, which aids visualization and rule building. This process is also referred to as log parsing.

Realtime Analysis and Correlation

In this phase, the SIEM uses different mechanisms to analyze the events, detect anomalies, and create alerts to notify the analysts. One of the most common mechanisms is using a rule engine containing built-in and custom rules and correlating information from different log sources to create alerts [28]. SIEM rules are created by analyzing attack patterns and including the patterns in conditional statements that are triggered when the patterns are discovered in the events from log sources. SIEM rules can also use contextual information and cyber threat intelligence to create and prioritize alerts [27]. However, Vielberth and Pernul highlight that the

issues with using rules for threat identification are poor detection rates and many false positives [28]. In describing the weaknesses of rules, Cinque et al. state that the problems lie in the fact that these rules rely on historical attack patterns, which do not always work for novel attack techniques [29].

Log Storage

The SIEM is also responsible for storing logs from the different sources. Log storage can be configured for either long- or short-term and usually depends on compliance requirements. However, log storage also serves a forensic purpose [27], especially when incidents are confirmed. The logs stored by the SIEM can be analyzed to discover activities by threat actors that the SIEM did not identify in real-time [27].

2.2.1 SIEM Types

Two types of SIEM solutions are available to organizations. There are commercial solutions and open source/free SIEM solutions [30]. The commercial solutions provide enterprise-grade coverage and are known to be more sophisticated than their open-source counterparts [30]. On the other hand, the free and open-source alternatives are ideal for SMEs, which usually have budgetary constraints for network security.

Examples of commercial SIEM solutions include Microsoft Sentinel, Splunk Enterprise Security (ES), IBM QRadar, LogRhythm's SIEM, AlienVault USM, McAfee Enterprise Security Manager (ESM), Micro Focus ArcSight ESM, Exabeam SMP, Securonix SIEM and Rapid7 InsightIDR [30]. Commercial SIEM solutions usually come bundled with additional features such as User Entity and Behavior Analytics (UEBA), SOAR, EDR, and cloud monitoring.

On the other hand, some of the popular free or open-source SIEM solutions include Wazuh, Elastic SIEM, OSSIM, and Splunk Free [30]. Open-source solutions

like Wazuh offer a large community of users and developers, which can be helpful to SMEs and startups looking to implement SIEM solutions.

Wazuh

Wazuh is an open-source security platform with SIEM and XDR capabilities. As a SIEM, Wazuh can collect logs from various sources, including servers, workstations, and network devices. Wazuh facilitates its log collection using an agent-server architecture. The Wazuh agent installed on the systems collects events and forwards those events to the Wazuh server for analysis and correlation. Additionally, the agents can take action on monitored endpoints, such as removing malicious files or updating firewall rules to block threats. Wazuh creates alerts based on an extensive set of rules to detect threats. The rules contain patterns and conditions that trigger alerts when met. Each alert also has a specific severity level, which indicates its importance. Wazuh comes with built-in rules, but it also allows users to create custom rules and modify the predefined rules. In their study which explored the use of an LLM agent to streamline tasks in the SOC, Kaheh et al. use Wazuh and highlight its adaptability and open-source nature [3]. In a comparative analysis of open-source SIEM solutions, Wazuh had the highest scores for solutions still available as open-source [30]. Figure 2.4 shows a sample Wazuh dashboard with alerts and security events from different endpoints [31].

2.3 The SOC Tier 1 Analyst

The SOC Tier 1 analyst is usually the least experienced [7] and is responsible for the initial investigation of alerts that the SIEM and other security products generate [32]. Nevertheless, the SOC Tier 1 analyst role is fundamental to the success of a SOC. This is because they are the first to assess and determine the criticality of a security event [7]. In other words, if a Tier 1 analyst does not correctly classify a

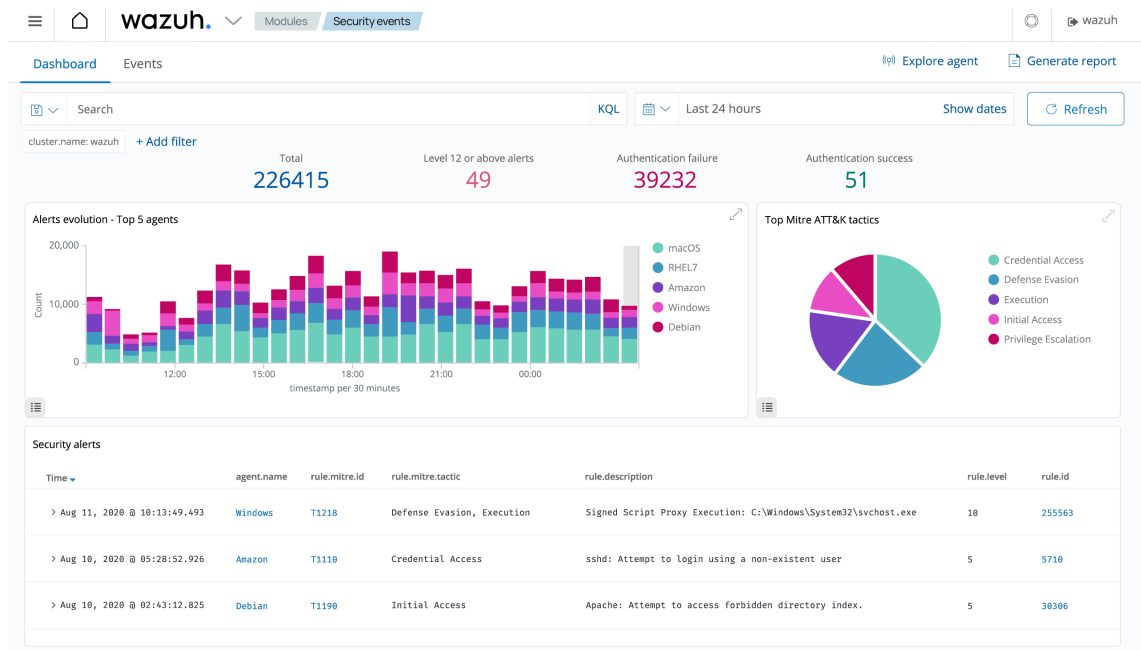


Figure 2.4: Sample Wazuh dashboard

security event, it could result in unidentified incidents.

The responsibility of the Tier 1 analyst is referred to as alert triage in several literature [4, 7, 27], and the process description is mostly the same. The process starts with the SIEM generating alerts based on the detection rules and mechanisms used in the SOC. The Tier 1 analyst must review each alert and determine whether it is interesting. An interesting alert is not a false positive and indicates a likely security incident. The Tier 1 analyst is required to escalate such alerts to the Tier 2 analyst along with related alerts. The Tier 2 analysts investigate further to confirm the incident and start remediation actions for confirmed incidents.

The SOC Tier 1 analysts, just like the SOC, have metrics that can be used to evaluate the performance of each Tier 1 analyst. Vielberth et al. identify people metrics as one of the SOC metrics categories, including the number of incidents closed in a shift and average workload [4]. Furthermore, metrics like average analysis time [4], false positive rate [4], and incident response time [17] also apply to each

SOC analyst as much as they apply to the SOC in general. In the case of Tier 1 analysts, the false positive rate refers to the percentage of false incidents identified by the analyst. In a survey, Agyepong et al. identified the number of incidents raised, time taken to detect incidents, and quality of analysis as the top metrics mentioned by study participants [32].

2.3.1 Triage Process

The Tier 1 analyst process is centered around alert triage and escalation. However, the alert triage process can include several sub-steps. Kersten et al. proposed a model of the Tier 1 analyst's triage process that provides a good overview of the required steps in the triage process [7]. This model is shown in Figure 2.5 and involves a four-stage process that enables analysts to correctly identify alerts as 'interesting' or not [7].

The first stage of the process involves the analyst reviewing what Kersten et al. refer to as 'Relevance indicators' [7]. The relevance indicators include reviewing the specificity and age of the rule that triggered the alert and whether it applies to the scope of the monitored environment. These considerations enable the analysts to decide whether an alert is relevant to the monitored environment. For example, an organization can have a guest network where low-priority alerts should be ignored.

The second stage of the process involves analyzing the alert history and surrounding alerts to decide whether it is a known 'non-interesting' alert. These two alert properties, their history, and surrounding alerts can provide additional context to alerts in review. For example, if a penetration test is ongoing and the SOC has been notified of the source, then all alerts from that source can be observed but do not need to be escalated as incidents.

The third stage of the process involves reviewing related logs (including those that did not generate alerts) and analyzing traffic and the target host information

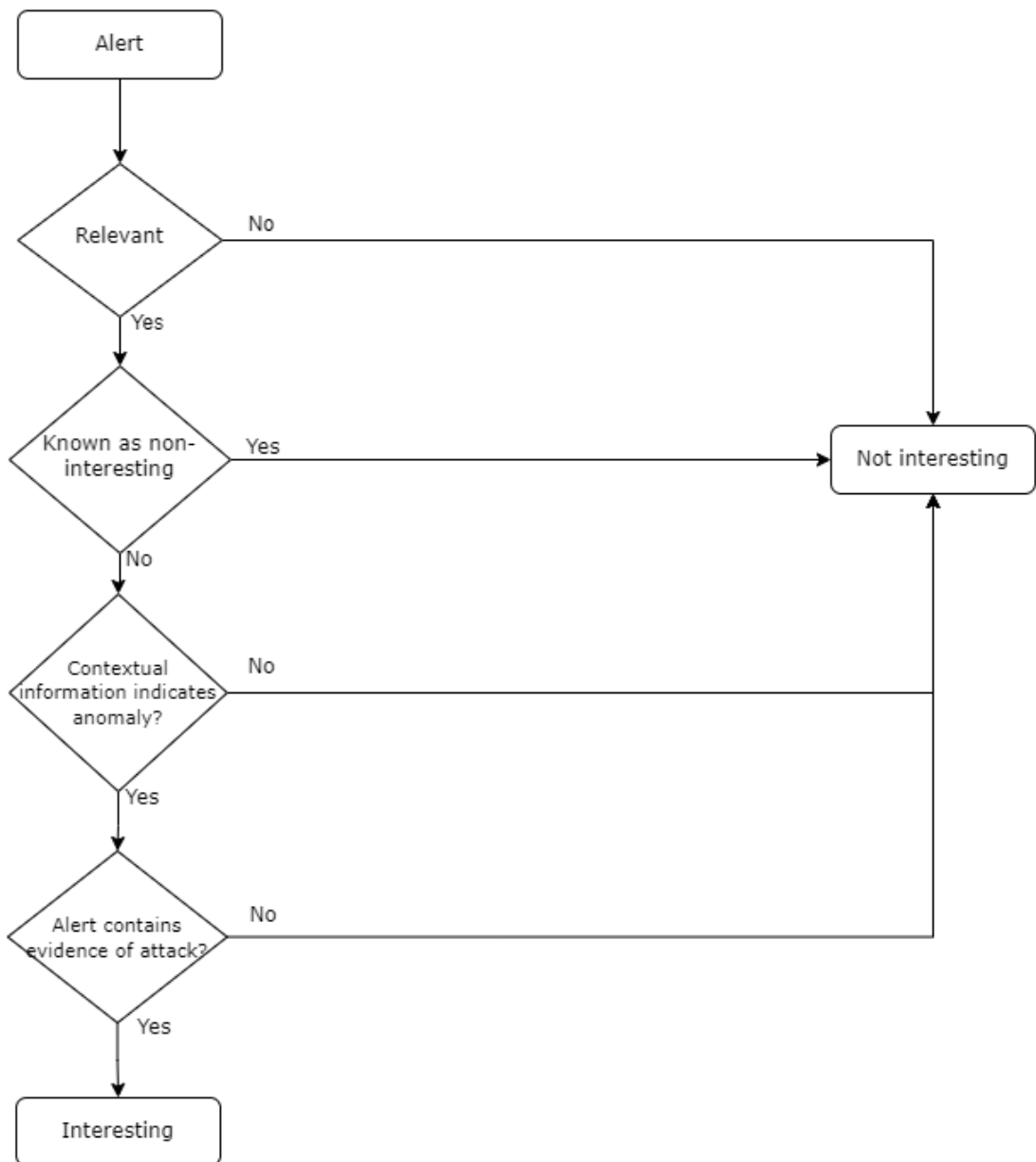


Figure 2.5: Tier 1 analyst workflow [33].

to find evidence of a potential attack. This phase is necessary when no surrounding context exists around an alert to suggest it is 'not interesting.' Therefore, in this stage, the analyst analyses every possible source of information to find evidence of an attack.

In the final stage, the analysts review the attack to find information about the attacker, indicators of compromise, and related use cases for the attack. The investigation in this stage provides the Tier 1 analyst with the necessary information to decide whether the alert was successful and escalate the alert to the Tier 2 analysts, if necessary. The related use cases for the attack can also inform the Tier 1 analyst of the correct way to report the alert.

This Tier 1 analyst triage process model shows that deciding on an alert's authenticity must involve reviewing the alert itself, along with all contextual information about it, surrounding alerts, and the monitored environment. In addition, understanding attack types and extracting indicators of compromise are critical to alert escalation.

2.3.2 Related Work

Due to the monotonous nature of the Tier 1 analyst role [7], some security operation centers usually develop and use playbooks or standard operating procedures for the Tier 1 analysts [4]. These playbooks contain recommendations that Tier 1 analysts follow to perform their roles in the SOC. Regardless of this measure, Kersten et al. highlight that Tier 1 analysts are still known to provide error-prone analysis of security events [7]. Apart from this, the repetitive nature of the process also quickly results in analyst burnout and huge turnover. This turnover results in experienced Tier 1 analysts being replaced by inexperienced substitutes, continuing the vicious cycle. This thesis aims to resolve the issue by developing and testing a process that uses a Large Language Model to perform the duties of the SOC Tier 1 analyst.

However, this section describes other attempts at optimizing the SOC Tier 1 analyst process and improving its performance.

Although there has been in-depth research into solutions that support or automate the Tier 1 analyst process, there is little research into using Large Language Models (LLMs) to automate the process. Filar et al. developed a conversational agent that supports analysts during alert triage, which increased analyst efficiency and improved focus [34]. However, this proposed system only supports the Tier 1 analyst process without automating it. Perera et al. integrated artificial intelligence into a conversational SOC chatbot [9], similar to [34]. Kaheh et al. proposed a system that uses LLMs in a SOC conversational agent to explain cyber threats to the analysts. Kaheh et al. reports that the agents were also capable of taking proactive actions like a SOAR tool to remediate identified incident [3].

Other attempts at automating or improving the Tier 1 analyst process focus on developing models or frameworks. Danquah [20] proposed a framework to automate the entire lifecycle of a SOC alert from triage to escalation and containment. The proposed framework uses artificial intelligence to develop the SOC baseline intelligence and respond to identified vulnerabilities. Although the study does not test the framework, it mentions the viability of an artificial intelligence model in automating certain SOC processes. Mulders [33] proposes and tests a model that automates the triage process in a SOC to identify non-interesting events. To verify the model, SAIBERSOC, a tool which uses adversary emulation to test the performance of a SOC and its analyst [25], is employed. SAIBERSOC is a tool based on the MITRE ATT&CK framework, and it can be used to identify areas of improvement in the SOC. Apart from showing the value of using MITRE ATT&CK based testing methodologies, the model developed by [33] accurately identified non-interesting events. The model focused on breaking down the Tier 1 analyst process into steps rather than implementing a single tool or machine learning model to au-

tomate the entire process. The model focuses on using the model to improve the Tier 1 analyst process rather than automating it entirely. In their tests, Kersten et al. showed that Tier 1 analysts who followed the proposed model produced better results than those who did not [7].

Finally, Ban et al. proposed an alert triage system that used machine learning models to triage and visualize alerts [8]. This system is the closest implementation to this thesis and reported a recall rate of 99.598% for highly critical alerts and a false positive rate of 0.001%. Ban et al. collected network logs into a security appliance and tested them against several machine learning algorithms [8]. The study showed the viability of using artificial intelligence in the Tier 1 analyst triage process.

3 Large Language Models in Cybersecurity

Artificial intelligence (AI) is a branch of computer science that explores the theories and techniques necessary for the simulation, extensions, and expansion of human intelligence [35]. AI technology is usually comprised of systems capable of analyzing their environment and autonomously responding to achieve specific goals [36]. Language modeling is one of the ways to enhance machine intelligence [37], and a Large Language Models (LLM) is a type of AI that “possess deep language comprehension, human-like text generation capabilities, contextual awareness, and robust problem-solving skills” [38]. This combination of features makes them useful in performing various tasks in different domains. For example, Motlagh et al. mention that LLMs can support teachers in education, develop music captions and video game scripts in the entertainment industry, automate customer service and supply chain management, and support medical professionals in the healthcare sector [10]. The use of LLMs in cybersecurity is also growing both for offensive and defensive purposes. The rest of this chapter is dedicated to highlighting literature concerning core LLM concepts and the use of LLMs in cybersecurity, specifically for threat detection. This chapter will also present related work on using LLMs to automate cybersecurity processes.

3.1 Evolution of Large Language Models

The recent growth in the development of Large Language Models (LLMs) can be credited to the improvements in deep learning research, the increase in the availability and relative affordability of massive computational resources, and the huge amounts of available training data [39]. This section briefly describes the significant milestones and core technology behind Large Language Models. Large language models are a type of language model. In general, language models aim to determine the probability of words occurring in a sequence [37]. Hadi et al. note that the history of language models can be traced to the 1950s and 1960s when the first language models were developed [39]. The models were rule-based and relied on manual linguistic rules and features to process language. Researchers agree that there were four stages in the development of LLMs with the first stage being the development of Statistical Language Models (SLM) [37, 39]. SLMs estimated the likelihood of the next word using the frequency of occurrence of the previous words. These models were developed in the 1990s but suffered from the curse of dimensionality, which made it difficult to accurately estimate higher-order language models [37]. However, SLMs were still used to improve the performance of information retrieval tasks [37].

The second stage in the development of LLMs was the introduction of neural network-based language models (NLM). This technique used neural networks to estimate the probability of the next words in a sequence [39]. The third stage was the development of Pre-trained Language Models (PLMs), which attempted to capture the meaning and context of words in a piece of text [39]. PLMs used pre-training tasks that incorporated unlabeled data to develop context-aware word representations [37]. Finally, the fourth stage occurred when researchers discovered that PLMs could be scaled to improve performance. This gave birth to LLMs which are fundamentally PLMs whose model or data size has been scaled up exponentially [37]. Furthermore, researchers observed that scaling these models resulted in “emergent

abilities” [37], useful to language translation and solving tasks [39].

The ability of LLMs to comprehend patterns and linguistic nuances in the training data is responsible for its current success. Using this context awareness and observed patterns, Yao et al. highlight the ability of LLMs to create coherent and contextually relevant responses [38]. Examples of LLMs include OpenAI’s ChatGPT [40], Meta AI and Microsoft’s Llama [41], and Google’s PaLM [42]. These models have been used in chatbots to create conversational experiences for users [39]. These models can either be proprietary or open-source models. For example, OpenAI’s GPT-3.5 and GPT-4 models are proprietary while models like Llama, Mistral, and OpenHermes are open source. Open source LLMs can be trained, tuned, and deployed locally while proprietary models require a subscription to use. Yao et al. state that LLMs should have four core capabilities; excellent comprehension of natural language context, the ability to generate human-like text, the ability to display awareness of knowledge-intensive domains, and the ability to follow instructions [38]. The last capability enables LLMs to perform new tasks out of the box without new training data. This feature is known as in-context learning and it allows LLMs to be controlled using natural language task specifications or in short, prompts [43].

3.2 Prompt Engineering

As mentioned in the previous section, prompts are used to instruct LLMs on how to perform new tasks that the LLM may have not been originally trained for. Arora et al. define prompts as instructions that describe the inputs and outputs for a task [43]. Chang et al. highlight the usefulness of using prompts to guide LLMs in providing accurate feedback and completing tasks [44]. In the survey Chang et al. also highlights that prompts are one of the ways researchers evaluate LLM performance. Furthermore, Hadi et al. assert that LLM performance can be affected by the specificity and clarity of the words used in a prompt [39]. This makes it

essential for any prompts to provide as much context and relevant information as possible, to ensure the model performs the task better. Hadi et al. also mention that such kind of prompts leads to more accurate responses from the model [39]. Zhao et al. state that there are four features of a good prompt; it should contain a clear task description, appropriate input data, contextual or background information for the task, and a prompt style that contains positive reinforcements [37]. Additionally, the prompt style should also break down complex prompts into sub-tasks, which would help the LLM reason step by step [37]. For example, Zhao et al. state that using a prefix like “You are an expert in this field” can boost LLM performance [37].

3.2.1 Zero-shot Learning

Zero-shot learning is a machine learning feature where a model is trained to solve new tasks without labeled training data [45]. LLMs have shown an ability to perform tasks that they were not originally trained for, and prompt engineering is one of the ways to improve this ability [43]. This indicates that with a good description of a task, an LLM can perform tasks that involve textual input and outputs. Other ways to improve the zero-shot learning capabilities of LLMs are bespoke training, prompt aggregation, and weak supervision. This thesis will use a zero-shot approach to the prompts since there is no research into the capabilities of LLM for performing the triage process of SIEM alerts using this or any other approach.

3.2.2 Chain-of-thought Prompting

Chain-of-thought (CoT) prompting compels LLMs to follow and provide intermediate reasoning steps when completing a task [45]. This technique has shown the effectiveness of LLMs in zero-shot tasks. This is because chain-of-thought prompting encourages LLMs to elaborate on their reasoning, and this has been shown to improve performance [10].

3.2.3 LLM Public APIs

Due to the high computational cost of building and training LLMs, Zhao et al. highlight that model checkpoints, also known as Application Programming Interfaces (API) can be used for experimental study [37]. For example, OpenAI provides interfaces to several models, including GPT-3, GPT-3.5, and GPT-4. OpenAI also provides an API called Assistants API, which makes developing task-specific agents easier. Using this API, researchers can create agents with specific prompts to fulfill the needs of tasks [37]. For example, the Python code snippet in Snippet 3.1 will create an agent that is a math tutor who uses code to solve math problems [46].

```
1 from openai import OpenAI
2 client = OpenAI()
3
4 assistant = client.beta.assistants.create(
5     name="Math Tutor",
6     instructions="You are a personal math tutor. Write and run code
7     to answer math questions.",
8     tools=[{"type": "code_interpreter"}],
9     model="gpt-4o",
10 )
```

Snippet 3.1: Python code to create an OpenAI assistant math tutor

Fang et al. used the assistant API with GPT-3.5 and GPT-4 in a research that tested the ability of LLM agents to exploit one-day vulnerabilities autonomously [47]. The research used the Together AI API to assess other open-source LLMs. The Together API provides an interface to models like LLama, Mistral, and OpenHermes.

3.3 LLMs and Threat Detection

The rest of this chapter will focus on research into applying LLMs in cybersecurity with special emphasis on threat detection. As discussed in Chapter 2, the Tier 1 analyst triage process is a part of the detection stage of the NIST Computer Incident Handling guide. The application of LLMs in different phases of cybersecurity is currently growing. For example, Motlagh et al. identified several applications of LLMs through the lens of the NIST cybersecurity framework [10]. The survey shows the application of LLMs in the ‘Identify’, ‘Protect’, ‘Detect’, and ‘Respond’ functions of the framework. The ‘Detect’ function encompasses the work performed by Tier 1 analysts in the SOC. The ‘Detect’ function also includes log analysis, and Karlsen et al. discovered that LLM-based approaches displayed superior performance compared to syntactic approaches in analyzing two of the three used datasets [48]. Motlagh et al. also identifies VulDetect [49], a vulnerability detection framework that detects vulnerable code using GPT-2. CyBERT [50], an enhanced version of BERT, which was trained using open-source Cyber Threat Intelligence (CTI) data, displayed a high performance and usefulness for common SOC tasks. Using chain-of-thought prompting, Koide et al. demonstrated a technique for detecting phishing websites using LLMs [51]. The system that used GPT-4 had an accuracy of 99.7%, suggesting that LLM-based detection mechanisms can be effective in threat detection.

3.3.1 Other Cybersecurity applications of LLMs

Apart from threat detection, LLMs have been used in other areas of cybersecurity. This section covers some of these applications. de Lima et al. uses LLMs to identify risk through application reviews [52]. In another research, Charalambous et al. prescribe a three-step process that uses LLMs to automate vulnerability remediation [53]. LLMs have also been extensively applied in offensive security, the area of cybersecurity concerned with developing attacks and testing cybersecurity architec-

ture. Motlagh et al. mentions several LLM-based tools that cover several MITRE ATT&CK Tactics [10]. The tools include malware generators, phishing emails and websites creators, and password guessing. Fang et al. also demonstrated that an LLM agent using GPT-4 could exploit 87% of 15 one-day vulnerabilities when given an adequate description of the vulnerability [47]. Kaheh et al. proposed Cyber Sentinel, a dialogue engine that chains together several GPT-4 models to streamline SOC tasks [3]. Cyber Sentinel can query and analyze cyber threat intelligence data and remediate identified threats using prompt engineering.

However, the continuous advancements in the development and applications of LLMs mean malicious actors can also use these models. Two famous examples of LLM-based tools created for executing malicious actions are WormGPT and FraudGPT [54]. WormGPT is designed to automate the creation of spear phishing emails, while FraudGPT can be used to create enticing content to trick users into clicking on malicious links. The use of LLMs for malicious intent is not limited to LLMs alone but applies to other AI approaches. Like other AI approaches, LLMs are subject to adversarial attacks that can affect their learning and decision-making [35, 39]. For example, Chao et al. describe an algorithm that jailbreaks black box LLMs to violate their safeguards and causes the LLM to generate malicious content [55].

3.3.2 LLM Limitations

Different researchers have highlighted several limitations of LLMs. For example, Karlsen et al. assert that although the evaluated LLMs showed potential promises in log analysis, the models had a high computational cost [48]. The high computational cost can be reduced by better adapting the Large Language Model to the log files. Wu et al. posit that LLMs like ChatGPT can generate responses with factual errors, often referred to as hallucinations [11]. Additionally, Chang et al. identified that

LLMs are sensitive to adversarial prompts, which are prompts designed to deceive LLMs to induce damaging, biased, or inaccurate outputs from them [44].

LLMs as autonomous agents

LLMs can be used as “copilots” or autonomous agents in a system. As copilots, LLMs can provide assistance and suggestions but does not make decisions without human intervention. On the other hand, an autonomous agent can make choices and act without human input. The success of LLMs as the backbone of autonomous agents relies on the capabilities discussed in previous sections, including in-context learning and the ability to self-reflect, process documentation, and take actions via integrated tools [47].

However, given the limitations of LLMs, it can be argued that they are better used as “assistants” rather than autonomous agents, as concluded by Hadi et al. in their survey [39]. However, the success demonstrated by LLMs in [47], and several similar research is encouraging.

Ethics and Privacy

Other issues in the discourse of LLMs and artificial intelligence in general are ethics and privacy. Although this thesis is not primarily concerned with the ethics of developing and using artificial intelligence, it should be noted that the ethics issue is incredibly complex in cybersecurity because the same models that can be trained for good can also be used for malicious intent. For example, Wu et al. make an essential point about how ChatGPT inadvertently lowers the skill level required to create and facilitate cyber-attacks [11]. Furthermore, LLMs have been seen to internalize and propagate social biases based on the information in crawled training datasets, which can hurt society [44].

Privacy issues with LLMs have already been identified in healthcare [44]. These

issues also affect cybersecurity applications of LLMs that rely on confidential data and external models. To mitigate these privacy issues, LLM datasets are usually scanned with rules to remove Personally Identifiable Information (PII) before being used [37]. However, Hadi et al. advise organizations using LLMs with confidential data to ensure that techniques like encryption and appropriate data protection policies are implemented [39]. However, Yao et al. conclude in their research that LLMs contribute more positively than negatively to security and privacy [38].

3.3.3 LLMs and the SOC Tier 1 Analyst Triage Process

This section summarises the key observations from different literature concerning LLMs and the SOC Tier 1 analyst triage process. The SIEM system plays a pivotal role because it is responsible for creating the alerts that will be analyzed by the Tier 1 analyst [3, 7]. Furthermore, concerning LLM prompts, there is a need to provide carefully designed prompts that follow best practices to enhance the model's performance in the given task. The prompts should contain positive reinforcement for the ability of the LLM, and describe the triage process as best as possible, including the expected output from the LLM agent. Although some LLMs can receive input in different forms apart from text, the triage process will be best served by presenting the alerts in a textual form. Fang et al. reported failures when LLM agents were instructed to navigate a web application [47]. To avoid this issue, it is best to provide the alerts in a textual form and receive the output similarly.

4 Framework and Experiment

Design

The methodology of this thesis is presented in two sections. The first section introduces a novel system that leverages an LLM agent to automate the SOC Tier 1 analyst triage process. This innovative approach is a key highlight of the research. The second section details the experiment procedure and evaluation metrics used to test the LLM agents.

4.1 Triage System

This section describes the components of the system, which uses an LLM agent to perform the SOC Tier 1 analyst triage process. The system comprises an alert generation and processing module, the LLM agent, and a reporting component.

4.1.1 Alert Generation and Processing Component (SIEM)

Just like in a conventional SOC, this component acts as the central hub for the logs and alerts generated by the different devices in the network. This component is responsible for log collection, correlation, and alert generation and processing. The alert generation function requires an analysis engine to identify anomalies in collected alerts and generate alerts that indicate possible incidents. Additionally, the

component can present and store the alerts in JavaScript object notation (JSON) format. In this thesis, the SIEM module used is Wazuh, an open-source SIEM solution that was described in Chapter 2. Wazuh generates alerts in JSON which is the same format that will be sent to the LLM agent for triage. A typical Wazuh alert contains the rule description and severity of the rule that triggered the alert. Furthermore, other fields in the alert are extracted from the raw logs, with the SIEM analysis engine adding relevant compliance details. The alerts are saved to `/var/ossec/logs/alerts/alerts.json` on the Wazuh server. A sample Wazuh alert is provided in Appendix A.

4.1.2 LLM Agent

The LLM agent uses the underlying training of the LLM with carefully crafted prompts to classify ‘interesting’ alerts and ‘non-interesting’ alerts. The definition of an interesting alert is an alert that requires further investigation because it might indicate a cybersecurity incident. A non-interesting alert is an alert that is either informational or can be treated as a false positive. This design closely follows the triage process of a SOC Tier 1 analyst, including the expected outputs from the process. The LLM agents in this thesis are accessed using publicly available APIs: the OpenAI Assistant API and the Together API.

4.1.3 Reporting Component

The reporting component uses the output from the LLM to generate a report that contains the number of interesting and non-interesting alerts. The report also includes a Uniform Resource Locator (URL) for the alerts to facilitate further review.

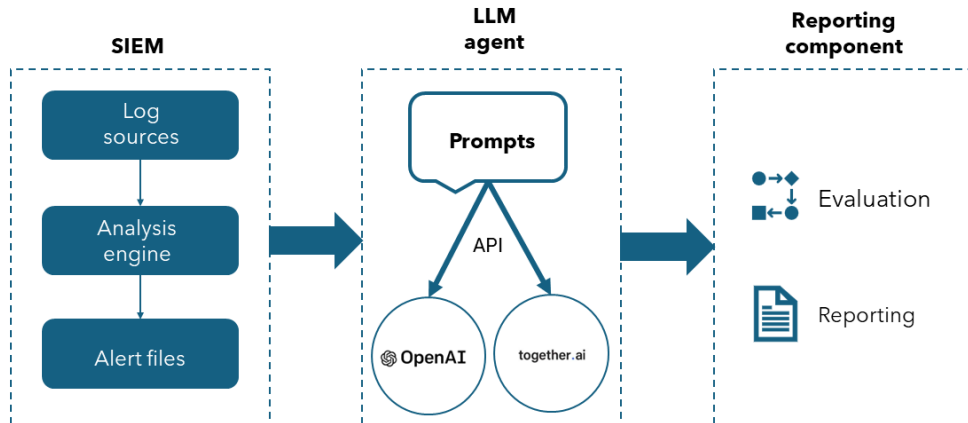


Figure 4.1: Framework for automated alert triage

4.2 Experiment Design

This section will describe the experimental procedures used to evaluate the performance of the LLM agents using the framework described in the previous section.

4.2.1 Dataset Preparation

Log collection and alert generation

Similar to the process used by Ban et al. [8], logs were collected and forwarded to the Wazuh server from three different endpoints using the Wazuh agent. The endpoints had different operating systems installed, Windows, Ubuntu Linux, and macOS, the most popular desktop operating systems in use worldwide [56]. The versions were Windows 11, Ubuntu Linux 22, and macOS Ventura. Figure 4.3 shows the Wazuh dashboard displaying the connected agents. Logs from these agents were forwarded to the Wazuh server for correlation and analysis. To generate logs that indicate “attacks”, CALDERA was employed. CALDERA is a cybersecurity frame-

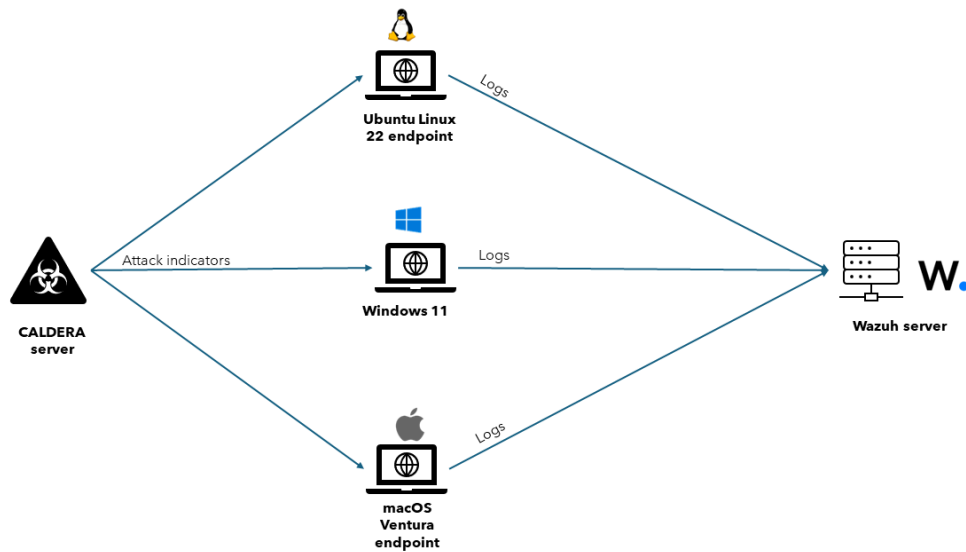


Figure 4.2: Log collection architecture

work developed by MITRE, which allows cybersecurity teams to perform adversary emulation. CALDERA enables automated security assessments and, in this case, was used to trigger alerts in the Wazuh analysis engine.

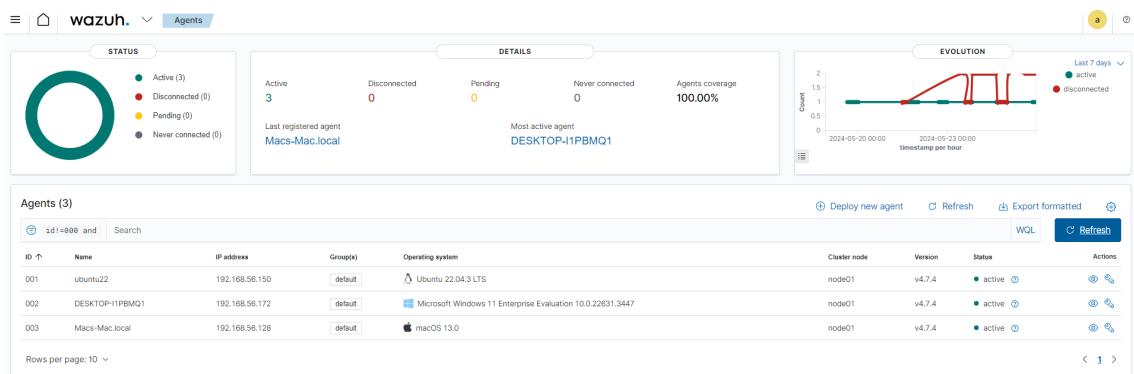


Figure 4.3: Wazuh agent dashboard

Table 4.1: Alert classification

Operating system	Interesting	Non-interesting
Ubuntu Linux 22	12	11
Windows 11	11	15
macOS Ventura	2	7

Alert selection

From the alerts generated, 58 unique alerts were selected. The alerts contained 33 uninteresting and 25 interesting alerts. The 25 interesting alerts are incident-related from the attack simulation performed using CALDERA. The interesting alerts were representative of the MITRE ATT&CK framework, which provides a good coverage of the different kinds of alerts encountered by SOC Tier 1 analysts. The alerts contain a unique ID and are labeled appropriately, whether interesting or not. Furthermore, the alerts were saved to a JSON file, with each alert being in the JSON format, albeit without the labels.

4.2.2 LLM Agent Alert Triage

To measure the performance of the LLM agent in the triage of the selected alerts, the OpenAI Assistant API and Together AI API were used. The OpenAI Assistant API granted access to the GPT-3.5 and GPT-4 models. Meanwhile, the Together AI API enabled access to other open-source models. The focus was testing the top models in the ChatBot Arena [57], a leaderboard that ranks LLM based on their performance in different categories.

Models

The following models were tested using the triage framework:

1. GPT-4o
2. GPT-3.5

3. OpenHermes-2.5-Mistral (7B)
4. Meta Llama 3 70B Instruct
5. Mixtral-8x22B Instruct v0.1

GPT-4o (GPT-4 Omni) is a multilingual and multi-modal Large Language Model developed by OpenAI and released in May 2024. It can process and generate text, images, and audio [58]. The knowledge cutoff for the model used in this thesis is October 2023. GPT-3.5 is an older generation of GPT-4o with a knowledge cutoff of September 2021.

Llama 3 is an open-source model developed and released by Meta. It comes in two sizes: 8B and 70B parameters. The variant tested in this thesis contains pre-trained and instruction-tuned generative text models in 70B sizes. The 70B variant performs better and has a knowledge cutoff of December 2023 [59]. Mixtral 8x22B is an open-source model developed by Mistral AI and is known for its cost efficiency [60]. OpenHermes 2.5 Mistral 7B continues the OpenHermes 2 model, another open-source LLM. The model performs worse than the other models in this test using LLM evaluation benchmarks. However, this model was selected to see how well it would perform against the other more advanced models.

Prompts

The LLM APIs were queried using prompts that used optimization techniques recommended by [37, 39]. The prompts include positive reinforcement and a clear description of the required tasks. The description of the task also contains the needed outputs or deliverables that are expected of the agent. The prompts instruct the LLM agent to access the alerts saved to a file, evaluate them, and return feedback based on their evaluation. The APIs are accessed using Python, a high-level general-purpose programming language. A significant difference between the

OpenAI Assistant API and the Together AI API is the alert files are handled. The OpenAI Assistant API allows the upload of files separate from the prompts, while the file contents are added to the prompts used to access the models using the Together AI API.

Open AI Python script

The OpenAI Python script starts with importing the OpenAI package and creating a client using the API key. Using the instructions from OpenAI, the script creates an assistant using the prompt in Snippet 4.1.

```
1 from openai import OpenAI
2
3 client = OpenAI(api_key=$API_KEY)
4
5 assistant = client.beta.assistants.create(
6     name="Tier 1 SOC Analyst",
7     instructions="You are an experienced tier 1 SOC analyst. Review
8     the SIEM alerts and decide which alerts are "
9     "interesting or not -interesting. An
10    interesting alert indicates a potential incident that needs "
11    "to be investigated further. A non-interesting
12    alert is either informational or a false positive. "
13    "Make this decision based on the details in the
14    alert and respond with your decision",
15    tools=[{"type": "file_search"}],
16    model="gpt-4o",
17 )
```

Snippet 4.1: Create OpenAI Assistant

The script continues in Snippet 4.2 with creating a vector store for the alert file, and then uploading the alert file `alert.json` to the OpenAI assistant API.

```
1 # Create a vector store called "Alert"
```

```
2 vector_store = client.beta.vector_stores.create(name="Alert")
3
4 # Ready the files for upload to OpenAI
5 file_paths = ["alert.json"]
6 file_streams = [open(path, "rb") for path in file_paths]
7
8 # Use the upload and poll SDK helper to upload the files, add them
   to the vector store,
9 # and poll the status of the file batch for completion.
10 file_batch = client.beta.vector_stores.file_batches.upload_and_poll
   (
11     vector_store_id=vector_store.id, files=file_streams
12 )
13
14 # Print the status and the file counts of the batch to see the
   result of this operation.
15 print(file_batch.status)
16 print(file_batch.file_counts)
17
18 assistant = client.beta.assistants.update(
19     assistant_id=assistant.id,
20     tool_resources={"file_search": {"vector_store_ids": [
   vector_store.id]}},
21 )
22
23 # Upload the user provided file to OpenAI
24 message_file = client.files.create(
25     file=open("alert.json", "rb"), purpose="assistants"
26 )
```

Snippet 4.2: Upload alert file to OpenAI assistant

The last part of the script creates a message thread and sends a message instructing the LLM agent to review the uploaded alert file and provide an output

based on its analysis. The structure of the expected output is defined in the message content sent to the OpenAI assistant, as seen in Lines 8 and 9 of Snippet 4.3. The script ends by printing the output of the LLM agent to the execution window.

```
1 # Create a thread and attach the file to the message
2 thread = client.beta.threads.create(
3     messages=[
4         {
5             "role": "user",
6             "content": "There is 1 Wazuh alert in the file. Please
7 review the alert and indicate whether it is "
8             "'interesting' or 'not-interesting'. Your "
9             "message should use the following format:
10 alert_id: \n alert_description: \n alert_decision: "
11             "\nreason:",
12             # Attach the new file to the message.
13             "attachments": [
14                 {"file_id": message_file.id, "tools": [{"type": "
15 file_search"}]}
16             ],
17         }
18     ]
19 )
20
21 # The thread now has a vector store with that file in its tool
22 resources.
23 print(thread.tool_resources.file_search)
24
25 # Use the create and poll SDK helper to create a run and poll the
26 status of
27 # the run until it's in a terminal state.
28
29 run = client.beta.threads.runs.create_and_poll(
```

```
25     thread_id=thread.id, assistant_id=assistant.id
26 )
27
28 messages = list(client.beta.threads.messages.list(thread_id=thread.
    id, run_id=run.id))
29
30 message_content = messages[0].content[0].text
31 annotations = message_content.annotations
32 citations = []
33 for index, annotation in enumerate(annotations):
34     message_content.value = message_content.value.replace(
        annotation.text, f"[{index}]")
35     if file_citation := getattr(annotation, "file_citation", None):
36         cited_file = client.files.retrieve(file_citation.file_id)
37         citations.append(f"[{index}] {cited_file.filename}")
38
39 print(message_content.value)
40 print("\n".join(citations))
```

Snippet 4.3: Create and send message to OpenAI assistant

Together AI Python script

The Together AI Python script is much smaller although it uses the same prompt used with the OpenAI Assistant API. The major difference is that alert is appended to the message in the Together AI API script, which is different from OpenAI Assistant API which provides an interface to upload the files directly. Another difference is that the prompt to create the Tier 1 analyst assistant is also appended to the message sent to the tested models. The full script is shown in Snippet 4.4.

```
1 import json
2 from together import Together
3
```

```
4 client = Together(api_key=$API_KEY)
5
6
7 with open('alert.json') as file:
8     alerts = json.load(file)
9
10 for index, alert in enumerate(alerts):
11     message = (f"You are an experienced tier 1 SOC analyst. Review
12     the SIEM alerts and decide which alerts are "
13     f"      interesting      or not -interesting. An
14     interesting alert indicates a potential incident that needs "
15     f"to be investigated further. A non-interesting
16     alert is either informational or a false positive. "
17     f"Make this decision based on the details in the
18     alert and respond with your decision. There is one "
19     f"Wazuh alert in JSON format. Please review the
20     alert and indicate whether it is 'interesting' "
21     f"or 'not-interesting'. Your message should use the
22     following format: alert_id: alert_description: "
23     f"alert_decision: reason:' Alert: {alert}]"
24
25     response = client.chat.completions.create(
26         model="meta-llama/Llama-3-70b-chat-hf",
27         messages=[{"role": "user", "content": f'{message}'}]
28     )
29     print(f'{index+1}\n{response.choices[0].message.content}\n')
```

Snippet 4.4: Together AI prompt

4.2.3 Evaluation Metrics

This section will describe the metrics used to evaluate the LLM performance in the alert triage process. Hadi et al. state that accuracy, precision, recall, and F1 score

are fundamental to evaluating the performance of a model on a given task [39]. These metrics can also be used to identify areas of improvement.

The descriptions of these metrics are linked to the intermediate measures described below:

- True Positive (TP): This is the number of alerts correctly labeled as interesting by the model.
- False Positive (FP): This is the number of alerts incorrectly labeled as interesting by the model.
- True Negative (TN): This is the number of alerts correctly labeled as non-interesting by the model.
- False Negative (FN): This is the number of alerts incorrectly labeled as non-interesting by the model.

The accuracy is the percentage of the dataset that the model correctly labels. It is used to show how often the model is correct overall. It is calculated using the formula below where n is the number of alerts in the dataset:

$$Accuracy = \frac{TP + TN}{n} \quad (4.1)$$

The precision is the probability that the labeled interesting alerts are correctly classified. This metric evaluates how correct the model is in predicting interesting alerts. It is calculated using the formula:

$$Precision = \frac{TP}{TP + FP} \quad (4.2)$$

The recall is the probability that the labeled interesting alerts are correctly classified. It measures whether the model can find all the interesting alerts in the dataset. It is calculated using the formula:

$$Recall = \frac{TP}{TP + FN} \quad (4.3)$$

The F-score is the harmonic mean of the precision and recall and can evaluate the balance of the two:

$$F - score = \frac{Recall \cdot Precision}{Recall + Precision} \quad (4.4)$$

The models can also be evaluated using the false positive rate (FPR). The FPR is the probability that an uninteresting alert is incorrectly predicted to be interesting by the model.

$$FPR = 2 \cdot \frac{FP}{FP + TN} \quad (4.5)$$

5 Results and Discussion

This research involves building and testing a system that uses LLM agents to perform the typical triage process of tier 1 analysts in a SOC. The research tested 5 LLM agents against 58 alerts from an open-source SIEM. This chapter describes the alerts, results, and analysis of the results.

5.1 Description of Alert Dataset

Similar to methods used by [7, 25], a bespoke alert dataset was designed and used in this research. The dataset contained 58 alerts generated by the SIEM using logs from three different operating system endpoints. Of the 58 alerts, 26 were labeled as interesting, while the rest were generic events with little or no security significance. The interesting alerts were generated using adversary emulation techniques performed by CALDERA. Thirty-two non-interesting alerts were generated from normal system operations. The rest of this section will describe the alerts.

5.1.1 Windows Alerts

The non-interesting Windows alerts include Wazuh agent activities, process activities by genuine Microsoft binaries, and generic system activities. The interesting alerts triggered by CALDERA included process activities by suspicious binaries, obfuscation actions performed in the Windows registry, user account activities involving the Administrator and Guest accounts (which should always be disabled in

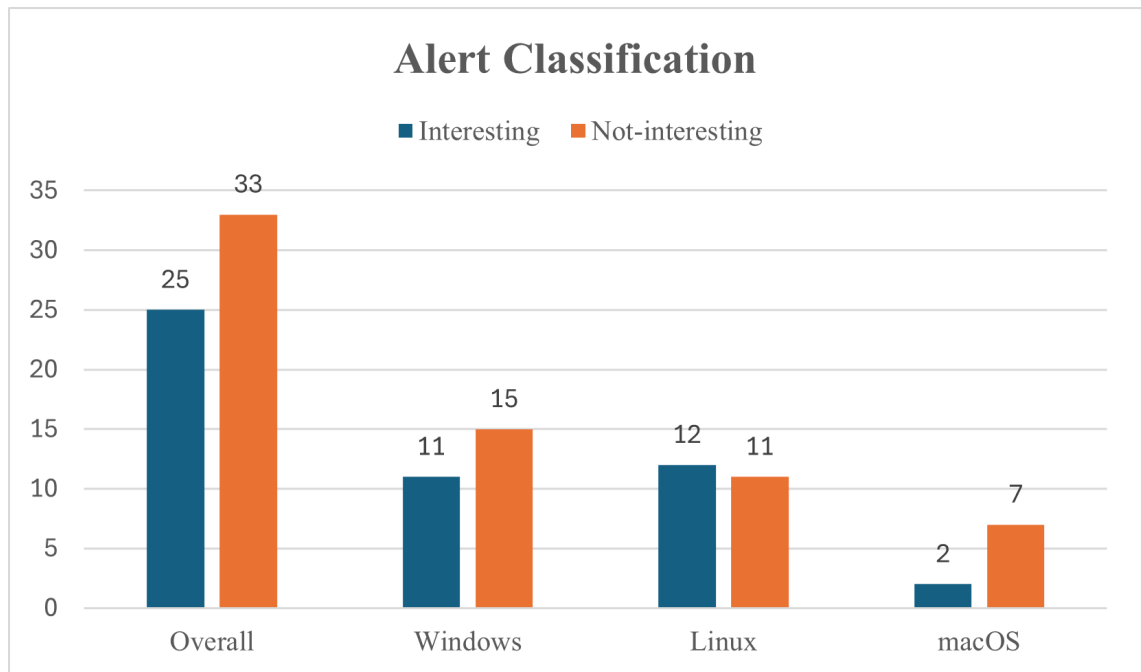


Figure 5.1: Alert classification overall and for each operating system

Windows endpoints), and clearing of audit logs, which is an activity performed by adversaries to hide their footprint.

5.1.2 Linux Alerts

The non-interesting Linux alerts included authentication success and timeout alerts. The non-interesting alerts included creating privileged login sessions by a “trusted” user, Wazuh agent activities, a port change alert, and generic log and Debian package activities. The interesting alerts included user and group tampering, root account exploitation, and account brute-force attacks.

5.1.3 macOS Alerts

The closed-source nature and lack of in-depth documentation make generating security alerts for macOS endpoints difficult. This research contained generic screen lock and unlock alerts, port status changes, and Wazuh agent activities. The interest-

Table 5.1: Windows alerts

Alert ID	Description	Label
1716460351.2348	New Wazuh agent connected	Not-interesting
1716460446.1542036	The start type of the Background Intelligent Transfer Service service was changed from auto start to demand start.	Not-interesting
1716460457.1543868	Windows logon success (eventID 4624) on host DESKTOP-IIPBMQ1.	Not-interesting
1716461131.1560198	Wazuh agent started.	Not-interesting
1716461133.1560539	Discovery activity executed	Not-interesting
1716461783.1683166	Software protection service scheduled successfully.	Not-interesting
1716461784.1684765	Process loaded taskschd.dll module. May be used to create delayed malware execution	Not-interesting
1716465387.1851263	Application Compatibility Database launched	Not-interesting
1716484801.2290155	WSearch was unavailable to handle a notification event.	Not-interesting
1716484801.2291659	Windows User Logoff.	Not-interesting
1716553887.166930	Printer spooler service loaded a dll file. Possible PrintNightmare exploit: CVE-2021-34527	Not-interesting
1716553886.161133	Sysmon - Suspicious Process - svchost.exe	Not-interesting
1716553907.185312	The Windows search service started.	Not-interesting
1716579663.1598575	Windows logon success.	Not-interesting
1716579671.1615206	Explorer process was accessed by powershell.exe, possible process injection	Not-interesting
1716579897.1687901	Windows command prompt started by an abnormal process	Interesting
1716579897.1705930	User account enabled or created.	Interesting
1716580030.2239372	Scripting interpreter spawned a new process	Interesting
1716580845.2470765	Suspicious Windows cmd shell execution - "wmic useraccount get /ALL /format:csv"	Interesting
1716581007.2585021	Value added to registry key has Base64-like pattern	Interesting
1716581058.2628503	Domain users group changed.	Interesting
1716581058.2631698	User account enabled or created.	Interesting
1716581605.2740787	User account changed.	Interesting
1716582127.2814135	The audit log was cleared.	Interesting
1716583670.3337247	User account disabled or deleted.	Interesting
1716583692.3439135	User account disabled or deleted.	Interesting

Table 5.2: Linux alerts

Alert ID	Description	Label
1716123945.13961	sshd: authentication success.	Not-interesting
1716123945.14470	PAM: Login session opened.	Not-interesting
1716124422.24818	Successful sudo to ROOT executed.	Not-interesting
1716124706.25744	PAM: Login session closed.	Not-interesting
1716126525.33374	sshd: Timeout while logging in.	Not-interesting
1716125093.30331	PAM: Login session opened.	Not-interesting
1716128884.33634	Log file rotated.	Not-interesting
1716134605.0	Wazuh agent started.	Not-interesting
1716134607.321	Listened ports status (netstat) changed (new port opened or closed).	Not-interesting
1716134616.13709	Wazuh server started.	Not-interesting
1716154961.54191	Dpkg (Debian Package) half configured.	Not-interesting
1716327886.22797	New group added to the system.	Interesting
1716327886.22212	New user added to the system.	Interesting
1716327954.24067	Information from the user was changed.	Interesting
1716329950.41028	Group (or user) deleted from the system.	Interesting
1716330807.42863	Integrity checksum changed.	Interesting
1716407215.30217	Root's crontab entry changed.	Interesting
1716407561.34643	Successful sudo to ROOT executed.	Interesting
1716407661.46394	PAM: User changed password.	Interesting
1716410397.125073	Group (or user) deleted from the system.	Interesting
1716410399.125547	Group (or user) deleted from the system.	Interesting
1716411003.145038	PAM: Multiple failed logins in a small period of time.	Interesting
1716411005.153348	sshd: brute force trying to get access to the system. Authentication failed.	Interesting

Table 5.3: macOS alerts

Alert ID	Description	Label
1716658325.0	New wazuh agent connected.	Not-interesting
1716658333.1528	Wazuh agent started.	Not-interesting
1716658337.2271	Listened ports status (netstat) changed (new port opened or closed).	Not-interesting
1716659042.294078	Session 4433 has been created.	Not-interesting
1716659070.296176	Session 4433 has been destroyed.	Not-interesting
1716660702.633659	Screen locked with userID:501.	Not-interesting
1716661307.634213	Screen unlocked with userID:501.	Not-interesting
1716659010.291893	/usr/libexec/sshd-keygen-wrapper has been granted permission to kTCCServiceSystemPolicyAllFiles at 17:43:46.	Interesting
1716665618.647220	sshd: brute force trying to get access to the system. Authentication failed.	Interesting

ing macOS alerts were triggered by permission tampering and account brute-forcing attacks.

5.2 Numerical Results

This section discusses the numerical results of the experiments. Table 5.4 shows the results obtained after testing the different LLM agents against the prepared alert dataset. Chapter 4 describes the criteria used to evaluate the LLM agents in the table. The numerical results show the performance of each LLM in the overall alert dataset, which contains alerts for the three operating systems used in the experiment.

The table shows that the Llama 3 agent performs the best in accuracy. This shows the model is the most balanced of the five models tested in this experiment. Llama 3’s performance is balanced because it identifies interesting and not-interesting alerts well in the dataset. The results also indicate that the latest models from the models tested produced better results than older models. For example, GPT-4o, Llama 3, and Mixtral-8x22B Instruct v0.1 outperformed older models like GPT-3.5 and OpenHermes-2.5-Mistral (7B) in almost all categories.

The best three models for identifying interesting alerts were GPT-4o, Llama 3,

and Mixtral-8x22B Instruct v0.1. Digging deeper into the performance data, GPT-4o identified all 25 interesting alerts from every operating system, which is reflected in its recall percentage. Llama 3 did not identify one of the macOS interesting alerts and rather labeled the alert as a false positive.

The best-performing model at identifying uninteresting alerts was Llama 3, which correctly identified 29 of the 33 uninteresting alerts. Interestingly, the worst performer at identifying not-interesting alerts was GPT-4o, indicating a high ‘paranoia level.’ GPT-4o’s performance at identifying non-interesting alerts is reflected in its FPR, which is the highest for all the models tested. Ban et al. noted that a high FPR can affect the effectiveness of SOC operations, but in a field like LLMs, this can be a good thing because it gives room for further tuning to improve performance [8]. Also, OpenHermes-2.5-Mistral (7B) identified all the macOS alerts as not interesting, indicating that it might have had an issue with the alert structure or field names.

The performance of the tested models reflects their ranking on [57], except in the case of Llama 3, which outperformed the higher-ranked GPT-4o. This shows an emerging ability in LLMs that might require benchmarks different from those conventionally used for general evaluation. The results of this research resemble the results and sentiments from Fang et al. who successfully used LLMs for a cybersecurity task like exploiting 1-day vulnerabilities [47].

The numerical results suggest that Llama 3, GPT-4o, and Mixtral-8X22B-Instruct-v0.1 are suitable for the alert triage process if trained and tuned specifically for a specific organization’s data. However, in the next section, this research will evaluate if tuning using prompts can improve the baseline performance of these models.

Table 5.4: Performance comparison of LLM agents

Models	Accuracy (%)	Precision (%)	Recall (%)	F-score	FPR(%)
GPT-4o	84.5	73.5	100	42.4	54.6
GPT-3.5	77.6	77.3	68	36.2	30.3
Mixtral-8x22B Instruct v0.1	84.5	78.6	88	41.5	36.4
Meta Llama 3 70B Instruct	91.4	85.7	96	45.3	24.2
OpenHermes-2.5-Mistral (7B))	70.7	70	56	31.1	36.4

5.3 Qualitative Analysis

As noted above, part of the prompt used to test the LLM agents contained a required output: to state the reason or motivation for the decision. This section will try to qualitatively analyze the differences between the tested models by looking at the reasons the LLM models provided for their choices. Firstly, the provided outputs by the LLM agents showed different degrees of verbosity and reasoning. The more advanced models like Llama 3, Mixtral-8x22B Instruct v0.1, and GPT-based models showed a higher degree of verbosity in their responses, unlike the less advanced OpenHermes-2.5-Mistral (7B). For example, in Listing 5.1, you can see the responses of three of the models to the same alert using the same prompt.

```

1 OpenHermes-2.5-Mistral (7B)
2 alert_description: New wazuh agent connected.
3 alert_decision: not-interesting
4 reason: The alert is informational and does not indicate a
   potential incident.
5
6 Mixtral-8x22B Instruct v0.1
7 alert_description: New wazuh agent connected.
8 alert_decision: not-interesting
9 reason: This alert indicates that a new Wazuh agent has connected
   to the server. This is an informational alert and does not
   indicate any malicious activity or a potential incident that
   needs to be investigated further.

```

10

```
11 Llama 3
12 alert_id: 1716658325.0: alert_description: New wazuh agent
    connected.: alert_decision: not-interesting: reason: This alert
    is informational, indicating that a new Wazuh agent has
    connected to the manager, which is a normal and expected event
    in a Wazuh deployment. It does not indicate any potential
    security incident.
```

Snippet 5.1: Output from LLM agents

Analyzing the level of verbosity is important because it can uncover right and wrong patterns in the way that the LLM agents process the alerts. The lack of verbosity in the responses provided by OpenHermes-2.5-Mistral (7B) is not a configuration issue because all the parameters for the APIs used were kept the same throughout the tests. A deeper look at the reports from the LLM agents also revealed that the most important metric the LLMs cited in their decisions was the severity attached to the alert by the SIEM solution. This can be seen in the output in Listing 5.2.

```
1 GPT-4o
2 alert_description: Session 4433 has been destroyed.
3 alert_decision: not-interesting:
4 reason: This alert appears to be a normal system event, indicating
    that a session has been destroyed. There is no indication of
    malicious activity or potential security threat. The alert level
    is 3, which is informational, and the rule groups are related
    to compliance and auditing, further supporting the conclusion
    that this is a non-interesting alert.
```

Snippet 5.2: LLM agent referencing alert severity

However, some of the tests involved similar alerts triggered by the same rule. One of the alerts was triggered by normal user activity, while the other one was triggered using adversary emulation techniques. An example of such an alert is the

Linux alert ‘Successful sudo to ROOT executed.’ The first alert was triggered by a ‘trusted’ user initiating a privileged shell. From the alert context, the user is allowed to do this. The second alert shows a Linux privilege escalation technique using the command `/usr/bin/tee -a /etc/rc.local` to open a privileged shell on the endpoint. Alerts like this can reveal how the LLM handles additional contexts, even when the other parameters are the same. The only LLM agent that successfully identified both was OpenHermes-2.5-Mistral (7B). However, analyzing the response shows that the LLM agent did not understand the alert contexts. However, the responses provided by Mixtral-8x22B Instruct v0.1, GPT-4o, and Llama 3 can serve as a baseline for further testing.

5.3.1 Testing Understanding of Alert Context

As described in Chapter 2, understanding the context of an alert is important for deciding whether an alert is informational or a potential incident. Furthermore, it is important for analysts to identify an incident incorrectly rather than misinterpret alerts that are incident-related. Reflecting on the performance of the LLM agents, GPT-4o correctly identified every interesting alert used in the testing. The closest performer was Llama 3, which identified all but one of the interesting alerts. This performance means it is possible to progressively train and retune these LLM agents to support the SOC analyst triage process by providing additional contexts in the prompts. For example, the assumption was that GPT-4o’s level of paranoia meant that it continuously identified certain alerts as interesting because it did not have the required context in the prompts to declare them as informational. To test this theory, the ‘Successful sudo to ROOT executed.’ alert was tested. Context was added to the prompts by stating that the vagrant user regularly performs sudo activities, and as such, all ‘Successful sudo to ROOT executed.’ alerts for this user should be tagged as informational, and alerts without this user should be flagged

as interesting. This information was added as a message to GPT-4o but placed as part of the prompt to Llama 3 using the Together AI API. As seen in Listing 5.3, the alerts were correctly identified as ‘interesting’ and ‘not-interesting’ by GPT-4o. Llama 3, on the other hand, hallucinated the vagrant user into the logs and declared the interesting alert as not interesting, showing it might have difficulty understanding the alert context and following instructions in the prompts. A reason for GPT-4o’s performance might be that the model using the Assistant API can analyze the file and the prompts separately. This is unlike the Together AI API, which requires appending the alerts to the prompts. However, this test is limited in scope, and it would be exciting to test GPT-4o’s ability further.

```
1 GPT-4o
2 alert_description: Successful sudo to ROOT executed.
3 alert_decision: interesting
4 reason: The alert indicates a successful sudo to ROOT command
      executed by the root user; however, it is pertinent to note that
      the alert does not involve the vagrant user. Therefore, this
      alert warrants further investigation as it does not fall under
      the exempted user category.
5
6 alert_id: 1716124422.24818
7 alert_description: Successful sudo to ROOT executed.
8 alert_decision: Not-interesting
9 reason: The vagrant user regularly performs sudo activities. As
      such, all "Successful sudo to ROOT executed" alerts for this
      user do not require further investigation.
```

Snippet 5.3: GPT-4o context analysis

5.4 Cost Analysis

Following the procedures used in similar research by Fang et al. [47], this section will evaluate the cost of using LLM agents for the SOC analyst triage process. The evaluation will provide the estimated cost for a small or medium-sized organization. At the time of writing, the most advanced GPT-4o model costs \$5.00 for 1 million input tokens and \$15.00 for 1 million output tokens, where tokens are comparable with the amount of words. OpenAI states that 750 words are about 1000 tokens. The average word count for the alerts used in this research is 201. In their study, Pecchia et al. found that the SIEM in a production environment produced 1000 alerts on average each day [61]. This amounts to about 201000 words, which would be roughly 150,000 tokens, resulting in about \$1 per day. This cost could be higher or lower depending on the model, API, and volume of prompts. However, it can be argued that an organization might benefit more from internal training and hosting an open-source LLM. This would be comparably cheaper and provide more organization-specific analysis.

5.5 Research Implications

This research suggests that LLM agents can be used as autonomous or copilots in the triage process of tier 1 SOC analysts. An advantage of using LLM agents for the triage process is that it can help reduce the time and manual effort required by analysts [62]. LLM agents can also improve threat detection by providing reasoning to analysts to make informed decisions [62]. However, as was seen in one of the tests, LLM agents have the potential to hallucinate details based on the prompts provided. It is, therefore, necessary to test the LLM agent's performance using different prompts properly and continuously. Although the recommended implementation is their use as copilots in the security process, this can also lead to

over-reliance on the LLM agent. Over-reliance can lead to missing complex threats that require proper investigation to uncover. Privacy risks are also a serious issue with the use of LLM agents. This research was comfortable with sending the alerts to third-party APIs because they did not contain any real confidential information. This is not the same for organizations who would rather ensure their private data is not transferred to a third party [63]. This suggests that the best way to implement LLM agents in security processes is to have them completely deployed and trained in-house or as part of a suite of products provided by a trusted third party. Furthermore, LLMs are vulnerable to attacks that lead them to reveal internal prompts and certain confidential data [38]. Therefore, integrating LLM agents would require adversary emulation to ensure that the agents do not reveal confidential data.

6 Conclusion

This research set out to design and test a system that used LLM agents to perform the triage process of tier 1 analysts in the SOC. The increasing development of LLMs has provided several research opportunities around their integration into different cybersecurity processes. This integration can help reduce the workload on SOC analysts and help to prevent alert fatigue.

The objective of this thesis focuses on designing a framework that utilizes LLM agents to perform triage on security alerts. The first component of the system is the SIEM which is responsible for generating and processing alerts. The SIEM is the central component in most SOC environments and plays the same role in this framework. The second component is the LLM agent which is accessed through the use of publicly available APIs. This component can also be used with internal APIs, depending on the mode of implementation. The LLM agent is accessed using carefully designed prompts to get the desired output from the agent. The last component is the reporting component which is used to process the output from the LLM agent. The reporting component provides a report using unique identity value of the alert, the decision of the LLM agent, and the reason for the decision.

The thesis also tested the performance of different LLM agents using the proposed system. This involved designing and using an experiment with various stages. The first stage involved preparing the dataset by generating logs and alerts from a Windows, Ubuntu Linux, and macOS endpoint. In this phase, the study utilizes

an adversary emulation tool, CALDERA, to generate logs which trigger ‘interesting’ alerts on the SIEM solution. Regular user activities are used to trigger ‘not-interesting’ and informational alerts on the SIEM solutions. This study selected and labeled 58 alerts which were either unique in the alert description or because of the malicious signature in the alerts.

The next stage of the experiment involved using APIs to prompt 5 LLM models to perform triage on the alerts. Using widely used metrics, the study evaluates the performance of the models in the prepared alert set. The results showed two standout performers among the LLM models tested. GPT-4o identified all interesting alerts in the dataset, showing a 100% recall rate. LLaMA-3 had the highest accuracy among both ‘interesting’ and ‘not-interesting’ alert categories.

Beyond the numerical results, the research also performs qualitative analysis on the responses provided by the LLM agents. A key finding of the research was that advanced LLM models were more likely to produce more verbose responses to the prompts. The level of verbosity in the LLM agent’s response can help with training and tuning the model further. Additionally, the two highest performing models, GPT-4o and LLaMA-3 were tested to confirm how they handled additional alert contexts in the prompt. The experiment showed that GPT-4o handled the contexts provided well enough, and correctly identified two alerts that had the same description but different signatures. LLaMA-3, on the other hand, hallucinated the contexts in the prompts into the alerts provided, which resulted in wrong alert classification. This thesis posits that this hallucination is caused by the alerts being part of the prompts, unlike GPT-4o’s Assistant API that allows uploading the alerts in a different file. The thesis also provides a cost analysis of the LLM agent triage framework, and the implications of this research.

Overall, this study contributes to the growing body of research surrounding the use of LLM agents by employing LLM agents in novel way. The study also

contributes by designing and testing a system that relies on LLM agents to perform the SOC tier 1 analyst triage process.

6.1 Limitations of the Study

This section describes this study's limitations and references to suggested future works (f1, f2, f3, and f4). The next section recommends future work to overcome these limitations.

The first limitation observed is the limited alert dataset used in this thesis (**f1**). Although the alert dataset generated in this thesis contained a sufficient representation of the MITRE ATT&CK framework, the study used a dataset with fewer SIEM alerts than the average number of alerts generated daily by organizations. This is related to the problem identified by Hindy et al. [64] on the lack of current threat detection datasets. To address this issue, a custom alert dataset was generated and used to test the LLM agents. However, the dataset contained few alerts, with only 58 alerts tested. In comparison, in their study, Kersten et al. [7] tested 200 alerts, albeit the study was conducted on human Tier 1 analysts. This thesis's relatively low number of alerts and similar studies also show the need to create and collect alert datasets representative of real-world attack scenarios.

Another notable limitation of this research is the reliance on the general training of the LLMs that were tested (**f2**). As discussed in Chapter 3, LLMs are typically trained on extensive datasets that span various research fields. While this training equips the LLMs with emergent capabilities, it also implies that the LLMs are not specifically trained in any particular field. For instance, in the medical field, training datasets are used to train custom LLMs on specific medical issues and patient history, leading to enhanced performance [65]. However, it's important to note that training custom models entails a significant computational cost, although it is more cost-effective in the long run than relying on third-party APIs.

Furthermore, this research did not delve into how different prompts can influence the performance of LLM agents in the alert triage process (**f3**). As highlighted in Chapter 3, the way a prompt is formulated can impact the performance of an LLM on the same task. This suggests that it's possible to enhance LLM performance through prompt engineering and testing. This research used the same prompt for all the tested LLM agents, incorporating optimization recommendations from the literature analyzed. However, a thorough analysis was not conducted to ascertain how different text in the prompts would fare for different models.

As described in the experiment design, this study used the output of the models after the first execution of the prompts (**f4**). This thesis did not test whether running the same tests repeatedly would affect the model's results. For example, it has been observed that sending the same API request multiple times to OpenAI's GPT-4 can lead to different answers [66]. SOC alerts are usually repetitive, so testing the LLM agents to determine if the responses are consistent regardless of how often a request is sent is necessary.

6.2 Future Work

This section describes the suggested future works for this study.

Future research should involve a dataset with many alerts with varied alert signatures representative of real-world attack scenarios (**f1**). This would include creating and collecting a large dataset of alerts, which would serve future research. A reliable alert dataset would make it easier to benchmark the performance of LLMs in alert triage.

Future research should incorporate training and tuning a custom LLM model specifically for the alert triage process (**f2**). Custom models have shown the ability to perform better than those not trained on the specific dataset being tested [65]. Therefore, training a custom SOC Tier 1 analyst model would involve using a train-

ing set that contains alerts. Furthermore, the custom model can be tested to see if it performs better than the general-purpose models.

Future research should measure how the use of different prompts can affect the performance of the LLM models in the alert triage process (**f3**).

Future research should measure if running multiple tests on the same alerts can change the LLM agent's decisions (**f4**). This research will determine the reliability of LLM agents for the SOC Tier 1 analyst triage process.

References

- [1] C. Zhong, J. Yen, P. Liu, and R. F. Erbacher, “Automate Cybersecurity Data Triage by Leveraging Human Analysts’ Cognitive Process”, in *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, IEEE, 2016, pp. 357–363. DOI: 10.1109/BigDataSecurity-HPSC-IDS.2016.41.
- [2] M. Mutemwa, J. Mtsweni, and L. Zimba, “Integrating a Security Operations Centre with an Organization’s Existing Procedures, Policies and Information Technology Systems”, in *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, IEEE, 2018, pp. 1–6. DOI: 10.1109/ICONIC.2018.8601251.
- [3] M. Kaheh, D. K. Kholgh, and P. Kostakos, *Cyber Sentinel: Exploring Conversational Agents in Streamlining Security Tasks with GPT-4*, 2023. DOI: 10.48550/ARXIV.2309.16422. [Online]. Available: <https://arxiv.org/abs/2309.16422>.
- [4] M. Vielberth, F. Böhm, I. Fichtinger, and G. Pernul, “Security Operations Center: A Systematic Study and Open Challenges”, *IEEE Access*, vol. 8, pp. 227 756–227 779, 2020. DOI: 10.1109/ACCESS.2020.3045514.

-
- [5] B. D. Bryant and H. Saiedian, “Improving siem alert metadata aggregation with a novel kill-chain based classification model”, *Computers amp; Security*, vol. 94, p. 101 817, Jul. 2020, ISSN: 0167-4048. DOI: 10.1016/j.cose.2020.101817. [Online]. Available: <http://dx.doi.org/10.1016/j.cose.2020.101817>.
- [6] K. Sornalakshmi, “Detection of dos attack and zero day threat with siem”, in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE, Jun. 2017. DOI: 10.1109/iccons.2017.8250515. [Online]. Available: <http://dx.doi.org/10.1109/ICCONS.2017.8250515>.
- [7] L. Kersten, T. Mulders, E. Zambon, C. Snijders, and L. Allodi, “‘Give Me Structure’: Synthesis and Evaluation of a (Network) Threat Analysis Process Supporting Tier 1 Investigations in a Security Operation Center”, in *Nineteenth Symposium on Usable Privacy and Security (SOUPS 2023)*, Anaheim, CA: USENIX Association, Aug. 2023, pp. 97–111, ISBN: 978-1-939133-36-6. [Online]. Available: <https://www.usenix.org/conference/soups2023/presentation/kersten>.
- [8] T. Ban, N. Samuel, T. Takahashi, and D. Inoue, “Combat Security Alert Fatigue with AI-Assisted Techniques”, in *Proceedings of the 14th Cyber Security Experimentation and Test Workshop*, ser. CSET ’21, Virtual, CA, USA: Association for Computing Machinery, 2021, pp. 9–16, ISBN: 9781450390651. DOI: 10.1145/3474718.3474723. [Online]. Available: <https://doi.org/10.1145/3474718.3474723>.
- [9] V. H. Perera, A. N. Senarathne, and L. Rupasinghe, “Intelligent SOC Chatbot for Security Operation Center”, in *2019 International Conference on Advancements in Computing (ICAC)*, IEEE, 2019, pp. 340–345. DOI: 10.1109/ICAC49085.2019.9103388.

- [10] F. N. Motlagh, M. Hajizadeh, M. Majd, P. Najafi, F. Cheng, and C. Meinel, *Large Language Models in Cybersecurity: State-of-the-Art*, 2024. DOI: 10.48550/ARXIV.2402.00891. [Online]. Available: <https://arxiv.org/abs/2402.00891>.
- [11] T. Wu, S. He, J. Liu, S. Sun, K. Liu, Q.-L. Han, and Y. Tang, “A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development”, *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 5, pp. 1122–1136, 2023. DOI: 10.1109/JAS.2023.123618. [Online]. Available: <https://www.ieee-jas.net/en/article/doi/10.1109/JAS.2023.123618>.
- [12] S. Zeadally, E. Adi, Z. Baig, and I. A. Khan, “Harnessing Artificial Intelligence Capabilities to Improve Cybersecurity”, *IEEE Access*, vol. 8, pp. 23 817–23 837, 2020. DOI: 10.1109/ACCESS.2020.2968045.
- [13] Gartner Inc. “Gartner Survey Revealed 34% of Organizations Are Already Using or Implementing AI Application Security Tools”. (Sep. 2023), [Online]. Available: <https://www.gartner.com/en/newsroom/press-releases/2023-09-18-gartner-survey-revealed-34-percent-of-organizations-are-already-using-or-implementing-ai-application-security-tools>.
- [14] M. Cubric, “Drivers, barriers and social considerations for AI adoption in business and management: A tertiary study”, *Technology in Society*, vol. 62, p. 101 257, 2020, ISSN: 0160-791X. DOI: <https://doi.org/10.1016/j.techsoc.2020.101257>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0160791X19307171>.
- [15] S. Furnell, “The cybersecurity workforce and skills”, *Computers & Security*, vol. 100, p. 102 080, 2021, ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2020.102080>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167404820303539>.

- [16] 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)*, Ottawa: USENIX Association, Jul. 2015, pp. 347–359, ISBN: 978-1-931971-249. [Online]. Available: <https://www.usenix.org/conference/soups2015/proceedings/presentation/sundaramurthy>.
- [17] A. A. Mughal, “Building and Securing the Modern Security Operations Center (SOC)”, *International Journal of Business Intelligence and Big Data Analytics*, vol. 5, no. 1, pp. 1–15, Jan. 2022. [Online]. Available: <https://research.tensorgate.org/index.php/IJBIBDA/article/view/21>.
- [18] K. Knerler, I. Parker, and C. Zimmerman, *11 Strategies of a World-Class Cybersecurity Operations Center*, Second Edition. The MITRE Corporation, 2022, ISBN: 979-8-9856450-7-1. [Online]. Available: <https://www.mitre.org/sites/default/files/2022-04/11-strategies-of-a-world-class-cybersecurity-operations-center.pdf>.
- [19] K. Demertzis, N. Tziritas, P. Kikiras, S. L. Sanchez, and L. Iliadis, “The Next Generation Cognitive Security Operations Center: Adaptive Analytic Lambda Architecture for Efficient Defense against Adversarial Attacks”, *Big Data and Cognitive Computing*, vol. 3, no. 1, 2019, ISSN: 2504-2289. DOI: 10.3390/bdcc3010006. [Online]. Available: <https://www.mdpi.com/2504-2289/3/1/6>.
- [20] P. Danquah, “Security Operations Center: A Framework for Automated Triage, Containment and Escalation”, *Journal of Information Security*, vol. 11, no. 4, pp. 225–240, 2020, ISSN: 2153-1242. DOI: 10.4236/jis.2020.114015. [Online]. Available: <http://dx.doi.org/10.4236/jis.2020.114015>.

-
- [21] P. Cichonski, T. Millar, T. Grance, and K. Scarfone, *Computer Security Incident Handling Guide: Recommendations of the National Institute of Standards and Technology*. National Institute of Standards and Technology, Aug. 2012. DOI: 10.6028/nist.sp.800-61r2. [Online]. Available: <http://dx.doi.org/10.6028/NIST.SP.800-61r2>.
- [22] C. Onwubiko, “Cyber security operations centre: Security monitoring for protecting business and supporting cyber defense strategy”, in *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, IEEE, Jun. 2015, pp. 1–10. DOI: 10.1109/CyberSA.2015.7166125.
- [23] Picus Labs. “What is Adversary Emulation?” [Accessed 04-05-2024]. (Dec. 2023), [Online]. Available: <https://www.picussecurity.com/resource/glossary/what-is-adversary-emulation>.
- [24] The MITRE Corporation. “MITRE ATT&CK”. [Accessed 04-05-2024]. (2024), [Online]. Available: <https://attack.mitre.org/>.
- [25] M. Rosso, M. Campobasso, G. Gankhuyag, and L. Allodi, “SAIBERSOC: Synthetic Attack Injection to Benchmark and Evaluate the Performance of Security Operation Centers”, in *Proceedings of the 36th Annual Computer Security Applications Conference*, ser. ACSAC '20, Austin, USA: Association for Computing Machinery, 2020, pp. 141–153, ISBN: 9781450388580. DOI: 10.1145/3427228.3427233. [Online]. Available: <https://doi.org/10.1145/3427228.3427233>.
- [26] G. González-Granadillo, S. González-Zarzosa, and R. Diaz, “Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures”, *Sensors*, vol. 21, no. 14, 2021, ISSN: 1424-8220. DOI:

- 10.3390/s21144759. [Online]. Available: <https://www.mdpi.com/1424-8220/21/14/4759>.
- [27] S. Bhatt, P. K. Manadhata, and L. Zomlot, “The Operational Role of Security Information and Event Management Systems”, *IEEE Security & Privacy*, vol. 12, no. 5, pp. 35–41, 2014. DOI: 10.1109/MSP.2014.103.
- [28] M. Vielberth and G. Pernul, “A Security Information and Event Management Pattern”, in *12th Latin American Conference on Pattern Languages of Programs (SLPLoP)*, Universit{a}, Nov. 2018. DOI: 10.5283/EPUB.41139. [Online]. Available: <https://epub.uni-regensburg.de/41139/>.
- [29] M. Cinque, D. Cotroneo, and A. Pecchia, “Challenges and Directions in Security Information and Event Management (SIEM)”, in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, IEEE, Oct. 2018, pp. 95–99. DOI: 10.1109/ISSREW.2018.00-24.
- [30] A. Tariq, J. Manzoor, M. A. Aziz, Z. U. A. Tariq, and A. Masood, “Open source SIEM solutions for an enterprise”, *Information & Computer Security*, vol. 31, no. 1, pp. 88–107, Sep. 2022, ISSN: 2056-4961. DOI: 10.1108/ics-09-2021-0146. [Online]. Available: <http://dx.doi.org/10.1108/ICS-09-2021-0146>.
- [31] Wazuh, Inc. “Getting started with wazuh”. [Accessed 18-05-2024]. (2024), [Online]. Available: <https://documentation.wazuh.com/current/getting-started/index.html>.
- [32] E. Agyepong, Y. Cherdantseva, P. Reinecke, and P. Burnap, “Towards a Framework for Measuring the Performance of a Security Operations Center Analyst”, in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, IEEE, Jun. 2020, pp. 1–8. DOI: 10.1109/CyberSecurity49315.2020.9138872.

- [33] T. Mulders, “Discerning Wheat from Chaff in SOCs: A Model to Identify ‘Non-Interesting’ Events in Security Operation Centers”, M.S. thesis, Eindhoven University of Technology, 2022, 17 pp. [Online]. Available: <https://research.tue.nl/en/studentTheses/31dc0bf1-4bc9-479f-9f6a-b4c3a8aed658>.
- [34] B. Filar, R. Seymour, and M. Park, “Ask Me Anything: A Conversational Interface to Augment Information Security Workers”, in *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, Santa Clara, CA: USENIX Association, Jul. 2017. [Online]. Available: <https://www.usenix.org/conference/soups2017/workshop-program/wsiw2017/filar>.
- [35] J.-h. Li, “Cyber security meets artificial intelligence: a survey”, *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 12, pp. 1462–1474, Dec. 2018, ISSN: 2095-9230. DOI: 10.1631/fitee.1800573. [Online]. Available: <http://dx.doi.org/10.1631/FITEE.1800573>.
- [36] R. Kaur, D. Gabrijelčič, and T. Klobučar, “Artificial intelligence for cybersecurity: Literature review and future research directions”, *Information Fusion*, vol. 97, p. 101804, 2023, ISSN: 1566-2535. DOI: <https://doi.org/10.1016/j.inffus.2023.101804>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1566253523001136>.
- [37] W. X. Zhao *et al.*, *A Survey of Large Language Models*, 2023. DOI: 10.48550/ARXIV.2303.18223. [Online]. Available: <https://arxiv.org/abs/2303.18223>.
- [38] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, “A Survey on Large Language Model (LLM) Security and Privacy: The Good, The Bad, and The Ugly”, *High-Confidence Computing*, vol. 4, no. 2, p. 100211, 2024, ISSN: 2667-2952. DOI: <https://doi.org/10.1016/j.hcc.2024.100211>. [Online].

- Available: <https://www.sciencedirect.com/science/article/pii/S266729522400014X>.
- [39] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, *et al.*, *A Survey on Large Language Models: Applications, Challenges, Limitations, and Practical Usage*, Jul. 2023. DOI: 10.36227/techrxiv.23589741.v4. [Online]. Available: <http://dx.doi.org/10.36227/techrxiv.23589741.v4>.
- [40] OpenAI *et al.*, *GPT-4 Technical Report*, 2024. DOI: 10.48550/ARXIV.2303.08774. [Online]. Available: <https://arxiv.org/abs/2303.08774>.
- [41] Meta. “Meta and Microsoft Introduce the Next Generation of Llama”. [Accessed 13-05-2024]. (Jul. 2023), [Online]. Available: <https://ai.meta.com/blog/llama-2/>.
- [42] Google. “AI ACROSS GOOGLE: PaLM 2”. [Accessed 13-05-2024]. (), [Online]. Available: <https://ai.google/discover/palm2/>.
- [43] S. Arora, A. Narayan, M. F. Chen, L. Orr, N. Guha, K. Bhatia, I. Chami, and C. Re, “Ask me anything: A simple strategy for prompting language models”, in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: <https://openreview.net/forum?id=bhUPJnS2g0X>.
- [44] Y. Chang *et al.*, “A Survey on Evaluation of Large Language Models”, *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, Mar. 2024, ISSN: 2157-6904. DOI: 10.1145/3641289. [Online]. Available: <https://doi.org/10.1145/3641289>.
- [45] C. Qin, A. Zhang, Z. Zhang, J. Chen, M. Yasunaga, and D. Yang, *Is ChatGPT a General-Purpose Natural Language Processing Task Solver?*, 2023. DOI: 10.48550/ARXIV.2302.06476. [Online]. Available: <https://arxiv.org/abs/2302.06476>.

- [46] OpenAI. “Assistants API”. [Accessed 18-05-2024]. (2024), [Online]. Available: <https://platform.openai.com/docs/assistants/overview>.
- [47] R. Fang, R. Bindu, A. Gupta, and D. Kang, *LLM Agents can Autonomously Exploit One-day Vulnerabilities*, 2024. DOI: 10.48550/ARXIV.2404.08144. [Online]. Available: <https://arxiv.org/abs/2404.08144>.
- [48] E. Karlsen, R. Copstein, X. Luo, J. Schwartzentruber, B. Niblett, A. Johnston, M. I. Heywood, and N. Zincir-Heywood, “Exploring semantic vs. Syntactic features for unsupervised learning on application log files”, in *2023 7th Cyber Security in Networking Conference (CSNet)*, IEEE, 2023, pp. 219–225. DOI: 10.1109/CSNet59123.2023.10339765.
- [49] M. Omar and S. Shiaeles, “VulDetect: A novel technique for detecting software vulnerabilities using Language Models”, in *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, IEEE, Jul. 2023, pp. 105–110. DOI: 10.1109/CSR57506.2023.10224924.
- [50] P. Ranade, A. Piplai, A. Joshi, and T. Finin, “CyBERT: Contextualized embeddings for the cybersecurity domain”, in *2021 IEEE International Conference on Big Data (Big Data)*, IEEE, 2021, pp. 3334–3342. DOI: 10.1109/BigData52589.2021.9671824.
- [51] T. Koide, N. Fukushi, H. Nakano, and D. Chiba, *ChatSpamDetector: Leveraging Large Language Models for Effective Phishing Email Detection*, 2024. DOI: 10.48550/ARXIV.2402.18093. [Online]. Available: <https://arxiv.org/abs/2402.18093>.
- [52] V. M. A. de Lima, J. R. Barbosa, and R. M. Marcacini, “Learning Risk Factors from App Reviews: A Large Language Model Approach for Risk Matrix Construction”, Jul. 2023. DOI: 10.21203/rs.3.rs-3182322/v1. [Online]. Available: <http://dx.doi.org/10.21203/rs.3.rs-3182322/v1>.

-
- [53] Y. Charalambous, N. Tihanyi, R. Jain, Y. Sun, M. A. Ferrag, and L. C. Cordeiro, *A New Era in Software Security: Towards Self-Healing Software via Large Language Models and Formal Verification*, 2023. DOI: 10.48550/ARXIV.2305.14752. [Online]. Available: <https://arxiv.org/abs/2305.14752>.
- [54] P. V. Falade, “Decoding the Threat Landscape: ChatGPT, FraudGPT, and WormGPT in Social Engineering Attacks”, *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, Oct. 2023, ISSN: 2456-3307. DOI: 10.32628/cseit2390533. [Online]. Available: <http://dx.doi.org/10.32628/CSEIT2390533>.
- [55] P. Chao, A. Robey, E. Dobriban, H. Hassani, G. J. Pappas, and E. Wong, *Jailbreaking black box large language models in twenty queries*, 2023. DOI: 10.48550/ARXIV.2310.08419. [Online]. Available: <https://arxiv.org/abs/2310.08419>.
- [56] Statcounter. “Desktop Operating System Market Share Worldwide - May 2024”. [Accessed 18-05-2024]. (2024), [Online]. Available: <https://gs.statcounter.com/os-market-share/desktop/worldwide>.
- [57] W.-L. Chiang *et al.*, *Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference*, 2024. DOI: 10.48550/ARXIV.2403.04132. [Online]. Available: <https://arxiv.org/abs/2403.04132>.
- [58] OpenAI, *Hello GPT-4o*, <https://openai.com/index/hello-gpt-4o/>, [Accessed 12-06-2024], 2024.
- [59] AI@Meta, “Llama 3 Model Card”, 2024. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.
- [60] Mistral AI, *Cheaper, Better, Faster, Stronger| Mistral AI | Frontier AI in your hands*, [Accessed 28-05-2024], Apr. 2024. [Online]. Available: <https://mistral.ai/news/mixtral-8x22b/>.

- [61] A. Pecchia, D. Cotroneo, R. Ganesan, and S. Sarkar, “Filtering Security Alerts for the Analysis of a Production SaaS Cloud”, in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, IEEE, Dec. 2014, pp. 233–241. DOI: 10.1109/UCC.2014.32. [Online]. Available: <http://dx.doi.org/10.1109/UCC.2014.32>.
- [62] D. M. Divakaran and S. T. Peddinti, *LLMs for Cyber Security: New Opportunities*, 2024. DOI: 10.48550/ARXIV.2404.11338. [Online]. Available: <https://arxiv.org/abs/2404.11338>.
- [63] W. L. Dang. “LLMs in security: What are the opportunities for startups?” [Accessed 28-05-2024], Unusual Ventures. (Sep. 2023), [Online]. Available: <https://www.unusual.vc/post/llms-security>.
- [64] H. Hindy, D. Brosset, E. Bayne, A. K. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, “A taxonomy of network threats and the effect of current datasets on intrusion detection systems”, *IEEE Access*, vol. 8, pp. 104 650–104 675, 2020, ISSN: 2169-3536. DOI: 10.1109/access.2020.3000179. [Online]. Available: <http://dx.doi.org/10.1109/ACCESS.2020.3000179>.
- [65] N. H. Shah, D. Entwistle, and M. A. Pfeffer, “Creation and adoption of large language models in medicine”, *JAMA*, vol. 330, no. 9, p. 866, Sep. 2023, ISSN: 0098-7484. DOI: 10.1001/jama.2023.14217. [Online]. Available: <http://dx.doi.org/10.1001/jama.2023.14217>.
- [66] M. Lumiste. “Why you should A/B test your LLM”. [Accessed 17-06-2024]. (Aug. 2023), [Online]. Available: <https://mlumiste.com/technical/ab-test-llm-evals/>.

Appendix A Sample Wazuh Alert

```
1 {
2   "_index": "wazuh-alerts-4.x-2024.05.23",
3   "_id": "FF0CpY8BpwWNZa7JjFdd",
4   "_version": 1,
5   "_score": null,
6   "_source": {
7     "input": {
8       "type": "log"
9     },
10    "agent": {
11      "name": "DESKTOP-I1PBMQ1",
12      "id": "002"
13    },
14    "manager": {
15      "name": "wazuh-server"
16    },
17    "data": {
18      "extra_data": "DESKTOP-I1PBMQ1->any"
19    },
20    "rule": {
21      "firedtimes": 1,
22      "mail": false,
23      "level": 3,
24      "pci_dss": [
25        "10.6.1"
```

```
26     ],
27     "hipaa": [
28         "164.312.b"
29     ],
30     "tsc": [
31         "CC7.2",
32         "CC7.3"
33     ],
34     "description": "New wazuh agent connected.",
35     "groups": [
36         "ossec"
37     ],
38     "id": "501",
39     "nist_800_53": [
40         "AU.6"
41     ],
42     "gpg13": [
43         "10.1"
44     ],
45     "gdpr": [
46         "IV_35.7.d"
47     ]
48 },
49 "location": "wazuh-agent",
50 "decoder": {
51     "parent": "ossec",
52     "name": "ossec"
53 },
54 "id": "1716460351.2348",
55 "full_log": "ossec: Agent started: 'DESKTOP-I1PBMQ1->any'.",
56 "timestamp": "2024-05-23T10:32:31.391+0000"
57 },
58 "fields": {
```

```
59     "timestamp": [  
60         "2024-05-23T10:32:31.391Z"  
61     ]  
62 },  
63 "highlight": {  
64     "agent.id": [  
65         "@opensearch-dashboards-highlighted-field@002@/opensearch -  
66         dashboards-highlighted-field@"  
67     ],  
68     "manager.name": [  
69         "@opensearch-dashboards-highlighted-field@wazuh-server@/  
70         opensearch-dashboards-highlighted-field@"  
71     ]  
72 },  
73 "sort": [  
74     1716460351391  
75 ]  
76 }
```

Snippet A.1: Sample Wazuh alert