Expert Systems With Applications 331 (2026) 133194

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Possibilities and limitations of using large language models (LLMs) for alert classification and prioritisation in security operations centers (SOCs)

Matthias Rieger[a], Atif Shah[a], Abu Alam[a,*], Md. Jakir Hossain[b]

[a] University of Gloucestershire, The Park, Cheltenham, GL50 2RH, Gloucestershire, United Kingdom
[b] University of Chester, Parkgate Road, Chester, CH1 4BJ, Cheshire, United Kingdom

ARTICLE INFO

Keywords:
Security operations center
SOC
Large language models
LLM
Alert triage
Alert prioritisation
Alert classification
Incident response
LLM aided triage
Threat detection

ABSTRACT

As cyber threats have become more sophisticated over time, security opera- tions centers (SOCs) have increasingly faced vast amounts of security alerts to be investigated, ultimately leading to overwhelmed security analysts and symptoms such as alert fatigue. While traditional automation has helped with streamlining parts of the incident response workflow, it remains limited, especially in regard to context-dependant tasks such as triage and prioriti- sation. Against this background, this research investigates the potential of large language models (LLMs) to augment SOC workflows through natural language understanding. Using a dataset of 178 manually labeled alerts, eight general-purpose LLMs from OpenAI, DeepSeek and Ai2 were tasked with independently classifying the alerts into true and false positives as well as prioritising them as low, medium, high or critical. In addition, traditional supervised machine learning baselines, including Logistic Regression, Random Forest and Linear Support Vector Machine (SVM), were implemented for comparative evaluation on the binary classification task. The performance of the models was assessed using standard evaluation metrics such as accuracy, precision, recall, F1-score and false positive rates as well as operational factors like runtime and cost per alert. Results show that while several LLMs achieved strong classification recall, lightweight machine learning models achieved competitive and, in some cases, superior binary classification performance, with the Linear SVM baseline achieving the highest overall F1-score. However, alert prioritisation proved substantially more challenging across all evaluated LLMs. While some models captured high- severity alerts with strong recall, precision remained consistently low, contributing to significant alert noise and elevated false positive rates. These findings suggest that while LLMs are able to support SOC analysts with ini- tial triage and contextual reasoning, their reliability for accurate prioritisation remains limited, and lightweight machine learning approaches continue to provide strong practical value for structured SOC alert classification tasks.

1. Introduction

With cyber security increasingly having gained importance over the years due to an ever evolving threat landscape, many enterprises have established security operations centers (SOCs) for the sake of organisational defence (Ismail et al., 2025; Knerler et al., 2022; Singh et al., 2024). At their heart, SOCs are constantly monitoring an organisation's Information Technology (IT) landscape for intrusions as well as analysing and responding to emerging security alerts (Knerler et al., 2022; Singh et al., 2024). The first task carried out by SOC analysts in this context is referred to as alert triage, which is the process of evaluating incoming alerts as well as determining their priority (Knerler et al., 2022; Vielberth et al., 2020). However, because of IT environments and networks steadily getting more complex by integrating new systems as well as defensive mechanisms, the number of security alerts - often generated by security information and event management (SIEM) systems - SOCs face on a daily basis, has been increasing ever since (Baruwal Chhetri et al., 2024; Jalalvand et al., 2024). As a result, SOC analysts suffer from alert fatigue, which means that they are constantly overwhelmed by the sheer amount of alerts, which in turn leads to delayed responses, overlooked incidents and psychological strain (Baruwal Chhetri et al., 2024; Sundaramurthy et al., 2015; Tariq et al., 2025).

In search of a solution to this problem, one common recommendation found in literature is to utilise automation in various forms, which often times includes using machine learning (ML) (Baruwal Chhetri et al., 2024; Tilbury & Flowerday, 2024b). At the same time, the issues of ML-based approaches such as the reliance on training data - especially in case the data is biased or incomplete - as well as the necessity of regular retraining, are pointed out (Jalalvand et al., 2024; Khayat et al.,

* Corresponding author.
E-mail addresses: matth.rieger@gmail.com (M. Rieger), ashah25@glos.ac.uk (A. Shah), aalam@glos.ac.uk (A. Alam), m.hossain@chester.ac.uk (Md.J. Hossain).

https://doi.org/10.1016/j.eswa.2026.133194
Received 3 December 2025; Received in revised form 13 May 2026; Accepted 5 June 2026
Available online 15 June 2026

2025). Therefore, while automation is most commonly used for countering alert fatigue (Tariq et al., 2025), several limitations remain that leave an appropriate gap. One potentially promising direction literature points to for filling this gap, is the use of large language models (LLMs) for AI-assisted triage (Baruwal Chhetri et al., 2024; Ismail et al., 2025; Jalalvand et al., 2024; Tariq et al., 2025). In fact, several industry solutions such as Microsoft's security copilot or CrowdStrike's charlotte AI already exist that aim at supporting SOC analysts in a variety of ways, automatically performing tasks such as triage, investigation and remediation (Tariq et al., 2025; Tilbury & Flowerday, 2024b).

However, despite the fact that a myriad of industry solutions already exist that serve the purpose of supporting SOC analysts with their numerous tasks such as triage and investigation, research in this field remains meaningful, as existing solutions mostly act as a black box without providing any detailed insights into how they work. Furthermore, they are often provided as holistic all-in-one solutions that do not solely focus on alert classification and prioritisation. Additionally, only limited data on their effectiveness exists.

In regard to academic literature, many papers point to LLMs as promising solutions to enhance a SOC's efficiency and help with issues such as alert fatigue but only occasionally evaluate them in real-world SOC workflows. Besides that, only few works, such as Oniagbi (2024), evaluate the effectiveness of LLM-based classification of security alerts into true and false positive. Furthermore, there is only limited discussion on the LLM-based prioritisation of respective alarms.

1.1. Research questions

Based on the problem statement given, this research aims at answering the following research questions (RQs):

- RQ1: How can LLMs such as ChatGPT be leveraged to perform classification and prioritisation of security alerts within a SOC context?
- RQ2: How effective can LLMs such as ChatGPT classify security alerts as likely true or false positive?
- RQ3: What is the potential of LLMs such as ChatGPT in regard to assigning meaningful priority levels to security alerts?
- RQ4: What are the challenges and limitations of using LLMs like ChatGPT for alert classification and prioritisation in a SOC context?

1.2. Research objectives

Based on the previously given research questions, the primary research objectives (ROs) of this research are as follows, referring to one research question each:

- RO1: To explore how LLMs such as ChatGPT can be utilised for the classification and prioritisation of security alerts within SOC environments
- RO2: To evaluate the LLMs' accuracy in distinguishing between true and false positive security alerts
- RO3: To assess the capability of LLMs to meaningful prioritise security alerts in context
- RO4: To analyse the challenges and limitations associated with implementing LLMs for alert classification and prioritisation in SOC environments

2. Background

2.1. Security operations centers and alert triage

A SOC can be defined as "a team, primarily composed of cybersecurity specialists, organized to prevent, detect, analyze, respond to, and report on cybersecurity incidents." (Knerler et al., 2022). While in this paper the term SOC is being used, it is important to mention that there is no commonly agreed-upon definition for it in literature, with several synonyms existing such as Computer Security Incident Response Team (CSIRT) or Computer Emergency Response Team (CERT) among others (Knerler et al., 2022; Vielberth et al., 2020). In order to achieve its mission, a SOC comprises a mixture of people, processes, technologies as well as governance and compliance (Vielberth et al., 2020).

Technology-wise, at its heart, a SOC heavily relies on using a SIEM solution that is mainly responsible for collecting security-relevant data in a centralised fashion as well as for raising respective alerts (Vielberth et al., 2020). With regard to the people component, a typical SOC is split up into tiers 1 till 3 with respective responsibilities (Vielberth et al., 2020). The primary task of tier 1 analysts is initial alert triage, which encompasses confirming, determining or adjusting the criticality of alerts and putting them into context, i.e., for every alert, tier 1 analysts are required to determine whether it is justified or a false positive and to make sure it is treated according to its priority (Vielberth et al., 2020). Tier 2 analysts, on the other hand, conduct in-depth investigations of alerts escalated to them by tier 1, incorporating threat intelligence (TI) such as indicators of compromise (IoCs) (Vielberth et al., 2020). Lastly, tier 3 analysts primarily focus on proactively discovering possible threats in the environment such as unknown vulnerabilities and are escalated to in case tier 2 analysts struggle with the identification or mitigation of an attack (Vielberth et al., 2020).

As the alert triage phase forms the initial step in the whole incident response workflow, it is of fundamental importance (Baruwal Chhetri et al., 2024; Khayat et al., 2025). Yet, it is this exact phase that is often characterised by an enormous amount of security alerts to be prioritised that overwhelms SOC analysts, makes their work repetitive and exhausting and ultimately leads to a phenomenon referred to as alert fatigue (Baruwal Chhetri et al., 2024; Tariq et al., 2025).

2.2. The problem of alert fatigue

When it comes to the challenges SOCs are commonly facing, there seems to be strong agreement all throughout academic literature. The top most mentioned problem by far is the phenomenon of alert fatigue which security analysts inside SOCs face (Baruwal Chhetri et al., 2024; Ismail et al., 2025; Jalalvand et al., 2024; Khayat et al., 2025; Tilbury & Flowerday, 2024b). This is being supplemented by industry surveys such as Splunk (2025), in which respondents state that they suffer from alert overload for example.

Against this background, alert fatigue can generally be defined as a state of security analysts being overwhelmed mainly by the number of security alerts to triage, which might lead to them overlooking critical alerts due to weariness (Baruwal Chhetri et al., 2024; Khayat et al., 2025). More precisely, it is especially the repetitive nature of a SOC analyst's tasks combined with a constant information overload that lead to desensitisation regarding alarms and a reduced response accuracy, which are all alert fatigue implying symptoms (Tilbury & Flowerday, 2024b).

The reasons that constitute to alert fatigue are manifold and can be divided into people and process-related as well as technological reasons, according to Baruwal Chhetri et al. (2024). One people-related reason that is being worsen by a general shortage of skilled personnel in the cyber security industry, is understaffing of SOCs, which by nature increases the workload and thus leads to more pressure on the respective team members (Baruwal Chhetri et al., 2024; Ismail et al., 2025; Khayat et al., 2025; Tilbury & Flowerday, 2024b). Besides that, one process-related contributor to alert fatigue are poor automation practices, as well as the absence of automation in general (Baruwal Chhetri et al., 2024; Ismail et al., 2025; Khayat et al., 2025). Focusing on technologies, detection rules are often times configured as overly sensitive, resulting in a significant amount of false positives being raised, which is creating a lot of noise among security alerts, diverting analyst resources away from other important alerts as well as increasing the risk of genuine alerts being overlooked (Baruwal Chhetri et al., 2024; Jalalvand et al., 2024).

2.3. Limitations of traditional automation in SOCs

Traditional automation in SOC contexts can take on several forms and be performed to varying degrees (Tilbury & Flowerday, 2024b). General application areas of automation include automated cyber threat intelligence (CTI), incident detection as well as incident response among others (Tilbury & Flowerday, 2024b). Technology-wise, security orchestration, automation, and response (SOAR) platforms often times serve as an enabler for the automation of security processes, facilitating efficient incident response workflows (Baruwal Chhetri et al., 2024).

Besides that, some research has taken the automation of the triage process into focus (Jalalvand et al., 2024; Tilbury & Flowerday, 2024a). In this context, ML-based solutions have most commonly been explored, but other methods such as game theory or graph-based approaches are also used in rare instances (Jalalvand et al., 2024). Focusing on the ML-based approaches, these encompass numerous techniques like supervised and unsupervised learning as well as reinforcement learning, which can all be further divided into a myriad of subtechniques such as k-means clustering and isolation forests among others (Jalalvand et al., 2024). In a SOC context, these methods might be integrated with SIEM solutions (Khayat et al., 2025).

One outstanding advantage of the ML-based approaches is their ability to analyse and process large amounts of alert data (Jalalvand et al., 2024). They enable classifying and prioritising alerts based on specific features and anomaly scores as well as uncovering patterns within alert data (Jalalvand et al., 2024). However, ML-based methods are highly dependant on good quality, ideally labelled training data, which often times is challenging to obtain in real-world environments (Jalalvand et al., 2024; Khayat et al., 2025). Furthermore, the ML models necessitate constant maintenance, such as frequent retraining and other model adjustments, in order to perform appropriately in a dynamic SOC environment (Jalalvand et al., 2024; Tariq et al., 2025). In addition to that, many ML models operate in a black box fashion, lacking appropriate transparency in decision-making, which hinders the analysts' trust (Khayat et al., 2025). Even though explainable AI (XAI) techniques exist that aim at more transparency, their practical applicability in real-world SOC settings stays limited due to the complexity of these methods (Khayat et al., 2025). Besides that, ML-based approaches often times lack validation in real-world scenarios (Khayat et al., 2025).

Looking beyond academic literature, despite some industry solutions existing in this domain, these tools also struggle with challenges similar to the ones previously mentioned, such as reliance on high quality input data as well as the necessity of frequent model retraining (Tariq et al., 2025).

2.4. Large language models and their emerging role in SOC environments

Approximately in the last three years, LLMs have increasingly gained popularity in both industry and academia due to their rapid development and unprecedented performance that opened up new possibilities in various fields, including cyber security (Chang et al., 2024; Zhang et al., 2025). Society's sparked interest in LLMs is being reflected and proven by the astonishing amount of research papers that have been published especially since the launch of ChatGPT on November 30, 2022 (Chang et al., 2024; OpenAI, 2022; Zhao et al., 2025), which represents the most important milestone in the development of LLMs (Minaee et al., 2025). ChatGPT is a chatbot that, while demonstrating an amazing conversation ability, can also perform tasks such as question answering, information seeking as well as text summarisation among others (Minaee et al., 2025; Zhao et al., 2025).

Overall, the rapid pace of development in this field of LLMs is ongoing, which is being reflected by the fact that new findings, techniques and models are being published within a few months or even weeks (Minaee et al., 2025). Besides that, due to their strong performance, there is already talk of LLMs becoming a building block of artificial general intelligence (AGI) - a type of AI that matches or exceeds human capabilities (Minaee et al., 2025; OpenAI, 2023; Zhao et al., 2025).

Definition-wise, in essence, "LLMs are large-scale, pre-trained, statistical language models based on neural networks" (Minaee et al., 2025). To be more precise, LLMs are primarily referred to as transformer-based neural language models (NLMs) with tens to hundreds of billions of parameters, which are pre-trained on massive text corpora, i.e. datasets that include internet knowledge such as Wikipedia as well as books among others (Minaee et al., 2025). Decades of research and development of language models (LMs) with four major development stages - from statistical language models (SLMs) to NLMs to pre-trained language models (PLMs) to LLMs - have led to the recent advances in this field (Minaee et al., 2025; Zhao et al., 2025).

Roughly speaking, LLMs can be divided into open source, open weight and closed source models (Zhang et al., 2025). For open source LLMs such as Ai2's Olmo 3, full transparency is provided as training data, training code and model weights are provided to the public (Olmo, 2026). In regard to open weight LLMs, only model weights are published which allows them to be adjusted appropriately for the sake of research for example (Zhang et al., 2025). Closed source LLMs, on the contrary, are maintained by commercial entities that commonly only offer limited access to their models such as through an application programming interface (API) (Minaee et al., 2025; Zhang et al., 2025). When compared to each other, open source and open weight models, besides being customizable, may lack the scale and performance of closed source LLMs, while closed source models offer state-of-the-art performance, but lack appropriate transparency (Zhang et al., 2025).

Popular LLM families include the closed source generative pre-trained transformer (GPT) family developed by OpenAI, Meta's open weight large language model meta AI (LlaMA) family as well as Google's pathways language model (PaLM) family (Minaee et al., 2025). Besides these popular ones, there is also numerous other LLMs that do not belong to the aforementioned families, yet they have significantly contributed to the developments in the field of LLMs (Minaee et al., 2025).

The usage of an LLM mostly happens through so-called prompts. These are "the textual input provided by users to guide the model's output" (Minaee et al., 2025). Prompts usually include questions, descriptions or specific instructions. The significance of prompts is being reflected by the fact that prompt engineering has turned into its own rapidly evolving discipline, with several techniques being discussed in industry and academia (Minaee et al., 2025; Zhao et al., 2025).

With regard to their capabilities, LLMs can perform a wide range of natural language understanding (NLU) and generation tasks (Chang et al., 2024; Minaee et al., 2025) as depicted in Fig. 1.

More precisely, their basic capabilities encompass world knowledge in the form of question answering, comprehension, which is demonstrated in particular by their ability to summarise content, multilingual capabilities such as translation as well as code-generation capabilities (Minaee et al., 2025). Besides these, especially LLMs' emergent abilities, which smaller-scale language models do not possess, stand out. Among them are in-context learning, instruction following and multi-step reasoning.

The former refers to the LLMs' ability of learning a new task without prior training, based on just a few examples provided within a prompt during inference (Minaee et al., 2025). Instruction following, on the other hand, means that LLMs can adapt instructions given to new types of tasks without examples being provided (Minaee et al., 2025). Lastly, multi-step reasoning describes that LLMs are able to solve complex tasks by decomposing them into smaller logical steps, which is often associated with the use of chain-of-thought prompting where the respective model explicitly presents its intermediate reasoning process (Minaee et al., 2025). LLMs' reasoning capabilities are commonly categorized into common sense, arithmetic, logical and domain-specific reasoning (Chang et al., 2024; Minaee et al., 2025).

It is important to mention that among all of the LLMs' abilities, some are weaker and some are stronger. While performing strong in NLU tasks

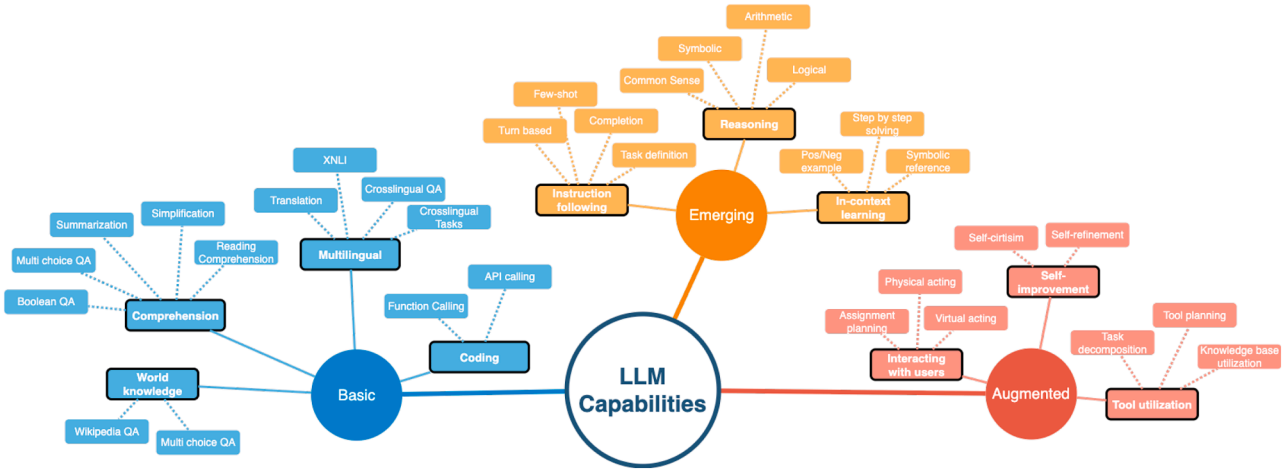

Fig. 1. LLM capabilities (Minaee et al., 2025).

such as sentiment analysis and text classification for example, they perform poorly in natural language inference (NLI), which is the act of determining the logical relationship between a premise and a hypothesis (Chang et al., 2024). Furthermore, their performance in translation tasks is currently satisfactory and despite their reasoning capabilities being promising, more research and optimisation are required due to many limitations.

However, besides showcasing an overall unprecedented performance in their abilities, LLMs still face several challenges, so-called hallucinations above all (Chang et al., 2024; Minaee et al., 2025; OpenAI, 2024; Zhao et al., 2025). Hallucination is defined as the undesired phenomenon of models generating content which is "nonsensical or unfaithful to the provided source content" (Ji et al., 2023) - Put simply, LLMs can produce answers which may sound plausible but are in fact untruthful (Minaee et al., 2025). Mitigating hallucinations remains a persistent challenge that requires tailored strategies and even involvement of human judgement.

Against this background, LLMs have also attracted attention for being used in the cyber security domain. Their wide-ranging applications include threat hunting, threat intelligence as well as vulnerability, malware and anomaly detection among others (Sai et al., 2024; Zhang et al., 2025). Turning the focus to SOC environments, assisting analysts with an LLM-powered solution that is supposed to augment their capabilities has emerged as a promising avenue (Ismail et al., 2025; Tariq et al., 2025).

3. Methodology

This research was conducted based on the research methodology depicted in Fig. 2 below.

The first step, *Alert Generation & Collection*, was comprised of generating and collecting true positive (TP) and false positive (FP) alerts using a local lab environment. In the next stage, *Alert Processing*, this collection of alerts was preprocessed and given to several LLMs for classification and prioritisation. The results received from the LLMs were then saved and postprocessed for evaluation reasons. In the last phase, *Result Evaluation*, the actual evaluation and comparison of the different models' performance took place. Each of the stages is described in detail in the following. All scripts referenced are available at (Rieger, 2025).

Moreover this research is grounded in several critical methodological assumptions that ensure experimental rigor. First, we assume that isolated alert analysis is a valid proxy for evaluating the intrinsic semantic reasoning capabilities of an LLM. By stripping away external context, we isolate the model's ability to interpret raw security telemetry independently. Second, we treat each alert as an independent and identically distributed (i.i.d.) sample, which justifies the use of random data shuffling to eliminate ordering effects. This ensures that the model cannot identify sequential patterns, such as a predictable stream of true positives, which would otherwise bias the quantitative results. Finally, we assume that high fidelity simulation via the Atomic Red Team framework mapped strictly to the MITRE ATT&CK knowledge base is technically representative of real world adversarial techniques, providing a controlled yet authentic dataset for evaluation.

3.1. Alert generation & collection

3.1.1. Lab architecture

The local lab environment used for generating alerts is illustrated in Fig. 3 and explained in detail afterwards.

Using *VirtualBox* as a hypervisor-solution, an isolated network with six virtual machines (VMs) serving different purposes was created. All machines were allowed to communicate with each other as well as to reach the internet.

One VM hosted the open source threat detection and incident response platform *Wazuh*, providing SIEM as well as extended detection and response (XDR) capabilities. Another VM acted as a network-based intrusion detection system (NIDS), running the open source network analysis and threat detection software *Suricata*. It was configured to monitor and inspect traffic on the isolated network and its logs were forwarded to the Wazuh instance. A third VM running Ubuntu served as an attacking target, being vulnerable on purpose. Vulnerabilities on this machine were introduced by hosting intentionally insecure web applications - namely damn vulnerable web application (DVWA) and Mutillidae. A fourth VM running Windows Server 2019 Standard was configured to function as a domain controller (DC), hosting Microsoft's active directory (AD) service, which provides centralized user and computer management for organisations. A fifth VM ran Windows 11 Pro that was joined to the aforementioned DC environment, representing a client computer in this fictitious setting. It was used for simulating usual endpoint behaviour as well as for triggering security alerts using the Invoke-Atomic framework, which helps executing security tests that are intended to simulate adversarial activity. The sixth VM represented a typical attacker's machine running Kali Linux - a linux distribution specifically designed for penetration testing and security auditing. The machine was used to carry out attacks with Kali's built-in tools like the Metasploit framework or hydra for brute-force attacks.

In sum, this architecture allowed for controlled experimentation with both benign and malicious activity as well as for collection of host and network-based alerts, ultimately leading to a dataset of TP and FP alerts that was used subsequently for the LLMs' classification and prioritisation tasks.The methodology for generating each type of alert is detailed in the following.

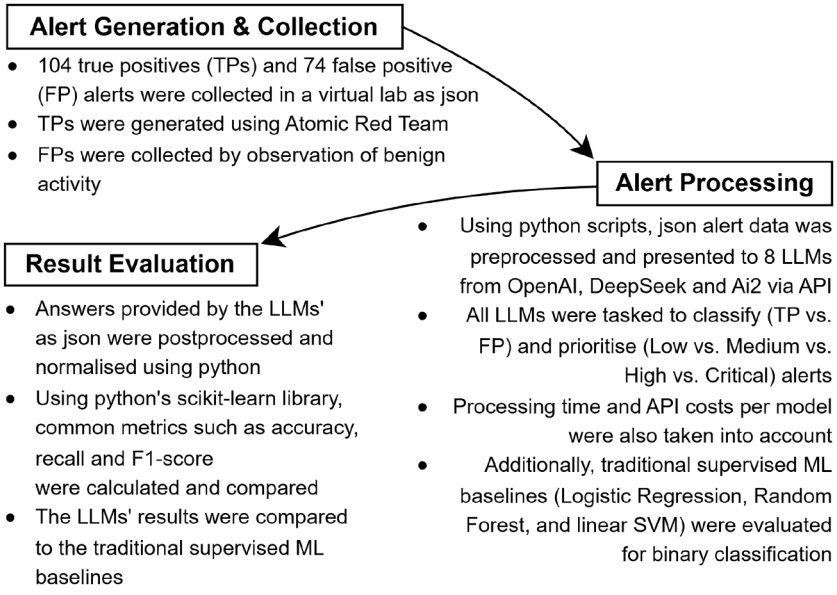

Fig. 2. Research methodology.

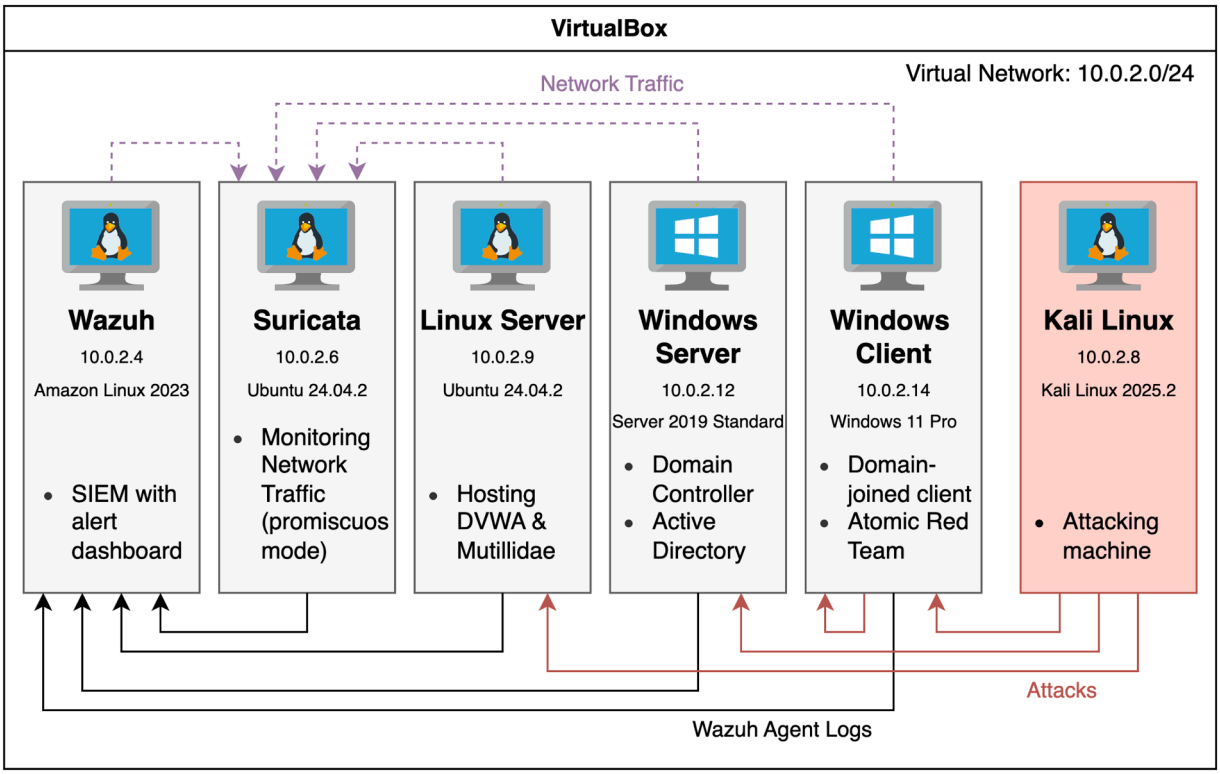

Fig. 3. System overview of the simulated environment.

3.1.2. False positive alert generation

FPs weregenerated and collected immediately after setting up the lab environment, knowing that the systems were free of malicious activity at that point in time. Despite none of the activities occurred did correspond to actual threats in this phase, some overly sensitive out-of-the-box detection rules still triggered. However, the environment's initial cleanliness allowed to confidently assume that all of the triggered alerts resulted from non-malicious activities. Thus, in essence, the gathering of FPs took place through extended observation of the whole system under normal operation over the course of several days. FP alerts triggered by typical, non-malicious background activity and originating from Wazuh agents across all monitored machines were continuously ingested into the SIEM. Based off this stream of alerts, a series of samples were selected and saved appropriately as described in Section 3.1.4. Additionally, specific benign but suspicious-looking actions, which are summarised in Table 1, were performed on purpose to generate FP alerts.

These scenarios were designed to mimic legitimate user or administrative behavior that may be misclassified as malicious in a SOC surrounding. The resulting alerts triggered by these actions were also saved respectively.

Table 1
Additional FP scenarios.

Actions	Fictitious Scenario
Multiple failed SSH login attempts	Administrator misses password three times nnnn
Downloading the EICAR test file	Administrator testing detection mechanisms using the EICAR test file
Execution of a PowerShell script called "mouse mover"	User executes a script to keep the screen unlocked without having to move the mouse manually. The script includes base64-encoded commands to hide the script content

Table 2
Tests executed for TP alert generation.

MITRE Technique	Short Description
T1059.003	Windows Command Shell
T1059.001	PowerShell
T1569.002	Service Execution
T1047	Windows Management Instrumentation (WMI)
T1218.005	Mshta
T1105	Ingress Tool Transfer

3.1.3. True positive alert generation

TPs were generated primarily through adversary simulation using Atomic Red Team - a popular library of security tests intended to simulate adversarial techniques that is developed and maintained by Red Canary. The tests included are mapped to the MITRE ATT&CK framework, which is "a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations." (MITRE , n.d.). Thus, using Atomic Red Team, one is able to emulate specific MITRE ATT&CK techniques, which was done in this case. With the help of the Invoke-Atomic PowerShell module, a series of tests taken from the aforementioned library were executed on the Windows 11 Pro-client in order to simulate adversarial behavior and trigger corresponding detections in Wazuh. The selection of tests to execute was guided by Red Canary's recent threat detection report (Canary, 2025a) that mentions the top 10 most prevalent techniques observed in attacks carried out against their customer base in 2024 (Canary, 2025b). The actual techniques simulated with the aim of generating a series of TP alerts are listed in Table 2.

In addition to that, in order to reflect real-world attacker bahavior, efforts were made to simulate techniques associated with the top 10 threats identified in the aforementioned report (Canary, 2025c) such as SOCGolish, Impacket, Scarlet Goldfinch and Mimikatz. Besides running automated tests using Atomic Red Team, manual attacks were carried out in the lab environment using the Kali machine. For this, the tools preinstalled on the VM were utilized and Wazuh (n.d.) was used for guidance where applicable - specifically in regard to necessary detection rule adjustments. The attacks performed include brute-force login attempts using hydra, running unauthorized processes such as Netcat, conducting network reconnaissance in the form of port scanning with Nmap, the execution of a ShellShock exploit as well as the usage of the diamorphine rootkit on the linux server VM.

3.1.4. Alert collection process

While and after raising TP as well as FP alerts, a subset of them was collected in the manner depicted in Fig. 4.

As shown, the Wazuh dashboard was used to filter for relevant alerts first. Once identified, each of the respective alerts was extracted in JavaScript Object Notation (JSON) format. The exported JSON was then saved to a JSON lines (JSONL) file containing one JSON object per line. Throughout the process, two separate JSONL files were maintained - one for collecting TP alerts and another one for collecting FP alerts. During experimentation, a total of 65,567 alerts were generated - many of which were duplicates or irrelevant in this context. The dataset's inherent class imbalance skewing significantly toward low priority alerts

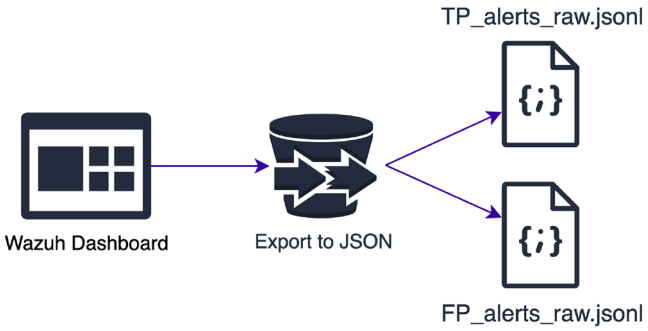

Fig. 4. Alert collection process.

Table 3
Dataset priority distribution.

Priority	Count and Percentage
Low	136 alerts (76.4 %)
Medium	12 alerts (6.7 %)
High	25 alerts (14.0 %)
Critical	5 alerts (2.8 %)

(76.4 %) and false positives (41.6 %) - is a deliberate design choice intended to mirror the operational reality of modern SOC environments . In real world settings, analysts routinely face a high volume of noise from low priority alerts and false positives, while critical incidents remain rare but high impact. This distribution is consistent with industry standard benchmarks such as the Microsoft GUIDE dataset, where only 19 % of 13 million records are true positives. While the limited sample size of critical alerts (2.8 %) may constrain broad statistical generalizability, this approach was a methodological necessity due to the extreme scarcity of publicly available, descriptive SOC data containing verified ground truth labels. Consequently, this study is positioned as an exploratory evaluation of LLM behavior under realistic SOC constraints, providing results that are indicative of model potential rather than a definitive large scale benchmark. Moreover From this broader pool, a set of 178 alerts was selected and saved respectively, consisting of 104 TPs (58.4 %) and 74 FPs (41.6 %). Priority wise, the distribution is as depicted in Table 3.

This set of alerts was used for the subsequent alert processing step described afterwards.

3.2. Alert processing

Following the generation and collection of alerts, the two JSONL files - `TP_alerts_raw.jsonl` and `FP_alerts_raw.jsonl` - served as the basis for the alert processing step. The processing pipeline, as depicted in Fig. 5, consisted of four dedicated Python scripts that were executed sequentially with the output of each script serving as the input for the next.

Each script was designed to perform a specific task in the pipeline and is described in detail in the following.

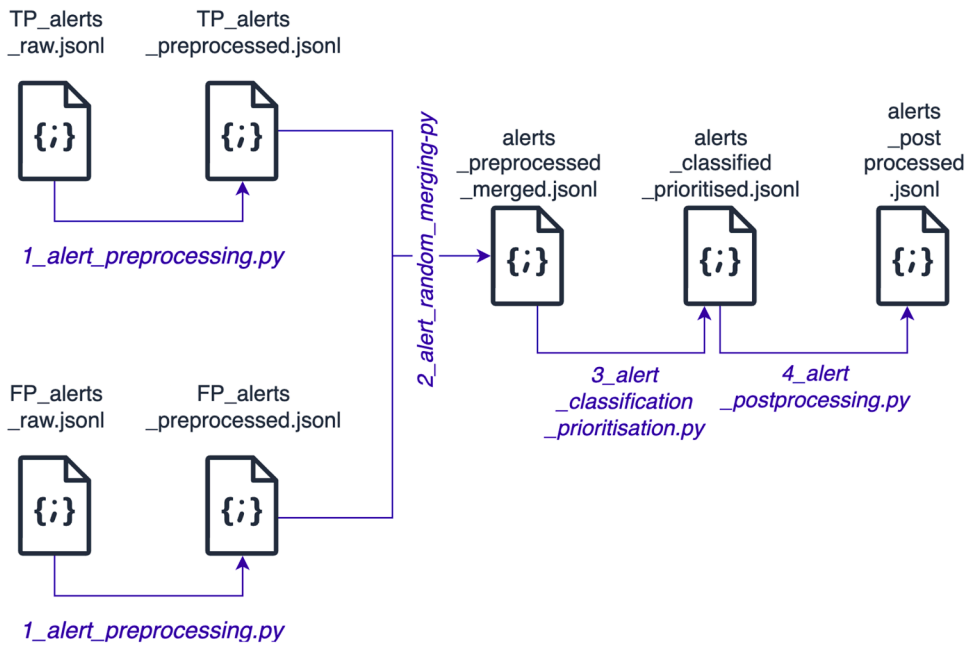

Fig. 5. Alert processing pipeline.

Table 4
Mapping of rule level and categorical severity.

Categorical Severity	Rule Level
Critical	15+
High	12 – 14
Medium	7 – 11
Low	0 – 6

3.2.1. Script 1: preprocessing

The first step was done using the script `1_alert_preprocessing.py`. Overall, the script was executed twice - once for the previously created JSONL file containing TPs and once for its counterpart containing FPs. On the one hand, establishing the ground truth for the classification task i.e. labeling was done using this script. On the other hand, cleaning and normalisation steps were performed. Fields deemed not necessary were removed from all alerts. Other fields such as the numerical priority assigned to each alert by Wazuh, were intentionally seperated from the alert data in order to avoid biasing the LLMs' prioritisation logic later in the prioritisation step. Furthermore, the aforementioned numerical priority was mapped to a categorical severity label for each alert based on the logic described in Table 4. This severity label represented the ground truth for the prioritisation task. Besides that, duplicate alerts were removed, ensuring each alert in the dataset was occurring only once. In summary, two preprocessed JSONL files resulted from the script execution. Each file contained cleaned alerts without ground truth information that could directly be presented to an LLM in the classification and prioritisation step.

Furthermore the ground truth for alert priority was derived directly from the categorical severity levels assigned by the underlying detection tools, such as Wazuh and Suricata. This methodological choice was made to reflect the operational reality of modern, high volume SOC environments where the sheer scale of telemetry necessitates tool based Rule Levels (0-15+) as the primary baseline for triage. While manual human labeling is common in smaller studies, our approach evaluates the LLM's ability to act as a Tier 1 Analyst Assistant that must align with standardized organizational security policies and automated SIEM logic. Furthermore, by utilizing these established rules, we provide an objective, reproducible benchmark that eliminates the subjective variability inherent in manual human prioritization, particularly within the Medium category.

3.2.2. Script 2: merging and randomisation

Taking both of the previously generated output files as input, the second step was carried out using the script `2_alert_random_merging.py` that was executed once. The script merged both files into a single dataset of TPs and FPs that was shuffled and exported to a new JSONL file representing the script's final output. The shuffling was done intentionally to avoid creating a predictable sequence where alerts of the same class are grouped together - a pattern that could potentially bias the LLMs during their classification and prioritisation tasks. The aim was to increase the probability that the LLMs' decisions were based solely on the respective alert content, not on implicit structure or ordering.

Shuffling was implemented as a deliberate methodological choice to ensure the statistical integrity of the evaluation process by eliminating potential ordering effects. Without randomisation, the models might inadvertently detect patterns based on the sequence of data such as a series of consecutive True Positive alerts rather than relying solely on the semantic features of the alert content. We respectfully clarify that this study treats each alert as an independent classification task and does not aim to model temporal dependencies or real time alert streams. Consequently, shuffling does not compromise the nature of the data but instead ensures that no unintended sequence patterns influence the evaluation of the models' diagnostic capabilities.

3.2.3. Script 3: LLM classification and prioritisation

The shuffled JSONL file served as input for the script `3_alert_classification_prioritisation.py`. This core script took the randomised dataset as input and sent each alert to a selected LLM via API for classification and prioritisation. Besides that, key performance metrics were logged, including the total script execution time among others. Additionally, the costs incurred were taken into account, which were obtained from the respective API usage dashboard manually. The actual models used throughout this research are provided in Table 5.

Since interacting with the DeepSeek API required minor adjustments to the script, a second DeepSeek-specific version of this script was

Table 5
LLMs used throughout study.

Model (Common Name)	Model Version / Snapshot
GPT-4o	gpt-4o-2024-08-06
GPT-4.1	gpt-4.1-2025-04-14
GPT-4.5 Preview (Deprecated)	gpt-4.5-preview-2025-02-27
GPT-4o mini	gpt-4o-mini-2024-07-18
GPT-4.1 mini	gpt-4.1-mini-2024-04-14
DeepSeek-Chat	DeepSeek-V3-0324
DeepSeek-Reasoner	DeepSeek-R1-0528
Olmo 3	olmo-3.1-32b-instruct

created. Upon script execution, the following system prompt was provided to OpenAI models:

"You are a cybersecurity expert working as a SOC analyst assistant. Classify the security alert as TP (True Positive) or FP (False Positive) and assign a Priority: Low, Medium, High or Critical."

For DeepSeek models, the exact same prompt was used as base. However, due to DeepSeek's API design, it was necessary to extend it as follows:

"…Always respond in this exact JSON format: '{"id": "alert_id", "classification": "TP or FP", "priority": "Low/Medium/High/Critical", "justification": "short explanation"}'"

The script was run per model, which translated to six runs for the OpenAI variant - as this one could also be used for Olmo 3 - and two for the DeepSeek-specific version. For each run, the script produced a JSONL file containing all initial alerts with the respective LLM's responses appended.

Moreover the prompting strategy employed in this research was intentionally minimalistic to evaluate the intrinsic, zero shot capabilities of general purpose LLMs. Rather than attempting to optimize performance through prompt engineering techniques such as few shot examples, retrieval augmentation, or fine tuning the study aims to establish a conservative baseline, or lower bound, for LLM performance in SOC tasks. This allows for the isolation of fundamental model behavior and the identification of inherent semantic strengths and limitations independent of external enhancements. While we agree that advanced prompt engineering is a vital next step for improving performance, this study serves as a necessary control. We have expanded the Future Research section to highlight sophisticated prompt orchestration and fine tuning as key areas for further investigation by the community.

3.2.4. Script 4: postprocessing and validation

After having received the respective LLM's responses, postprocessing was performed using the script `4_alert_postprocessing.py` that was run eight times in total - once for each model's output file. At its core, the script performed several steps to clean and normalize the model responses. Among others, the LLM's alert classification was normalized to "TP" and "FP" and the LLM's priority classification was validated against the expected categories. The result of this step was a new JSONL file containing properly formatted model responses that was subsequently used for quantitative evaluation. In total, eight such files were generated.

3.3. Result evaluation

After postprocessing, each LLM's responses were evaluated in two steps using the scripts `5_result_evaluation.py` and `6_jsonl_result_to_excel.py` as illustrated in Fig. 6.

Both scripts used the postprocessed JSONL file generated by script 4 as input, were executed eight times in total and are explained in detail in the following.

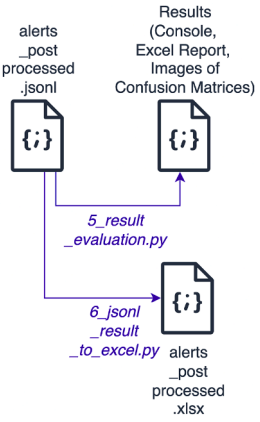

Fig. 6. Result evaluation pipeline.

3.3.1. Script 5: performance analysis

The script `5_result_evaluation.py` calculated and visualised all relevant performance metrics for the classification and prioritisation tasks. Using Python's scikit-learn library as well as manual calculations, key evaluation metrics were calculated based on the four possible prediction outcomes - true positive, true negative, false positive and false negative - explained in Table 6.

The metrics used for evaluation as well as their formulas are described in Table A.17 and Table A.18 of Appendix A. In addition to the metrics calculation, respective confusion matrices were visualised. The final result of the script included a report that was printed to the console and saved in an excel file as well as the confusion matrices saved to image files. The output files were then used for evaluation, manual verification of the metrics and performance comparison of the respective LLMs.

3.3.2. Script 6: result formatting

The final step in the whole processing and evaluation pipeline was performed using the script `6_jsonl_result_to_excel.py`. The script was executed once per model result and converted each postprocessed JSONL file into a structured Excel spreadsheet. This human-readable format allowed for inspection of the LLMs' responses, proving useful for exploratory review and documentation.

3.4. Limitations

The evaluated LLMs were general-purpose models not specifically trained on cyber security data, which may have limited their outputs in this context. Despite trying to simulate a SOC environment, full real-world complexity might not have been captured in the created dataset. The Windows-based test domain used was named "soclab.internal", the main user accounts used on the VMs were the default VirtualBox setup users called "vboxuser" and most alerts were triggered using the Atomic Red Team framework, which may have introduced recognition biases in model outputs. Besides that, most alerts originated from Windows-based systems with limited representation of Linux-based alerts. In addition to that, for classification, the dataset was slightly imbalanced, containing 104 TPs and 74 FPs and for prioritisation it was even further imbalanced towards lower-severity alerts - although this distribution to some degree reflects operational reality where high-severity TPs are comparatively rare. Moreover, all alerts were evaluated in an isolated fashion without preceding or subsequent event context, which in practice would be available to SOC analysts and could affect prioritisation and classification decisions. Furthermore, the study did not assess the qualitative usefulness of the LLMs' responses for SOC analysts, focusing solely on quantitative performance metrics. Finally, prompt engineering, i.e. the effect of different prompt formulations on model performance in this context, was not explored.

Table 6
Definitions of TP, TN, FP and FN.

Term	Meaning in This Context
True Positive (TP)	An alert correctly identified as belonging to its true class
True Negative (TN)	An alert that does not belong to the evaluated class and is correctly predicted as such
False Positive (FP)	An alert incorrectly classified as belonging to a given class when it does not
False Negative (FN)	An alert that truly belongs to a given class but was not classified as such

While we acknowledge the value of large scale datasets for generalizability, extending the current dataset with public repositories was limited by the NLP Security Gap. Most large scale public datasets, such as the Microsoft GUIDE repository, are optimized for traditional machine learning via heavy anonymization and numerical encoding. These sources lack the descriptive, natural language telemetry essential for evaluating an LLM's semantic reasoning capabilities. To maintain linguistic nuance while ensuring ground truth confidentiality which is often restricted in real world SOC environments. This study utilized a high fidelity simulated lab mirroring operational SOC architectures. We have repositioned this work as a foundational and exploratory study, establishing a controlled baseline for LLM behavior that can inform future research as more descriptive, real world text based datasets become available.

The prompting strategy employed in this research was intentionally minimalistic, utilizing a zero shot approach to evaluate the intrinsic capabilities of general purpose LLMs without task specific adaptation. By isolating the models from external enhancements such as few shot learning, retrieval-augmented generation (RAG), or fine-tuning, this study establishes a performance lower bound that reveals inherent model biases and default reasoning patterns in cybersecurity tasks. While increasing the number of prompts or providing expert labeled examples would likely improve accuracy, the current design provides a controlled experimental setting to identify model limitations independent of prompt engineering. Future research, as detailed in Section 6, should build upon this baseline to investigate how advanced orchestration affects both diagnostic accuracy and operational costs.

3.5. Baseline machine learning models

To provide comparative reference points for the binary classification task, lightweight supervised machine learning baselines were additionally implemented. Including baseline comparisons with traditional supervised machine learning models is important when evaluating Large Language Models (LLMs) for classification of Wazuh security alerts. Wazuh alerts already incorporate rule-based detection logic and contextual information, including alert descriptions, rule metadata, severity indicators, decoder types, and agent-related information, which can be effectively utilised by conventional machine learning classifiers through lightweight text representation techniques such as TF-IDF. Consequently, benchmarking LLM performance against these approaches enables assessment of whether semantic modelling and contextual reasoning provide meaningful improvements beyond existing rule-based and statistical prioritisation signals.

Moreover, because LLM-based approaches introduce additional computational overhead compared to lightweight classifiers, baseline comparisons help determine whether their operational cost is justified within SOC environments. This strengthens the methodological validity, interpretability, and practical relevance of the proposed framework.

To provide a comparative baseline, three traditional supervised machine learning models were implemented for the binary alert classification task: Logistic Regression, Random Forest, and Linear Support Vector Machine (SVM). The models were trained using TF-IDF representations extracted from the textual and contextual components of the alerts, including alert descriptions, rule information, command-line artefacts, and related event data.

The merged dataset consisting of 178 labelled alerts was utilised for training and evaluation. The binary classification task focused on distinguishing between true positive and false positive alerts. Stratified 5-fold cross-validation was employed to improve robustness and reduce bias associated with the relatively limited dataset size. All models were evaluated using the same metrics as the LLM-based approaches, including accuracy, precision, recall, F1-score, and false positive rate (FPR), ensuring consistency and comparability across approaches.

4. Results

4.1. Evaluation of GPT-4o

The following confusion matrices were generated based off the alert classifications and prioritisations done by the model.

The illustration shows that out of 104 true positives, 98 were correctly classified as such and only six were misclassified as false positives. This translates to an overall recall of 94.23 % for true positives as shown in Fig. 7 and Table 8. Put differently, the model exhibits a strong threat detection capability, missing only 5.77 % of actual threats. Besides its threat detection strength, the model at the same time classified over 70 % of benign events incorrectly as threats, raising a high amount of false positives. This, in turn, contributed to a relatively modest precision of 65.33 %, indicating that a considerable proportion of all alerts flagged as threats were in fact benign. Regarding the balance between precision and recall, an F1-score of 77.17 % was attained. Overall, the model achieved an accuracy of 67.42 %, meaning that out of all predictions made, 67.42 % were correct.

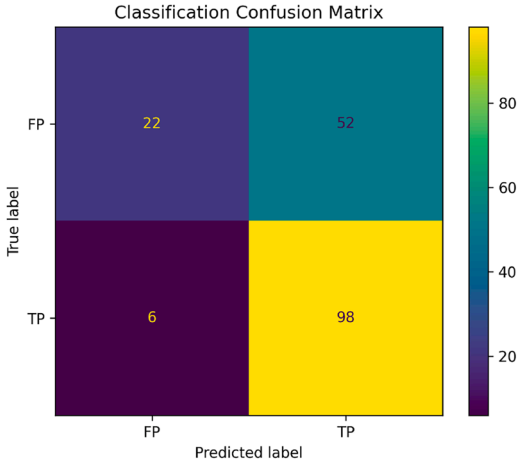

Fig. 7. GPT-4o classification confusion matrix.

Switching to the prioritisation task, the model showed a high precision of 94.44 % in identifying low-priority alerts while at the same time exhibiting a recall for this class of only 25 % as shown in Table 10. This indicates that most low-priority alerts were mistakenly elevated to higher priority levels as proven in the confusion matrix shown in Fig. 8. For high-priority alerts, as shown in Table 12, the recall was at 68 %, but the precision was as low as 22.97 %, meaning that many non-high alerts were wrongly classified as high-priority. For critical alerts, both precision and recall were below the 25 % mark as shown in Table 13

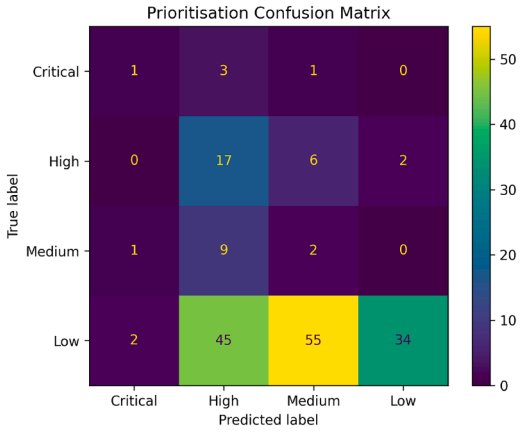

Fig. 8. GPT-4o prioritisation confusion matrix.

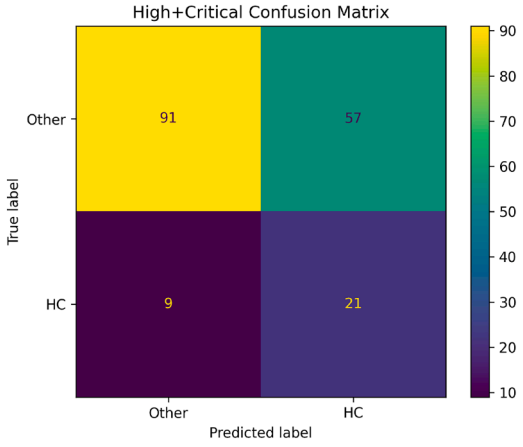

Fig. 9. GPT-4o HC prioritisation confusion matrix.

and medium-priority alerts were barely detected at all as can be seen from Table 11. When looking at the macro F1-score depicted in Table 9, which is at 25.34 %, the balance between recall and precision is relatively low across classes.

Considering the metrics depicted in Table 14, which have been calculated after combining critical as well as high-priority alerts - which both usually require immediate attention in a SOC - into one class "HC", despite the recall being 70 %, precision stays as low as 26.92 %. Translated, this means that, although 70 % of all "HC" alerts were correctly identified as such, out of all alerts predicted to be "HC", only 26.92 % were indeed of high or critical priority. Put differently, a recall of 70 % also means that 30 % of alerts that were of high or critical priority were not prioritised as such - which is reflected in the false negative rate (FNR) also shown in Table 14 and would pose a problem in a real-world SOC surrounding. On the contrary, the false positive rate (FPR) of 38.51 % again confirms the model's tendency to over-escalate low or medium alerts. For the sake of completeness, the confusion matrix for the combined "HC" calculation is depicted in Fig. 9.

With regard to the operational metrics, classification and prioritisation took the model around 2.98 s per alert. The costs incurred were $0.0051 per alert as shown in Table 15 with a total of $0.91 for 178 processed alerts. Overall, the model showcased a high recall - and thus a high threat detection rate - but low precision in alert classification. At the same time, the FPR was just above 70 %. For prioritisation, its performance can be described as rather weak with a tendency to inflate severity, frequently assigning higher priorities than appropriate.

4.2. Evaluation of GPT-4.1

As presented in the confusion matrix shown in Fig. 10, for the classification task, the model performed worse than its predecessor GPT-4o, showing only a moderate threat detection capability. In total, it identified only 73 out of 104 true positives, which corresponds to a recall of 70.19 %.

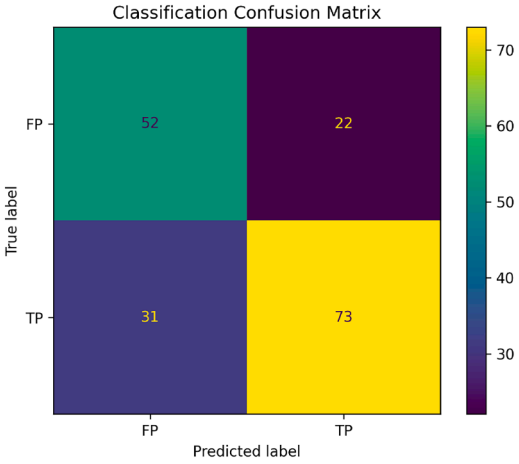

Fig. 10. GPT-4.1 classification confusion matrix.

This implies that nearly 30 % of actual threats went unnoticed, which is reflected in the FNR of 29.81 % shown in Table 8. The model's precision in the identification of TPs was at 76.84 % and thus slightly higher than its recall, indicating a solid level of reliability - most alerts flagged as malicious were indeed threats. As a result, the F1-score was at 73.37 %, reflecting a moderate balance between precision and recall. While exhibiting only moderate threat detection capabilities, the model at the same time misclassified merely 22 out of 74 benign alerts, yielding a FPR of just 29.73 %, which is a significant drop compared to the previously analysed model. In a figurative sense, this means less alert noise. The overall classification accuracy of the model was 70.22 %, meaning that in roughly 70 % of cases, the model's predictions were correct.

Focusing on the model's performance regarding prioritisation, it performed best in recognising low-priority alerts, achieving a precision of 85.88 % and a moderate recall of 53.68 %, as shown in Fig. 11 and Table 10. For higher-severity classes, the model's prioritisation performance deteriorated noticeably as shown in Tables 12 and 13. Similar to GPT-4o, medium-priority alerts were hardly detected, which is being reflected by the corresponding precision of 5.71 % and the recall of 16.67 % shown in Table 11. Besides that, as shown in Fig. 11, criti-

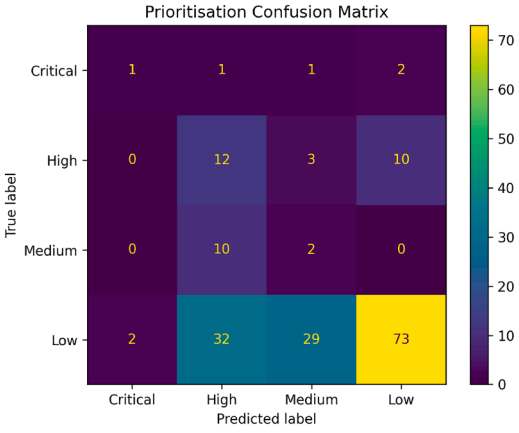

Fig. 11. GPT-4.1 prioritisation confusion matrix.

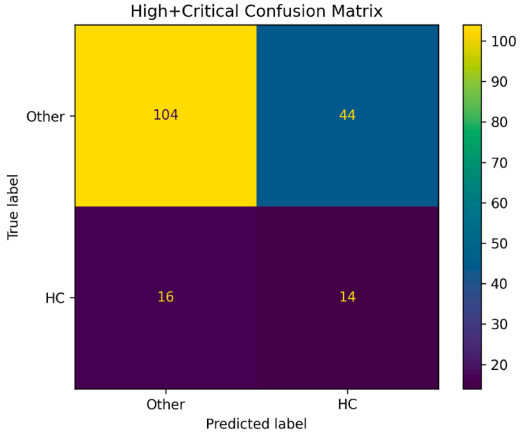

Fig. 12. GPT-4.1 HC prioritisation confusion matrix.

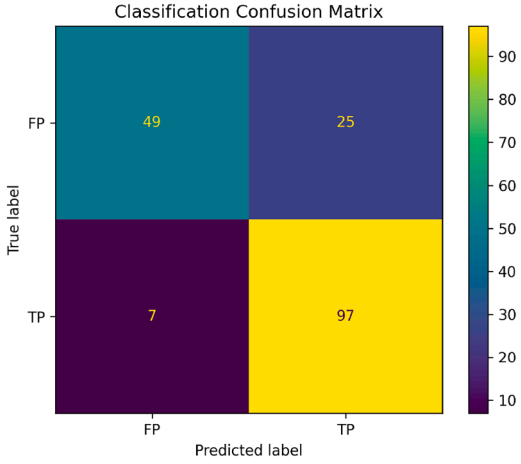

Fig. 13. GPT-4.5 preview classification confusion matrix.

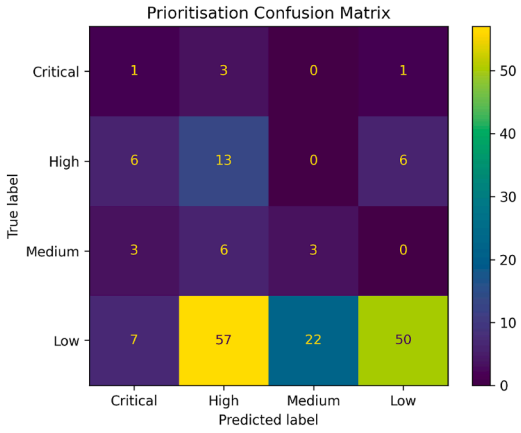

Fig. 14. GPT-4.5 preview prioritisation confusion matrix.

cal alerts were rarely identified correctly and high-priority alerts were flagged with a precision of only 21.82 %. Upon evaluation of the combined class "HC" of high and critical alerts, the model continued to underperform, as illustrated in Table 14. The high FNR of 53.33 % was particularly striking - indicating that more than half of all the high and critical alerts were not recognised as such. The corresponding confusion matrix for the "HC" class is depicted in Fig. 12.

Operationally, the total costs incurred with processing 178 alerts was $0.773, amounting to just $0.0043 per alert. The average processing time per alert was 3.46 s which is slightly slower than its predecessor GPT-4o. Overall, the model demonstrated a cautious classification approach, characterised by high precision and a low false positive rate, i.e. a small amount of alert noise. However, its threat detection coverage was only moderate. With regard to prioritisation, the model managed low-severity prioritisation with some success but struggled to correctly identify and prioritise high-severity alerts.

4.3. Evaluation of GPT-4.5 preview

GPT-4.5 Preview was included in this research's evaluation due to its continued availability through the OpenAI API at the time of experimentation. However, it was later discovered that the model had been officially deprecated by OpenAI prior to the testing period on April 14th, 2025 (OpenAI, 2025) - information that was not indicated during API access in the course of this research. GPT-4.5 Preview was finally removed from OpenAI's API on July 14th, 2025, shortly after conducting this experiment.

Looking at the model's classification performance, it achieved a relatively balanced result in terms of recall, precision and FPR. As depicted in Fig. 13 below, of the 104 true positives, 97 were correctly labelled, leading to a high recall, i.e. threat detection rate of 93.27 % - missing just 6.73 % of genuine threats.

Besides that, the model performed good with filtering out benign alerts, achieving a FPR of 33.78 % as illustrated in Table 8 - far lower than GPT-4o and comparable to GPT-4.1. The precision for TPs was 79.51 %, indicating that most alerts flagged as TP were legitimate threats. Combining this with the high recall, the model achieved a strong F1-score of 85.84 % - the best balance yet between detection capability and false alarm suppression. The model's overall accuracy across all predictions was 82.02 %, also representing the best result thus far.

Turning the focus to the prioritisation results, despite it's classification strength, the model's performance in assigning appropriate priority levels remained limited. As shown in Fig. 14 and Table 9, GPT-4.5 Preview handled low-priority alerts moderately, exhibiting a precision of 87.72 % and a recall of 36.76 % only, indicating that many alerts of that class were over-prioritised. In regard to higher-severity alerts, the model struggled significantly. For critical alerts, a precision of 5.88 % and a recall of only 20 % as shown in Table 13 indicated a severe under-detection of the most urgent threats. While the recall achieved for high-priority alerts was at 52 % as shown in Table 12 and thus better, the precision was again weaker and at 16.46 % only. As shown in Fig. 14 above, the performance on the medium class was mediocre, achieving a low precision of 12 % in combination with a moderate recall of 25 %. The average balance between recall and precision - represented as macro F1-score - across all classes was at just 25.53 % as depicted in Table 9. Considering the joint high-severity category "HC" - comprised of critical and high-priority alerts - prioritisation performance improved slightly, with precision being at roughly 24 % and recall increasing to 76.67 % as Table 14 proves. The corresponding "HC" confusion matrix is depicted in Fig. 15.

When it comes to the operational metrics depicted in Table 15, GPT-4.5 Preview performed similar to its predecessors time-wise, having an average runtime per alert of 3.35 s. However, the costs incurred with its usage were much higher - $22.20 in total for processing 178 alerts. This translates to a per-alert cost of $0.1247, which is more than 28 times higher than for GPT-4.1. Overall, the model exhibited reasonably good classification performance, delivering high recall and precision as well as a relatively low FPR, indicative of low alert noise. However, its prioritisation logic remains flawed. Most significantly, the model's operational cost is prohibitively high and the model itself has been deprecated already, limiting its practical use.

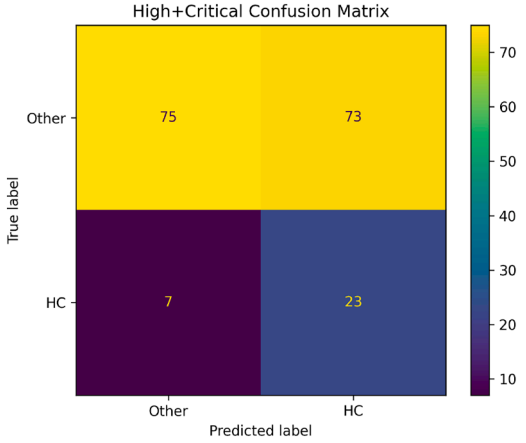

Fig. 15. GPT-4.5 preview HC prioritisation confusion matrix.

4.4. Evaluation of GPT-4o mini

In regard to classification, the model showed a very high recall of 95.19 % for TPs, correctly identifying 99 out of 104 actual threats, as illustrated in the confusion matrix shown in Fig. 16.

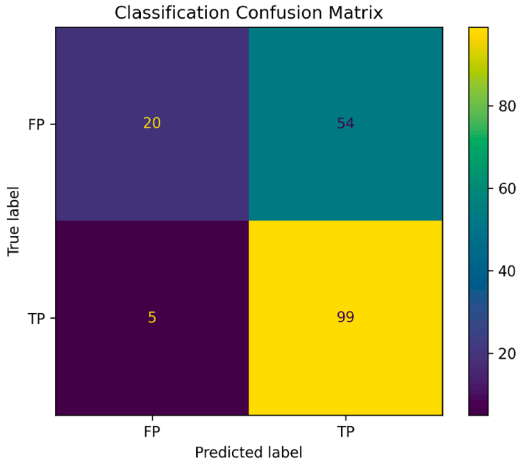

Fig. 16. GPT-4o mini classification confusion matrix.

As a result of the high recall, the FNR was limited to just 4.81 % as shown in Table 8, underscoring the model's ability to catch nearly all security-relevant alerts. In absolute terms, only 5 TPs were not detected. The corresponding TP precision achieved was 64.71 % only, indicating a considerable amount of alerts appeared threatening to the model, but were actually not. Despite its superior threat detection capability, the model at the same time incorrectly labelled 54 out of 74 false positives as threats, inflating the FPR to 72.97 %, which is equivalent to high alert noise. According to the accuracy achieved, the model's predictions were correct in 66.85 % of cases, which is moderate. When it comes to the balance between recall and precision, a mediocre F1-score of 77.04 % was achieved.

In terms of prioritisation, GPT-4o mini exhibited a highly unbalanced performance. As shown in Fig. 17, the model performed relatively well on high-priority alerts, achieving an 80 % recall but a low precision of 22.22 % only, which means that many alerts predicted as high-priority were classified wrongly, even though 80 % of all high-priority alerts were correctly detected. Critical alerts, however, were detected with just 20 % recall and 25 % precision as shown in Table 13, indicating the majority of critical alerts were not recognised as such. The medium class performed worst, with zero percent achieved for both precision

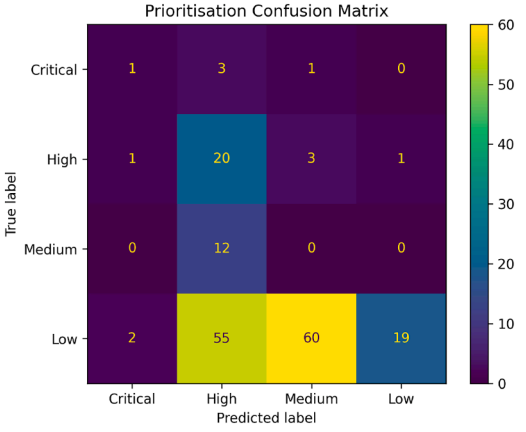

Fig. 17. GPT-4o mini prioritisation confusion matrix.

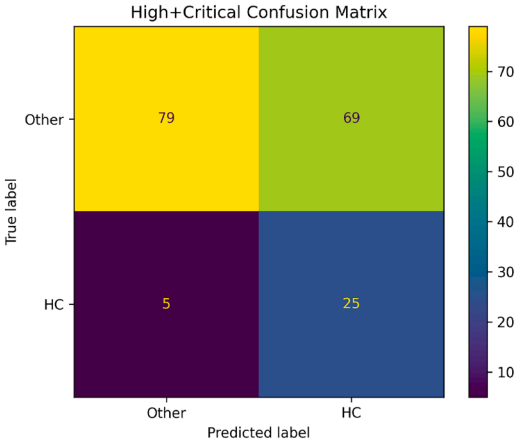

Fig. 18. GPT-4o mini HC prioritisation confusion matrix.

and recall as depicted in Table 11, indicating that the model failed entirely to identify this category. Besides that, low-priority alerts as shown in Table 10 were identified with an excellent precision of 95 % on the one hand, but with a very low recall of just 13.97 % on the other hand, meaning that most low-priority alerts were assigned higher severity levels than necessary - in absolute terms 117 of 136 actual low-priority alerts were incorrectly escalated. The macro F1-score achieved for prioritisation was at 20.34 % as shown in Table 9, indicating an overall poor balance between recall and precision. The results achieved for the combined class of critical and high-priority alerts are illustrated in Table 14. For the "HC" class, 83.33 % of severe alerts were detected and only 16.67 % of serious threats were downgraded. At the same time, the precision was at just 26.60 %, meaning that in 73.40 % of cases, the models predictions were wrong. As a result, the F1-score achieved in that case was at 40.32 %, which is comparatively high. All results previously mentioned can also be traced using the confusion matrix shown in Fig. 18.

Turning the focus to the operational metrics depicted in Table 15, the model processed all 178 alerts at a total cost of $0.008, translating to just $0.000045 per alert - much cheaper than any model evaluated so far. Time-wise, responding took the model only 1.88 s per alert on average which is the fastest processing observed thus far. Summing up, the model exhibits a recall-focused classification strategy, identifying nearly all genuine threats but with a substantial FP burden. Its prioritisation capabilities remain highly imbalanced and weak, particularly with no recognition of medium-priority alerts. Operationally, it sets a new benchmark, being the most cost-effective and at the same time fastest model so far.

4.5. Evaluation of GPT-4.1 mini

For classification and prioritisation, the confusion matrices are depicted in Figs. 19 and 20. As can be seen from the classification matrix, out of 104 threats, 86 were correctly detected only, corresponding to a comparatively moderate recall of 82.69 %. This, in turn, means that 17.31 % of TPs were missed.

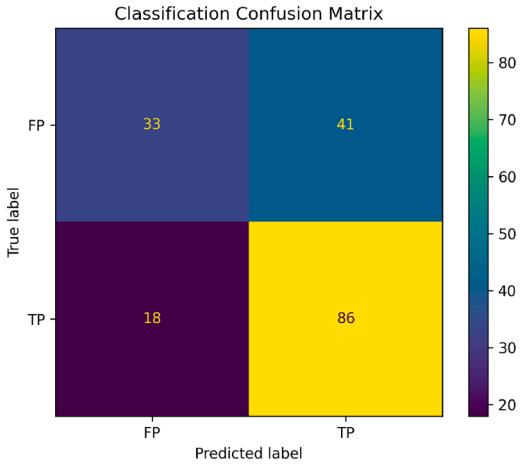

Fig. 19. GPT-4.1 mini classification confusion matrix.

The precision exhibited for TPs was 67.72 % as shown in Table 8, indicating that over one third of the models predictions were in fact wrong. In addition to that, the overall FPR for classification was only mediocre at 54.05 %, which indicates significant alert noise - 41 FPs in absolute terms. Overall, the models accuracy of classification landed at moderate 66.85 %. The balance achieved between recall and precision was at mediocre 74.46 %.

In terms of prioritisation, the model encountered notable difficulties as shown in Table 9. Even though for the low-priority class a high precision of 90.38 % was achieved as shown in Table 12, it suffered from a low recall of 34.56 %, meaning that alerts of this lowest class were often assigned undue urgency, as proven in Fig. 20. For high-priority alerts, the result was most balanced with a recall of 56 % and a precision of 24.56 % as depicted in Table 12. Critical alerts, on the other hand, were poorly handled, with both precision and recall being at 20 % as shown in Table 13 - showing the model's tendency to under-escalate serious threats. Similar to the previous model, the performance for the medium class depicted in Table 11 was also poor, registering just 4.69 % precision and 25 % recall. Overall, the macro F1-score achieved for prioritisation was just 28.01 %. Having combined the critical and high-priority classes, the results depicted in Table 14 were obtained. With a recall of 53.33 %, a concerning amount of 46.67 % of serious threats - equivalent to 14 alerts as illustrated in Fig. 21 - was downgraded to non-urgent status. Furthermore, the low precision of 25.81 % indicates that 74.19 % of alerts flagged as serious were in fact not serious. As a result, the F1-score achieved was only 34.78 %, reflecting the model's challenge with handling high-severity alerts.

From an operational SOC perspective, these findings are particularly significant because the High and Critical priority categories (Wazuh Rule Levels 12+) represent the primary thresholds for immediate incident response and escalation to Tier 2 or Tier 3 analysts. While models such as GPT-4o mini demonstrated strong recall performance at these thresholds, minimising the risk of missed high-impact threats, this was accompanied by substantially elevated false positive rates. For example, GPT-4o mini achieved a recall of 83.33 % for the combined HC category but also produced an FPR of 46.62 %. Such over-escalation behaviour would likely exacerbate alert fatigue and overwhelm senior analysts with non-critical events within operational SOC environments. These findings therefore suggest that while LLMs can effectively identify potentially serious threats and support threat coverage, their current precision limitations still require human-in-the-loop verification to prevent operational bottlenecks at critical escalation points.

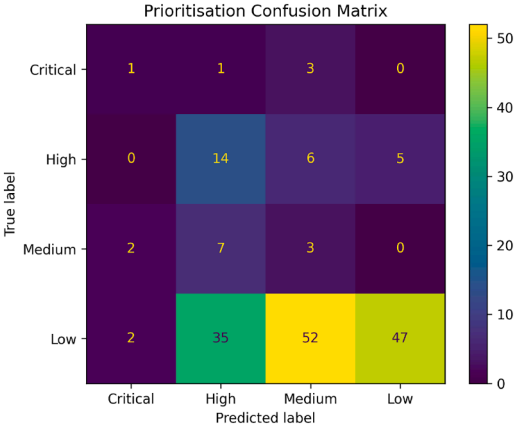

Fig. 20. GPT-4.1 mini prioritisation confusion matrix.

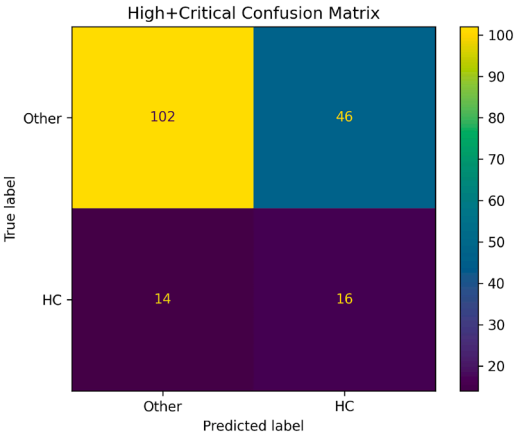

Fig. 21. GPT-4.1 mini HC prioritisation confusion matrix.

In terms of efficiency, GPT-4.1 mini incurred costs amounting to $0.14 in total as depicted in Table 15, resulting in a per-alert cost of $0.000787, which is cheaper than for larger models like GPT-4.5 Preview. Regarding the runtime, the model processed alerts on average within 2.43 s. In summary, the model delivered average classification performance with a relatively high false positive rate. In regard to prioritisation, it performed as imbalanced as the other models. Operationally, the costs incurred were rather low and the time taken to respond on a per-alert basis was also among the lowest ones.

4.6. Evaluation of deepseek-chat

Considering the model's classification results, illustrated in Fig. 22, reveals that it classified 96 out of 104 true threats correctly, resulting in a decent recall of 92.31 %. Accordingly, the FNR was limited to 7.69 %, meaning that 8 of the TPs were missed in total.

As shown in Table 8, the model's precision landed at 70.59 %, which reflects a moderate level of trustworthiness in threat predictions. Taken both recall and precision together, resulted in a strong F1-score of 80 %, indicating a strong balance between catching threats and labelling them accurately. Apart from that, the model flagged 40 out of 74 FPs as threats, which led to a mediocre FPR of 54.05 %. The correctness across all of the model's predictions was at reasonably well 73.03 %.

Turning to the prioritisation results shown in Fig. 23 and Table 9, underlines that the model's prioritisation performance was mixed. As shown in the corresponding confusion matrix above, low-priority alerts

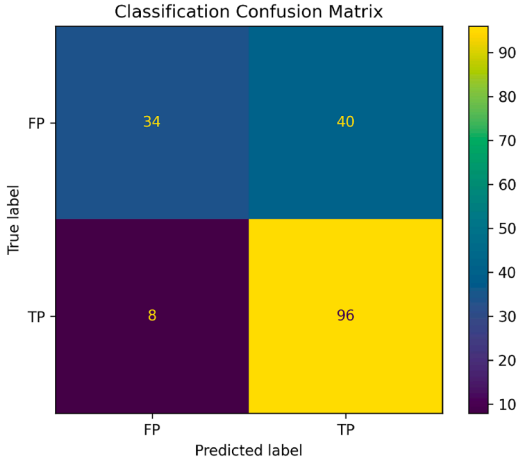

Fig. 22. DeepSeek-chat classification confusion matrix.

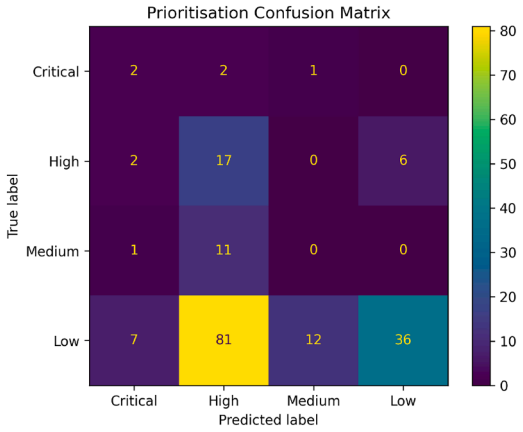

Fig. 23. DeepSeek-chat prioritisation confusion matrix.

were classified with 85.71 % precision but only 26.47 % recall, meaning that, although DeepSeek-Chat's predictions in this regard were precise, a large proportion of low-priority alerts was assigned a higher severity. Same as in the case of GPT-4o mini, the medium-priority class was left entirely unrecognised as depicted in Table 11. Regarding the high-priority category, a recall of 68 % in combination with a weak precision of 15.32 % was achieved as shown in Table 12. For the critical alerts, 40 % of them were correctly classified, albeit with a low precision of 16.67 %. While not yet reliable, this is the highest recall observed for critical alerts so far. As shown in Table 14, for the combined class "HC", a recall of 76.67 % was achieved, thus just 23.33 % of serious threats were downgraded to a lower priority. While achieving a decent recall, the precision for the combined class was at 18.70 % only, which means that out of all predictions for the "HC" class, 81.30 % were in fact wrong. As shown in the confusion matrix below, this translates to 100 alerts wrongly prioritised as serious. As a result, the achieved F1-score was as low as 30.06 % (Fig. 24).

In terms of resource usage, DeepSeek-Chat performed inexpensive but comparatively slow as shown in Table 15. With a total cost of $0.04 and $0.000225 per alert, the model is one of the less expensive models. However, the average runtime per alert was very slow at 9.03 s. Overall, the model provides a solid threat detection profile, combining high recall with a reasonable F1-score and accuracy. Its ability to detect critical alerts with such a high recall distinguishes it from the previously evaluated models. However, precision and FPR remain weak points. While it is cost-effective, it performs noticeably slower than the other models evaluated.

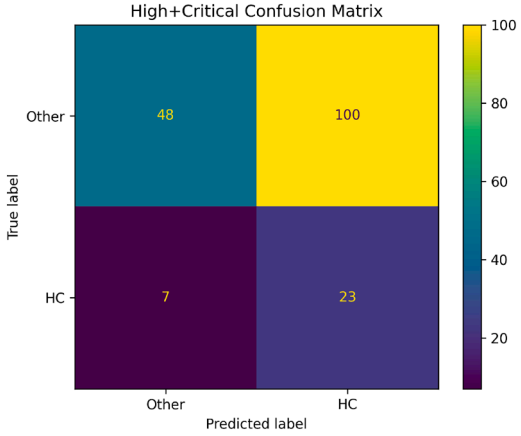

Fig. 24. DeepSeek-chat HC prioritisation confusion matrix.

4.7. Evaluation of DeepSeek-reasoner

Beginning with the classification results, as shown in the respective confusion matrix below, DeepSeek-R1-0528 identified only 58 of 104 TP alerts, which corresponds to a low recall of just 55.77 %. This translates to the highest false negative rate among all evaluated models of 44.23 %, meaning that the model missed nearly half of all actual threats (Fig. 25).

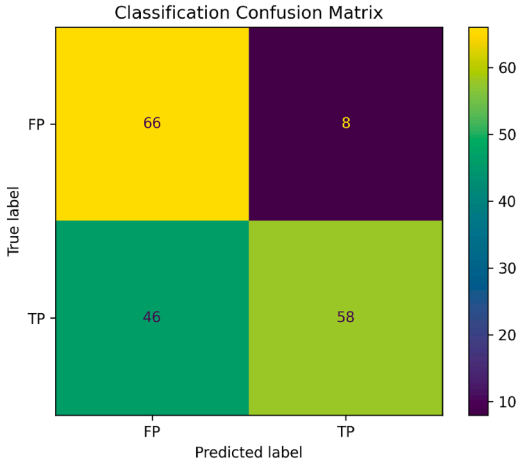

Fig. 25. DeepSeek-reasoner classification confusion matrix.

While the model's recall was rather low, the precision for TPs achieved was 87.88 %, as shown in Table 8, indicating highly trustworthy TP predictions. Yet, considering both recall and precision, resulted in the lowest F1-score observed among the evaluated models of 68.24 %. On the contrary, the model showed the best performance in filtering benign events, misclassifying only 8 out of 74 FPs, which resulted in an exceptionally low FPR of just 10.81 % - put differently, the model was strong in reducing alert noise. Overall, the model's classification accuracy was at mediocre 69.66 %.

Focusing on the prioritisation results, the model struggled with this task similar to the other models, as depicted in Fig. 26 and Table 9. The best performance was achieved for the low-priority class depicted in Table 10, classifying 58.09 % of it correctly with a precision of 82.29 %, which means that the majority of all low-priority alerts were recognised as such. Same as other models such as the previously evaluated DeepSeek model, DeepSeek-R1-0528 failed entirely to detect medium-priority alerts as shown in Table 11. In regard to high-priority alerts, a recall of just 36 % was achieved with an even lower precision of 14.75 % as shown in Table 12. For critical alerts, as depicted in Table 13, the situation was even worse, having achieved a recall of 20 % and a preci-

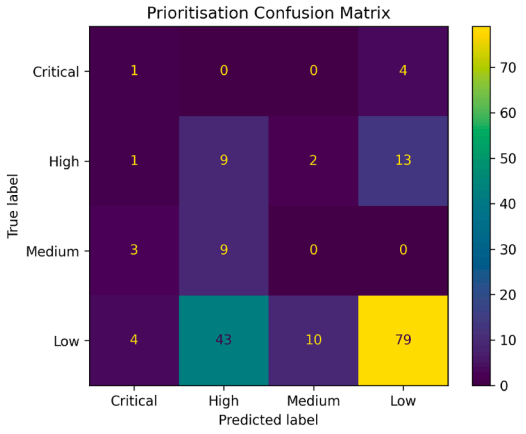

Fig. 26. DeepSeek-reasoner prioritisation confusion matrix.

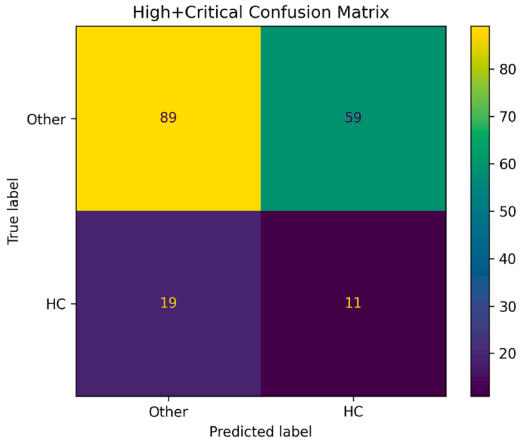

Fig. 27. DeepSeek-reasoner HC prioritisation confusion matrix.

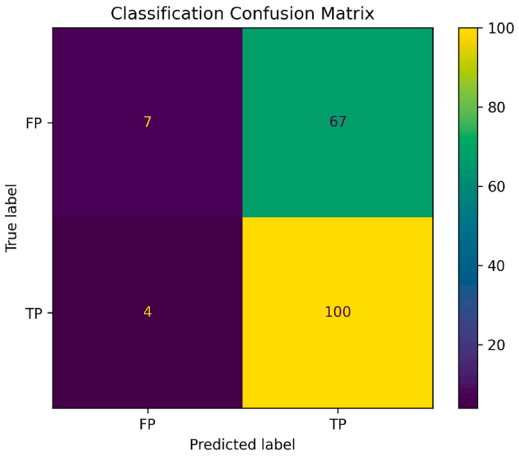

Fig. 28. Olmo 3 classification confusion matrix.

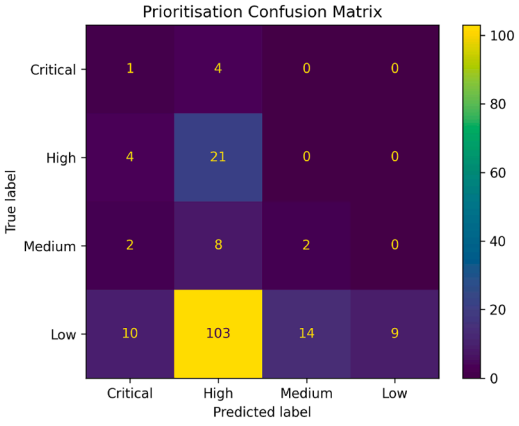

Fig. 29. Olmo 3 prioritisation confusion matrix.

sion of 11.11 %. As shown in the corresponding confusion matrix above, only 1 out of 5 critical alerts were correctly identified. Looking at the combined class of high-priority and critical alerts, the recall achieved was just 36.67 % as shown in Table 14, meaning that 63.33 % of serious threats were downgraded, which is the worst result among all models evaluated. In addition to that, serious threats were predicted with a precision of just 15.71 %. In total, this resulted in a low F1-score of 22.00 % for this combined class. The confusion matrix corresponding to the mentioned results is depicted in Fig. 27.

In regard to the model's operational metrics depicted in Table 15, it showed an extremely high latency of 85.8 s per alert, being significantly slower than any other of the evaluated models. The costs incurred with the usage of the model were $0.17 in total, which translates to $0.000955 per alert - comparable to the other models. In sum, the model exhibited poor threat detection performance with nearly half of all actual threats missed. At the same time, it performed reasonably well in regard to alert noise reduction. For prioritisation, it performed best on the low-priority class but showed an overall tendency to underestimate the severity of alerts. While the costs incurred with the model's usage are competitive, it exhibited the worst latency of 85.8 s per alert.

4.8. Evaluation of Olmo 3

For this model, a classification recall of 96.15 % was achieved which means that out of 104 TPs, 100 were identified as such and only four were misclassified as FPs as can be seen in the classification confusion matrix in Fig. 28. This in turn leads to a FNR of 3.85 % - that is, only 3.85 % of threats were missed which is the best result among all evaluated models.

At the same time however, the model misclassified 67 out of 74 FPs which results in the highest FPR among all models of 90.54 % as can be seen in Table 8. This translates to a lot of alert noise being generated. Furthermore, the model exhibited the worst precision as well as the worst accuracy in regard to TPs. The F1-score achieved was also at the lower end compared to the models previously evaluated.

Switching to the prioritisation results available in Fig. 29 and Table 9, it becomes clear that the model exhibited one of the worst prioritisation performances overall. While the precision in recognising low priority alerts was exceptionally good at 100 %, the recall for this type of alerts was at 6.62 % only as depicted in Table 10 - the worst performance by far. For alerts classified as medium, the model again exhibited the best precision of 12.50 % and a decent recall of 16.67 % only as shown in Table 11. This means that the model's classifications of low and medium alerts were trustworthy on the one hand but only very little of these kinds of alerts were correctly identified as low or medium. With respect to high priority alerts, the model identified them with the highest recall of 84.00 %, however the precision of 15.44 % for this class was among the lowest as depicted in Table 12. When it comes to alerts classified as critical, the model performed as bad as GPT-4.5 Preview, achieving a recall of 20.00 %, a precision of 5.88 % and an F1-score of 9.09 % only - as depicted in Table 13.

However, after combining the high and critical classes to the HC class, the model outperformed all other models when it comes to the recall as shown in Fig. 30 and Table 14. It correctly identified all HC alerts as such, achieving a recall of 100 %. This in turn leads to a FNR of 0.00 % which means that none of the most important alerts were missed. At the same time, the model's precision and F1-score were among the

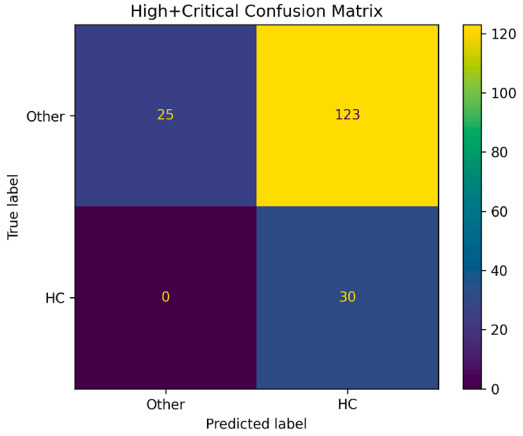

Fig. 30. Olmo 3 HC prioritisation confusion matrix.

Table 7
Baseline machine learning classification results.

Model	Accuracy	Precision	Recall	F1-score	FPR
Logistic Regression	83.78 %	85.07 %	88.48 %	86.18 %	22.97 %
Random Forest	83.17 %	89.40 %	80.71 %	84.60 %	13.51 %
Linear SVM	88.24 %	90.49 %	89.43 %	89.64 %	13.51 %

bottom three results and the FPR was the highest by far with 83.11 %, indicative of a massive amount of alert noise.

Focusing on the operational metrics, the model's results ranked among the bottom three with an average runtime per alert of 4.73 s. Cost-wise it was among the cheapest ones, incurring costs of $0.071 in total which translates to $0.0004 per alert.

Summing up, the model performed best in regards to threat detection while at the same time struggling with a lot of alert noise. In regard to prioritisation, it hardly prioritised low and medium alerts as such. For the high priority class, the model performed much better, however still struggling with precision. Its best performance was achieved for the combined HC class where it however significantly struggled with a lot of alert noise.

4.9. Baseline machine learning results

The performance of the baseline machine learning models for binary alert classification is summarised in Table 7.

The results demonstrate that traditional machine learning models achieve strong performance for the binary classification task. Among the evaluated approaches, the Linear SVM achieved the strongest overall performance, with an F1-score of 89.64 %, an accuracy of 88.24 %, the highest recall of 89.43 %, and a comparatively low false positive rate of 13.51 %. Random Forest demonstrated strong precision performance with a comparatively low false positive rate, while Logistic Regression also achieved balanced performance, although with a higher false positive rate of 22.97 %.

These findings indicate that lightweight statistical and machine learning approaches remain highly effective for structured SOC alert classification tasks. The strong performance achieved using TF-IDF feature representations suggests that conventional classifiers are capable of exploiting important contextual and rule-based signals already embedded within Wazuh alerts.

Compared to the LLM-based approaches presented in Table 8, the baseline machine learning models achieved competitive and, in some cases, superior classification performance. Several LLMs achieved exceptionally strong recall values, particularly Olmo 3, GPT-4o-mini, and GPT-4o. However, when considering overall balance across precision, recall, F1-score, and false positive rate, the machine learning baselines demonstrated more stable performance. GPT-4.5-preview achieved the strongest and most balanced LLM performance with an F1-score of 85.84 %; however, the Linear SVM baseline exceeded this result with an F1-score of 89.64 %. Similarly, Random Forest outperformed several proprietary and open weight LLMs, including Olmo 3, GPT-4o-mini, GPT-4.1-mini, and DeepSeek-Reasoner. Furthermore, the machine learning models achieved lower false positive rates than the evaluated LLMs in most of the cases, which is desirable in SOC environments as it contributes to reduced alert noise and lower analyst workload.

5. Discussion

This section focuses on comparing all eight models. Comparison is done across the three dimensions classification performance, prioritisation capability and operational metrics.

5.1. Classification results

The results achieved by all eight LLMs for classification are summarised in Table 8.

Considering recall as the most important metric in this context, as it directly reflects a model's ability to identify threats, Olmo 3 detected most of all threats - and thus missed the fewest - exhibiting a recall of 96.15 %. However, its other metrics are rather mediocre and its FPR is the highest among all models. The overall most balanced performance was demonstrated by GPT-4.5 Preview, achieving a recall of 93.27 %, both the highest F1-score and accuracy as well as a comparatively low FPR of 33.78 %, suggesting both excellent capabilities in detecting threats as well as in reducing FPs. On the contrary, DeepSeek Reasoner struggled the most with threat detection, had the highest precision and lowest FPR. Summing up, while the results are mixed, most of the models achieved a threat detection rate of over 90 %, proving their general ability to classify alerts appropriately. Also, no model performed ideal in all regards, but GPT-4.5 Preview showed that a balanced performance is possible in general. Thus, it can be concluded that for classification, in combination with human oversight, LLMs can be used in SOC contexts to support alert triage.

5.2. Prioritisation results

For prioritisation, the summarised results are provided in Table 9.

Considering the macro-averaged metrics, in this case, ensured that equal weight was given to all classes - low, medium, high and critical - despite the imbalance in the dataset. The best overall performance was exhibited by GPT-4.1, which achieved the best recall, precision and F1-score as well as the second best accuracy. GPT-4o mini, on the contrary, demonstrated the worst performance, having a good precision but weak recall, F1-score and accuracy. However, it must be stated that most of the metrics are very close together, with no major fluctuations, which suggests that the prioritisation task was difficult for all LLMs, as no model excelled universally. The results for each priority are considered in detail in the following. Table 10 lists the results achieved by the models for low-priority alerts.

Precision was considered the most critical metric in this case, as in a SOC context - and from an analyst's perspective - it is essential that alerts predicted as low can reliably be trusted to be low in priority and thus ignored - otherwise potential threats are missed. Based on this premise,

Table 8
Classification results comparison.

Model	Recall	FNR	Precision	F1-score	Accuracy	FPR
Olmo 3	96.15 %	3.85 %	59.88 %	73.80 %	60.11 %	90.54 %
gpt-4o-mini	95.19 %	4.81 %	64.71 %	77.04 %	66.85 %	72.97 %
gpt-4o	94.23 %	5.77 %	65.33 %	77.17 %	67.42 %	70.27 %
gpt-4.5-preview	93.27 %	6.73 %	79.51 %	85.84 %	82.02 %	33.78 %
deepseek-chat	92.31 %	7.69 %	70.59 %	80.00 %	73.03 %	54.05 %
gpt-4.1-mini	82.69 %	17.31 %	67.72 %	74.46 %	66.85 %	55.41 %
gpt-4.1	70.19 %	29.8 %	76.84 %	73.37 %	70.22 %	29.73 %
deepseek-reasoner	55.77 %	44.23 %	87.88 %	68.24 %	69.66 %	10.81 %

Table 9
Prioritisation results comparison.

Model	Macro Recall	Macro Precision	Macro F1-score	Accuracy
gpt-4.1	34.59 %	36.69 %	32.39 %	49.44 %
gpt-4.1-mini	33.89 %	34.91 %	28.01 %	36.52 %
deepseek-chat	33.62 %	29.42 %	22.24 %	30.90 %
gpt-4.5-preview	33.44 %	30.51 %	25.53 %	37.64 %
gpt-4o	32.42 %	36.39 %	25.34 %	30.34 %
Olmo 3	31.82 %	33.46 %	15.47 %	18.54 %
deepseek-reasoner	28.52 %	27.04 %	25.83 %	50.00 %
gpt-4o-mini	28.49 %	35.56 %	20.34 %	22.47 %

Table 10
Low-priority prioritisation results comparison.

Model	Precision	Recall	F1-score
Olmo 3	100 %	6.62 %	12.41 %
gpt-4o-mini	95.00 %	13.97 %	24.36 %
gpt-4o	94.44 %	25.00 %	39.53 %
gpt-4.1-mini	90.38 %	34.56 %	50.00 %
gpt-4.5-preview	87.72 %	36.76 %	51.81 %
gpt-4.1	85.88 %	53.68 %	66.06 %
deepseek-chat	85.71 %	26.47 %	40.45 %
deepseek-reasoner	82.29 %	58.09 %	68.10 %

Table 11
Medium-priority prioritisation results comparison.

Model	Recall	Precision	F1-score
gpt-4.5-preview	25.00 %	12.00 %	16.22 %
gpt-4.1-mini	25.00 %	4.69 %	7.89 %
Olmo 3	16.67 %	12.50 %	14.29 %
gpt-4o	16.67 %	3.13 %	5.26 %
gpt-4.1	16.67 %	5.71 %	8.51 %
gpt-4o-mini	0.00	0.00	0.00
deepseek-chat	0.00	0.00	0.00
deepseek-reasoner	0.00	0.00	0.00

Table 12
High-priority prioritisation results comparison.

Model	Recall	Precision	F1-score
Olmo 3	84.00 %	15.44 %	26.09 %
gpt-4o-mini	80.00 %	22.22 %	34.78 %
gpt-4o	68.00 %	22.97 %	34.34 %
deepseek-chat	68.00 %	15.32 %	25.00 %
gpt-4.1-mini	56.00 %	24.56 %	34.15 %
gpt-4.5-preview	52.00 %	16.46 %	25.00 %
gpt-4.1	48.00 %	21.82 %	30.00 %
deepseek-reasoner	36.00 %	14.75 %	20.93 %

Table 13
Critical-priority prioritisation results comparison.

Model	Recall	Precision	F1-score
deepseek-chat	40.00 %	16.67 %	23.53 %
gpt-4.1	20.00 %	33.33 %	25.00 %
gpt-4o	20.00 %	25.00 %	22.22 %
gpt-4o-mini	20.00 %	25.00 %	22.22 %
gpt-4.1-mini	20.00 %	20.00 %	20.00 %
deepseek-reasoner	20.00 %	11.11 %	14.29 %
gpt-4.5-preview	20.00 %	5.88 %	9.09 %
Olmo 3	20.00 %	5.88 %	9.09 %

Table 14
HC combined priority prioritisation results comparison.

Model	Recall	FNR	Precision	F1-score	FPR
Olmo 3	100 %	0.00 %	19.61 %	32.79 %	83.11 %
gpt-4o-mini	83.33 %	16.67 %	26.60 %	40.32 %	46.62 %
deepseek-chat	76.67 %	23.33 %	18.70 %	30.07 %	67.57 %
gpt-4.5-preview	76.67 %	23.33 %	23.96 %	36.51 %	49.32 %
gpt-4o	70.00 %	30.00 %	26.92 %	38.89 %	38.51 %
gpt-4.1-mini	53.33 %	46.67 %	25.81 %	34.78 %	31.08 %
gpt-4.1	46.67 %	53.33 %	24.14 %	31.82 %	29.73 %
deepseek-reasoner	36.67 %	63.33 %	15.71 %	22.00 %	39.86 %

Olmo 3 performed best, achieving a precision of 100 % but a recall as low as 6.62 %, which means that it missed most of the low-priority alerts. The overall most balanced performance in low-priority classification was demonstrated by the DeepSeek-Reasoner model that achieved the best recall and F1-score but the worst precision. The results suggest that, while LLMs can identify some low-priority alerts, their ability to do so without misclassifying them as more serious remains limited, i.e., models tend to over-escalate low-priority alerts. Changing the focus to the medium-priority alerts, it becomes apparent that all LLMs struggled to identify those, as Table 11 shows.

With a recall of 25 %, a precision of 12 % and an F1-score of 16.22 %, GPT-4.5 Preview performed best, achieving the highest values in to out of the three metrics. While all other models performed worse, some failed to detect this class at all. These results indicate systemic struggles, suggesting that medium-priority alerts are inherently harder to distinguish for LLMs than low-priority alerts for example. Switching to high-priority alerts, recall-wise, model performance was best compared to the other classes, as illustrated in Table 12.

Considering recall as the most important metric for this class's comparison - as in a SOC context it is important to identify all high-priority alerts - Olmo 3 achieved the best result with a recall of 84 %, i.e., of all high-priority alerts, it classified 84 % correctly as high. With preci-

Table 15
Operational metrics results comparison.

Model	Avg. Runtime per Alert (s)	Costs per Alert ($)
gpt-4o-mini	1.88	0.000045
gpt-4.1-mini	2.43	0.000787
gpt-4o	2.98	0.005112
gpt-4.5-preview	3.35	0.124730
gpt-4.1	3.46	0.004343
Olmo 3	4.73	0.000400
deepseek-chat	9.03	0.000225
deepseek-reasoner	85.8	0.000955

Table 16
Comparison between baseline machine learning models and LLM-based approaches for binary alert classification.

Approach	Model	Precision	Recall	F1-score
Machine Learning	Linear SVM	90.49 %	89.43 %	89.64 %
LLM (Proprietary)	GPT-4.5-preview	79.51 %	93.27 %	85.84 %
LLM (Open source)	Olmo 3	59.88 %	96.15 %	73.80 %

sion and F1-score taken into consideration, GPT-4o mini demonstrated the overall most balanced performance. Besides that, it is striking that recall for this class fluctuates significantly across models, ranging from 36 % to 84 %, which reflects substantial inconsistency among models in detecting these alerts. Additionally, the F1-scores achieved are generally low, peaking at 34.78 %. In sum, most models showed a reasonable ability to identify high-priority alerts, with some, like Olmo 3, achieving strong recall. However, consistently low precision indicates that many predicted high-priority alerts were in fact of other severity, limiting their operational trustworthiness and introducing alert noise. Continuing with considering the results for critical alerts, performance was limited as depicted in Table 13.

Most of the models struggled to detect critical alerts, achieving recall rates of only 20 %. Additionally, precision remained low across the board, resulting in uniformly weak F1-scores. Focusing on recall, DeepSeek-Chat outperformed the other LLMs, achieving twice the recall of all other models, while at the same time exhibiting a rather mediocre precision only. In sum, these results indicate that, although the DeepSeek model provided a comparatively better coverage of critical alerts, no model demonstrated reliable performance in this important class, highlighting a key limitation of current LLMs in prioritising the most impactful security threats. Taking the combined class of high and critical alerts - from an operational lens the most relevant one in this context - into focus, reveals that the models' performance varied considerably, as shown in Table 14.

The overall best performance recall-wise was exhibited by Olmo 3, achieving a recall of 100 % and thus a FNR of 0.00 %. While this is an excellent result, the model at the same time attained the highest FPR of 83.11 %. The overall most balanced performance was demonstrated by GPT-4o mini, achieving a recall of 83.33 %, 40.32 % F1-score and, in turn, a low FNR of 16.67 %, which is essential in real-world SOC surroundings, as it indicates the proportion of serious threats missed. However, the model at the same time exhibited a comparatively high FPR, indicating significant alert noise. In contrast, DeepSeek-Reasoner performed worst overall, failing to identify nearly two-thirds of all serious alerts - 63.33 % specifically. Summing up, the wide spreads in the results highlight that the models detected and prioritised serious threats differently well. While some models excelled at identifying high-severity alerts, achieving high recall and low FNRs, others missed a significant amount of these. However, precision remained consistently low and FPRs exceeded approximately 30 % across all models, indicating significant alert noise. Thus, regarding serious alerts, the findings highlight the challenge of LLMs with achieving a reliable balance between accurately identifying most of the high and critical alerts and avoiding the introduction of excessive alert noise at the same time.

Furthermore systemic failure of all evaluated models to accurately detect Medium priority alerts-and the resulting tendency toward over escalation highlights a fundamental limitation in current LLM based triage. This behavior is likely a combination of a risk averse bias where models prioritize avoiding false negatives by inflating severity and the inherent semantic ambiguity of intermediate priority levels. Unlike Critical or Low alerts, which often have distinct technical indicators, the Medium category lacks clear action oriented boundaries, even for human experts. Without organizational context such as asset criticality, LLMs struggle to differentiate this grey area from high priority threats, resulting in significant alert noise that could increase analyst fatigue in an operational setting.

5.3. Operational metrics

Focusing on the practical applicability of LLMs in real-world SOC contexts, Table 15 summarises the measured per-alert values for runtime and cost across all evaluated models.

GPT-4o mini offered the best operational profile with the fastest response time of 1.88s and the lowest cost of $0.000045 per alert. While most models performed similar time-wise, the DeepSeek models were at least roughly two times slower. In fact, with an average runtime per alert of 85.80s, the slowest model was DeepSeek-Reasoner - many times slower than all other models. Considering the costs, they were comparable in the majority of cases, with one exception being the costs incurred with using GPT-4.5 Preview. With $0.12 per alert, it is the most expensive model. This resulted in a total cost of approximately $22, while the total costs for all other models were below $1 each. Based on these results, it can be concluded that the deployment of most evaluated models appears operationally feasible in real-world SOC contexts, both in terms of per-alert cost and processing time.

5.4. Comparative analysis with machine learning baselines

To further evaluate the practical value of LLM-based approaches, the classification performance of the machine learning baselines was compared with the evaluated proprietary, open weight and open source LLMs. Table 16 summarises the best-performing machine learning and LLM-based approaches for the binary alert classification task.

The comparison highlights important differences in the behaviour of traditional machine learning models and LLM-based approaches. The machine learning baselines demonstrated more balanced performance across precision and recall metrics, achieving consistently strong F1-scores and relatively low false positive rates. In particular, the Linear SVM baseline achieved the strongest overall classification performance among all evaluated models.

In contrast, several LLMs demonstrated exceptionally high recall, particularly Olmo 3, GPT-4o-mini, and GPT-4o, indicating strong capability in identifying true positive alerts. However, this often came at the cost of substantially lower precision and elevated false positive rates. Such behaviour suggests that LLMs tend to over-classify alerts as malicious, potentially increasing alert fatigue within operational SOC environments.

The open source Olmo 3 model demonstrated competitive recall performance but exhibited lower precision and overall F1-score compared to both proprietary LLMs and traditional machine learning baselines. This indicates that while open source LLMs offer promising accessibility and lower deployment barriers, their practical effectiveness for SOC alert classification may still lag behind more mature proprietary systems and lightweight statistical classifiers.

From an operational perspective, the results suggest that lightweight machine learning models remain highly effective for binary SOC alert classification tasks due to their strong predictive performance, lower computational requirements, and greater stability. Conversely, LLM-based approaches appear more suitable for scenarios requiring richer contextual interpretation and semantic reasoning, particularly in more

complex prioritisation tasks where rigid statistical decision boundaries may be insufficient.

Overall, these findings indicate that LLMs do not universally outperform traditional machine learning approaches for SOC alert classification. Instead, the results suggest that lightweight machine learning baselines continue to provide strong practical value, while LLMs may offer complementary advantages in more context-dependent security analysis tasks.

6. Conclusions

This study examined the possibilities and limitations of using LLMs for alert classification and prioritisation in SOCs. Eight models were evaluated in terms of classification accuracy, prioritisation performance and operational feasibility. Classification performance across models varied, with most of them achieving a high recall for TPs of over 90 %, indicating strong threat detection capability. Often times, however, models also exhibited a significant FPR of over 50 %, indicating significant alert noise. One model - GPT-4.5 Preview in particular - demonstrated the overall most balanced performance, which allows to conclude that with regard to alert classification, LLMs can be used in SOC contexts, in combination with human oversight, to support alert triage. Besides that it was found that traditional ML models continue to provide strong practical value in regard to the binary classification task.

Prioritisation results showed inconsistent performance across severity levels, with prioritisation performance being weak universally. For low-priority alerts, a consistently high precision but low recall were exhibited, revealing the models' tendency to over-escalate low-priority alerts into higher-severity classes. Medium-priority alerts proved the most challenging, with uniformly low performance across all metrics - some models failed completely at identifying them. With regard to high-priority alerts, certain models achieved strong recall, detecting these alerts reasonably well, yet consistently low precision limited the trustworthiness of the respective predictions, introducing considerable alert noise. For critical alerts, low recall and precision were demonstrated, indicating overall unreliable performance for this important class. Considering the combined class of critical and high-priority alerts - the most important one in SOC contexts - a high recall and thus low false negative rate were achieved in most cases - indicating that only a small amount of high-severity threats were missed - yet still some high-severity threats were downgraded to lower severity, which is undesirable in SOC contexts.

Operationally, most models performed similarly and in a fashion suitable for close to real-time support in the alert triage process in real-world SOCs. Overall, the findings highlight that, while LLMs can generally support SOC analysts with initial alert triage - despite not being 100 % accurate - prioritisation of alerts in particular remains challenging. Future research should investigate the use of LLMs fine-tuned on cyber security data in this context, to explore to what extent this addresses the classification and prioritisation limitations observed with general-purpose models in this study. Furthermore, extending an LLMs knowledge base by allowing access to alert data surrounding the initial alert to be classified and prioritised, should be explored, investigating whether this improves a model's decision-making quality in such a SOC surrounding. Besides that, the performance of the respective LLMs on broader and more heterogeneous datasets should be explored and assessed to validate or contradict the observations from this study. In addition to that, future work should include a qualitative evaluation of the LLM-generated responses to determine their practical usefulness for SOC analysts in real-world settings. Finally, future research should examine the influence of prompt engineering in this context, as it might enhance classification and prioritisation outcomes.

CRediT authorship contribution statement

Matthias Rieger: Writing – original draft, Methodology; **Atif Shah:** Writing – review & editing; **Abu Alam:** Writing – review & editing; **Md. Jakir Hossain:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Supplementary material

Supplementary material associated with this article can be found in the online version at 10.1016/j.eswa.2026.133194.

References

Baruwal Chhetri, M., Tariq, S., Singh, R., Jalalvand, F., Paris, C., & Nepal, S. (2024). Towards human-AI teaming to mitigate alert fatigue in security operations centres. *ACM Transactions Internet Technology*, *24*(3). https://dl.acm.org/doi/10.1145/3670009. https://doi.org/10.1145/3670009

Canary, R. (2025a). 2025 Threat detection report. https://redcanary.com/threat-detection-report/.

Canary, R. (2025b). Top techniques. https://redcanary.com/threat-detection-report/techniques/.

Canary, R. (2025c). Top threats. https://redcanary.com/threat-detection-report/threats/.

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2024). A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, *15*(3). https://doi.org/10.1145/3641289

Ismail, Kurnia, R., Widyatama, F., Wibawa, I. M., Brata, Z. A., Ukasyah, Nelistiani, G. A., & Kim, H. (2025). Enhancing security operations center: Wazuh security event response with retrieval-augmented-generation-driven copilot. *Sensors*, *25*(3). https://doi.org/10.3390/s25030870

Jalalvand, F., Baruwal Chhetri, M., Nepal, S., & Paris, C. (2024). Alert prioritisation in security operations centres: A systematic survey on criteria and methods. *ACM Computing Surveys*, *57*(2). https://doi.org/10.1145/3695462

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, *55*(12). https://dl.acm.org/doi/10.1145/3571730. https://doi.org/10.1145/3571730

Khayat, M., Barka, E., Adel Serhani, M., Sallabi, F., Shuaib, K., & Khater, H. M. (2025). Empowering security operation center with artificial intelligence and machine learning – a systematic literature review. *IEEE Access*, *13*, 19162–19197. https://ieeexplore.ieee.org/document/10850912/?arnumber=10850912. https://doi.org/10.1109/ACCESS.2025.3532951

Knerler, K., Parker, I., & Zimmerman, C. (2022). 11 Strategies of a world-class cybersecurity operations center. MITRE. https://www.mitre.org/sites/default/files/2022-04/11-strategies-of-a-world-class-cybersecurity-operations-center.pdf.

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2025). Large language models: A survey. https://doi.org/10.48550/arXiv.2402.06196

MITRE (a). ATT&CK. https://attack.mitre.org/.

Olmo, T. (2026). Olmo 3. https://doi.org/10.48550/arXiv.2512.13961

Oniagbi, O. (2024). Evaluation of LLM agents for the SOC tier 1 analyst triage process. Master of Science Thesis University of Turku. https://www.utupub.fi/bitstream/handle/10024/178601/Oniagbi_Openime_Thesis.pdf?sequence=1.

OpenAI (2022). Introducing ChatGPT. https://openai.com/index/chatgpt/.

OpenAI (2023). Planning for AGI and beyond. https://openai.com/index/planning-for-agi-and-beyond/.

OpenAI (2024). GPT-4 technical report. https://doi.org/10.48550/arXiv.2303.08774

OpenAI (2025). Deprecations. https://platform.openai.com/docs/deprecations#2025_04_14_gpt_4_5_preview.

Rieger, M. (2025). LLM-SOC-alert-triage. https://github.com/c0deing/llm-soc-alert-triage.

Sai, S., Yashvardhan, U., Chamola, V., & Sikdar, B. (2024). Generative AI for cyber security: Analyzing the potential of chatgpt, DALL-E, and other models for enhancing the security space. *IEEE Access*, *12*, 53497–53516. https://ieeexplore.ieee.org/document/10491270. https://doi.org/10.1109/ACCESS.2024.3385107

Singh, Y., Patel, N. D., & Shandilya, S. K. (2024). Enhancing security operations center efficiency throughmulti-model integration of large language models and SIEMSystems. https://doi.org/10.21203/rs.3.rs-5615639/v1

Splunk (2025). State of security 2025. https://www.splunk.com/en_us/pdfs/gated/ebooks/state-of-security-2025.pdf.

Sundaramurthy, S. C., Bardas, A. G., Case, J., Ou, X., Wesch, M., McHugh, J., & Rajagopalan, S. R. (2015). A human capital model for mitigating security analyst burnout. In *Eleventh symposium on usable privacy and security (SOUPS 2015)* (pp. 347–359). https://www.usenix.org/conference/soups2015/proceedings/presentation/sundaramurthy.

Tariq, S., Baruwal Chhetri, M., Nepal, S., & Paris, C. (2025). Alert fatigue in security operations centres: research challenges and opportunities. *ACM Computing Surveys*, *57*(9). https://doi.org/10.1145/3723158

Tilbury, J., & Flowerday, S. (2024a). Automation Bias and complacency in security operation centers. *Computers*, *13*(7). https://www.mdpi.com/2073-431X/13/7/165. https://doi.org/10.3390/computers13070165

Tilbury, J., & Flowerday, S. (2024b). Humans and automation: Augmenting security operation centers. *Journal of Cybersecurity and Privacy*, *4*(3), 388–409. https://www.mdpi.com/2624-800X/4/3/20. https://doi.org/10.3390/jcp4030020

Vielberth, M., Böhm, F., Fichtinger, I., & Pernul, G. (2020). Security operations center: a systematic study and open challenges. *IEEE Access*, *8*, 227756–227779. https://ieeexplore.ieee.org/document/9296846. https://doi.org/10.1109/ACCESS.2020.3045514

Wazuh (b). Proof of concept guide. https://documentation.wazuh.com/current/proof-of-concept-guide/index.html.

Zhang, J., Bu, H., Wen, H., Liu, Y., Fei, H., Xi, R., Li, L., Yang, Y., Zhu, H., & Meng, D. (2025). When LLMs meet cybersecurity: A systematic literature review. *Cybersecurity*, *8*. https://doi.org/10.1186/s42400-025-00361-w

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., & Wen, J.-R. (2025). A survey of large language models. https://doi.org/10.48550/arXiv.2303.18223