

Let the Alerts Speak: LLM-Based IDS Alert Interpretation for SOC Triage

Alina Schärmer[1(✉)], Max Landauer[1], Florian Skopik[1], Markus Wurzenberger[1], and Marco Squarcina[2]

[1] Austrian Institute of Technology, Vienna, Austria
{alina.schaermer,max.landauer,florian.skopik,
markus.wurzenberger}@ait.ac.at
[2] TU Wien, Vienna, Austria
marco.squarcina@tuwien.ac.at

Abstract. Interpreting the large volume of alerts produced by Intrusion Detection Systems (IDS) each day can be challenging and fatiguing for Security Operation Center (SOC) analysts, especially when encountering numerous false positives. Although modern SOC tooling supports alert aggregation, correlation, and enrichment, it provides limited semantic interpretation of alert content and rarely offers human-understandable explanations to support triage decisions. We therefore present a concept that leverages Large Language Models (LLMs) to classify IDS alerts and assist SOC analysts during triage. Of particular interest is the ability of LLMs to distinguish attack-related alerts from false positives and to associate alerts with attack techniques. We perform an empirical evaluation of the alert classification performance of the LLMs ChatGPT and Gemini on alerts from network-based and host-based IDS. We further examine how system context information such as additional logs, configuration data, and few-shot examples impact the LLM's alert classification performance and consistency. Our results show that supplying LLMs with additional information, especially carefully chosen few-shot examples and system context, supports effective alert classification. At the same time, limitations related to cost, reliability, and sensitive data exposure could affect their practical use for alert interpretation.

Keywords: Large Language Models · IDS Triage · Alert Interpretation · SOC Analysis · Cyberdefense

1 Introduction

As cybersecurity attacks continue to rise, so does the need for stronger defenses. Effective defense relies on detection capabilities such as those offered by Intrusion Detection Systems (IDS). By monitoring networks, IT infrastructure, and user behavior, an IDS generates alerts in case of suspicious activity [28]. Security Operations Center (SOC) analysts must investigate these alerts using system context and threat intelligence. This process is resource-intensive and mentally

© The Author(s) 2027
Z. Afzal et al. (Eds.): ARES 2026 Workshops, LNCS 16901, pp. 346–364, 2027.
https://doi.org/10.1007/978-3-032-35579-9_19

demanding, particularly when analysts face large numbers of false positives [1], which can obscure genuine threats and cause human errors over time [27].

Automating parts of the alert analysis and pre-classification could reduce analyst workload and erroneous decisions. Moreover, potential explanations could accelerate alert triage and support in-depth analysis. Given the capabilities of Large Language Models (LLMs) for interpretation and reasoning [23], we propose a concept for interpreting IDS alerts using LLMs, focusing on ChatGPT and Gemini as representative examples. We evaluate these models ability to distinguish attack-related alerts from false positives and to associate alerts with MITRE ATT&CK technique IDs as a starting point for deeper analysis. Moreover, we investigate the impact of additional context, such as configuration data, system descriptions, supplementary logs, and few-shot examples, on classification performance. Throughout this study, we do not aim to replace SOC analysts, but present a proof-of-concept tool to support and augment their decision-making.

Our contributions can be summarized as follows:

- We evaluate LLMs' ability to distinguish attack-related alerts from false positives and to associate alerts with attack techniques.
- We assess the influence of contextual information such as configuration information, additional log information and classification examples on the LLM alert interpretation.
- We compare the interpretation performance across different alert types and identify trends of the LLM classifications.
- We discuss the feasibility of integrating this approach in the context of a SOC and identify potential challenges.

The remainder of this paper is structured as follows: Sect. 2 provides background on SOC analysis and related work on using LLMs to support analysts. Section 3 presents our LLM-based alert interpretation concept. Section 4 describes the experimental evaluation of LLM interpretation in a few-shot setting, which we extend upon by analyzing the impact of contextual information in Sect. 5. We discuss applicability in SOC environments in Sect. 6 and conclude in Sect. 7.

2 Background and Related Work

This section outlines the SOC analysis process and reviews prior work on applying LLMs in this context.

2.1 Intrusion Detection Systems (IDS) and SOC Workflow

Intrusion Detection Systems (IDS) identify threats to a system's confidentiality, integrity, or availability. Based on their detection principles, IDS are commonly categorized as signature-based (SIDS) or anomaly-based (AIDS). SIDS detect known attack patterns with high accuracy and low false positive rates but struggle with novel attacks and are sensitive to minor variations [10]. In contrast,

AIDS identify deviations from normal system behavior beyond defined thresholds, enabling the detection of unknown attacks at higher false alarm rates [4].

SOC teams detect, analyze, and respond to cybersecurity incidents, document findings, and implement preventive measures. This process is supported by tools such as IDS, which generate alerts that require human analysis [29]. To manage large alert volumes, SOCs commonly apply alert triage to categorize and prioritize alerts to focus analytical resources where they are most needed [11]. Many SOCs also follow a tiered structure, where Tier 1 analysts focus on rapidly responding to new alerts, filtering false positives, and escalating relevant alerts to Tier 2 analysts for in-depth investigation [29]. However, the high manual workload, combined with environmental factors such as repetitive tasks, has been shown to contribute to high burnout rates and erroneous decisions [27], particularly when analysts are confronted with high false positive rates [1].

2.2 Related Work

To address SOC analyst challenges, researchers have increasingly investigated applying LLMs across the incident response lifecycle. As surveyed by Habibzadeh et al. [6], while LLMs apply to every SOC phase, explainability and automation remain key hurdles. Our work directly targets these two challenges during the detection and analysis phases. Focusing on intrusion detection, Sun et al. [26] introduce SHIELD that leverages LLMs for host-based intrusion detection, whereas Nir et al. [21] and Jüttner et al. [9] apply LLMs to explain NIDS rules for non-experts. While these works emphasize explanation and interpretation, our work focuses on interpreting both HIDS and NIDS alerts to enable automated processing such as mapping to CTI. For multi-source SIEM environments, Du et al. [5] utilize zero-shot Chain-of-Thought prompting alongside the MITRE ATT&CK framework to aggregate multi-stage attacks, while Singh et al. [25] integrate an LLM ensemble with a Wazuh SIEM for alert classification and report generation. Unlike these approaches that require complex cross-source aggregation and SIEM infrastructures, our method prioritizes standalone IDS alert enrichment and we further evaluate how system-specific context and few-shot examples influence alert classification and mapping to the ATT&CK framework. Focussing on supporting alert triage, Singh et al. [24] demonstrate that analysts use LLMs primarily for alert understanding, context building and documentation. Finally, Nguyen et al. [20] evaluate applying open-source LLMs for Snort and Suricata alert analysis. While their approach emphasizes explanatory outputs such as alert descriptions and mitigations, their results show challenges in accurately identifying false positives. We address these performance limitations by injecting system-specific context and leveraging larger proprietary models.

3 Concept

This section describes our approach for using LLMs to interpret and classify IDS alerts. In SOC environments, IDSs process security-relevant events from computer networks and generate alerts for anomalous or suspicious behavior. These

alerts contain low-level technical indicators such as signatures and protocol fields that may indicate suspicious activity but are often insufficient to reliably assess malicious behavior or prioritize alerts. Analysts must therefore manually correlate alerts with system context and threat intelligence, a time-consuming process contributing to alert fatigue. Our pipeline addresses this challenge by combining alerts from AMiner [13], Suricata (https://suricata.io/), and Wazuh (https://wazuh.com/) with contextual system and threat information to produce semantically enriched alert representations supporting analyst decision-making. AMiner is a host-based anomaly detection IDS using statistical models on system and application logs. Suricata performs rule-based network traffic inspection using packet, protocol, and flow analysis. Wazuh is primarily host-based, combining signature and anomaly detection on endpoint logs and system events.

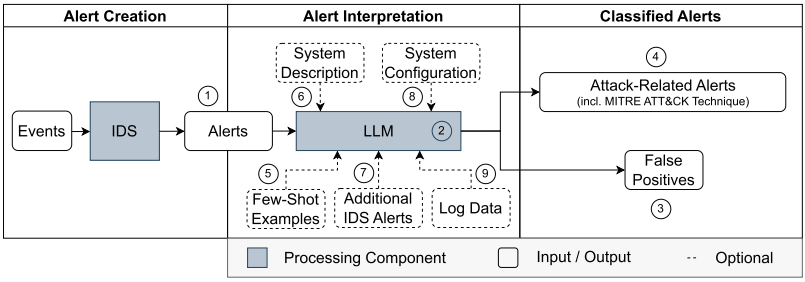


Fig. 1. Pipeline for IDS alert interpretation using LLMs.

As shown in Fig. 1, IDS process security-relevant events into alerts ① which are then passed to the LLM ②. The LLM classifies them into false positives ③ and attack-related alerts and maps relevant ATT&CK techniques ④ to explain observed behavior. This step abstracts low-level IDS outputs, which describe observable anomalies and signature matches, into a semantically meaningful representation of adversary tactics, techniques, and procedures (TTPs). By translating protocol-, payload-, and context-specific detections into these higher-level behavioral attacker patterns, this interpretation step enables automation such as alert grouping, attack path reconstruction, and threat intelligence enrichment.

Several intermediate cyber threat representations exist as described y Mavroeidis et al. [18], including CAPEC attack patterns, CWE software weakness enumeration, and STIX for standardized threat sharing. While these frameworks provide valuable descriptive and exchange capabilities, we select ATT&CK as our target representation due to its explicit focus on adversary behavior, hierarchical structure, and direct linkage to detection and mitigation strategies. Leveraging pre-trained language understanding, structured alert metadata, and additional inputs ⑤-⑨, the LLM performs semantic alignment between IDS alerts and ATT&CK techniques by identifying contextual relationships and behavioral patterns, producing structured JSON output suitable for automated processing. Importantly, the pipeline serves as a decision-support tool, assisting

analysts in prioritizing and correlating alerts while maintaining human oversight. This analyst-in-the-loop focus motivates the use of LLMs over traditional machine learning approaches such as SVMs, which are primarily optimized for classification under fixed metrics. Rather than focusing solely on predictive accuracy, our approach emphasizes interpretation and contextualization through natural-language reasoning. Accordingly, TP/FP assessments and ATT&CK mappings are treated as reasoning artifacts whose reliability can be evaluated.

4 LLM-Based IDS Alert Interpretation

This section describes the dataset and methodology for interpreting IDS alerts with LLMs in a few-shot setting (Fig. 1 ⑤) and presents the evaluation results.

4.1 Dataset

For the evaluation we use the IDS alert dataset by Landauer et al. [14,15], generated in a simulated enterprise network containing services such as mail, file sharing and WordPress servers. The dataset contains alerts from the IDS AMiner, Suricata, and Wazuh under both normal user activity and a multi-step attack scenario. Simulated attacks include data exfiltration, various scans, a web shell upload, password cracking, and privilege escalation, among others [12].

The alerts are labeled time- and event-based. Time-based labels indicate whether an alert occurred during the simulation window of an attack, while event-based labels are assigned based on the alert-triggering log line. Although time-based labels are less accurate, as they may classify unrelated alerts within an attack window as malicious, we use them because event-based labels are only available for a subset of attack types, limiting comparability across attacks.

The dataset includes multiple variations differing in parameters such as IP addresses, hostnames, and user behavior. We randomly select the alerts from the *Fox* variation. Since Wazuh and Suricata alerts are stored in a shared file, we treat them as a combined source and refer to them collectively as the Wazuh dataset. The experimental dataset comprises 100 attack-related and 100 benign alerts. It includes all available attack types while approximately preserving their relative distribution, though some attack types are not represented in every IDS source as shown in Fig. 7. The reduced evaluation dataset size allows for deeper inspection and controlled comparisons, while remaining sufficient to reveal systematic trends. Additionally, because each evaluation requires multiple high-token LLM invocations, limiting dataset size ensures experimental feasibility and reproducibility under realistic computational and cost constraints, without substantially affecting interpretive insights.

4.2 Evaluation Methodology

In our experiments, we use the APIs of ChatGPT 4o (Aug. 6, 2024 release) and Gemini 2.5 Flash Lite to automate alert interpretation. ChatGPT 4o was selected

as the newest available model at experiment time, while Gemini 2.5 Flash Lite was preferred over newer variants due to higher availability. We choose these proprietary models over open-source models because their managed APIs reduce operational overhead and fine-tuning requirements, supporting rapid experiment setup and enterprise integration. Weak alert classification results for open-source models such as Mistral-7B, Qwen3-8B, and LLaMA3.1-8B shown by Nguyen et al. [20] further motivates the use of these large general-purpose models. Experiments are repeated three times for every dataset-LLM combination to account for LLM output nondeterminism as described by Ouyang et al. [22]. We set temperature to 0 and top-p to 1 to maximize consistency while preserving expressive outputs, and limit responses to 1000 to control costs for complex alerts.

```
You are a security analyst. You will be given an IDS (Intrusion Detection
System) log entry in JSON format called ‘ids_line‘. Your task is to classify the
alert and return the output in the following JSON schema:
{
    "anomaly_timestamp": "<value from Timestamps in ids_line>",
    "classification": "TP | FP",
    "mitre_technique": ["<techniques from MITRE ATT&CK>"],
    "description": "<short reasoning about the classification>"
}
Follow these rules:
- Extract the anomaly timestamp from ’Timestamps’.
- Always output valid JSON only.
- Classification:
- TP = true positive (legitimate malicious activity)
- FP = false positive (benign or misclassified activity)
- For mitre_technique, use the best fitting MITRE ATT&CK techniques, you can
also supply more than one but stick to the minimum.
- Use short but precise description

<Additional information (few-shot examples,...)>
```

Fig. 2. Prompt template including output format and task instructions.

Given this setting, we provide the LLMs with zero-shot, one-shot and two-shot prompts. Few-shot prompts include a task description together with up to two example alert-output pairs, while zero-shot prompts contain only the task description and alerts, serving as a baseline. The task requires the model to return the alert timestamp, whetzer the alert is attack-related, a minimal set of MITRE ATT&CK techniques, and an explanation of its reasoning as shown in Fig. 2. Few-shot examples are included in the system prompt, following recommendations by Loukas et al. [17]. We select example alerts from dataset variants different from the evaluation set, with up to two alerts per attack type and up to two no-attack alerts for each IDS. Two-shot prompts extend one-shot prompts with an additional example per attack type and an additional benign alert chosen to maximize diversity from the first example. Few-shot prompting reduces task ambiguity but requires maintaining representative examples as threats evolve and has been shown to be less effective than fine-tuned models [3].

We analyze the LLM outputs by evaluating alert categorization (false positive vs. attack-related) and the correctness of assigned ATT&CK techniques. To assess technique assignments, we map the ATT&CK framework to the dataset attack types based on the dataset descriptions [14]. Assignments are classified as correct if all predicted techniques match the mapping, partially correct if at least one matches, and incorrect otherwise. Thus, partially correct classifications may include unrelated techniques but still contain useful ones. Although this definition could label exhaustive technique lists as partially correct, our prompt requests a minimal set, and hence outputs typically contain at most three techniques.

4.3 Results

Our results show that the analyzed LLMs classify no-attack alerts more accurately than attack-related alerts, as shown in Table 1. This gap is most pronounced in the zero-shot setting, where only about one-third of attack-related alerts are classified correctly, compared to over 90% of no-attack alerts. Introducing one- and two-shot examples approximately doubles the detection rate for attack-related alerts, while only slightly reducing performance on no-attack alerts, resulting in substantially improved overall classification performance.

Table 1. Average success rate of alert categorization for few-shot experiments

Experiment	Attack	No-attack	ChatGPT	Gemini	Total
Zero-Shot	31.00%	92.08%	67.08%	56.00%	61.54%
One-Shot	74.00%	84.83%	76.75%	82.08%	79.42%
Two-Shot	75.67%	82.92%	79.83%	78.75%	79.29%

To further compare the experimental variants, we report the average accuracy, precision, F1-score, TP-rate, and FP-rate across runs of each experiment type and rank them in Fig. 3. The figure also includes attack type classification performance for attack-related alerts, expressed as the ratio of correct and partially correct classifications to the total number of classifications. Overall, the results show that one- and two-shot prompting substantially improve the classification into attack and no-attack alerts compared to zero-shot prompting. This is reflected by the accuracy that increases from 51–68% in the zero-shot setting to 75–83.5%, while TP-rates improve from 7–61% to 49–91%. TP-rates are generally higher for AMiner alerts, likely due to their lower alert diversity. In contrast, Wazuh alerts achieve consistently higher precision values (over 94%) than AMiner alerts (51–79%), resulting in substantially lower false positive rates (0–4% to 5–40%). This indicates that the LLMs classify Wazuh alerts more reliably.

Regarding correctness of MITRE ATT&CK technique assignments, results clearly show the benefit of classification examples. The two-shot setting achieves

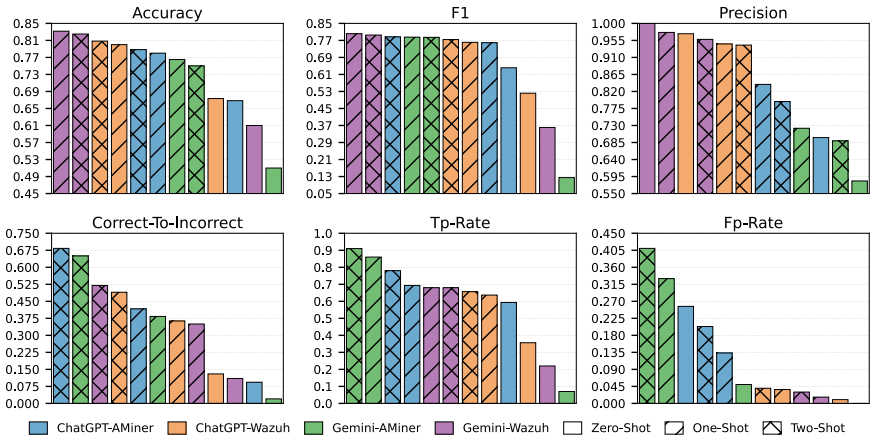


Fig. 3. Comparison of different few-shot experiments.

correct-to-incorrect ratios of approximately 50–70%, compared to 35–40% in the one-shot setting, which is more than three times higher than those of the zero-shot setting. As with attack detection, AMiner alerts achieve higher classification accuracy than Wazuh alerts, likely due to greater diversity of Wazuh alerts.

Overall, results show that few-shot examples substantially improve both the identification of attack-related alerts and the correctness of attack type classification, albeit at the cost of a slight reduction in false positive detectability.

5 Influence of Contextual Information

This chapter evaluates the impact of various context information, such as system descriptions, configurations and additional alerts and log data (Fig. 1 ⑥-⑨), on the LLMs classification performance and presents the corresponding results.

5.1 Evaluation Methodology

To evaluate the impact of contextual information, we apply the same methodology described in Sect. 4, but replace the few-shot examples with the respective context sources. These sources include full-text and keyword-based system descriptions, as well as device configuration data such as host names, IPv4 addresses, time zones, operating systems, and group affiliations describing the purpose of server and user devices. For server devices, we additionally provide the fully qualified domain name, while for host devices we include the username.

Furthermore, we evaluate the impact of additional log lines from the same source that triggered the alert, as well as preceding alerts related to the analyzed alert. To identify trends based on the amount of contextual information, we vary the number of supplied data items across experiments. In a first approach, the number of additional inputs is increased over 20 runs. If fewer items are available

in the respective data source, all available inputs are used. In addition to this simple approach, we conduct a time-based experiment with 10 runs, in which contextual data must fall within time windows ranging from 5 to 50 s before the analyzed alert. To avoid excessive prompt sizes, the number of additional inputs is limited to 20. For the time-based alert experiment, requests are only sent to the LLM if additional alerts are available, otherwise the setup would be equivalent to the zero-shot experiment.

5.2 Results of Configuration and System Description Experiments

Similar to the few-shot examples when providing the LLM with configuration data, system descriptions, and their combinations false positives are consistently easier to identify than attack-related alerts, as shown in Table 2. Across both LLMs and datasets, server configuration information has the strongest positive effect on identifying attack-alerts and false positives. Moreover, keyword-based descriptions improve attack-related alert interpretation, whereas full-text descriptions perform better for false positive classification. In general, a single context source outperforms combining multiple sources, suggesting that excessive context introduces noise rather than improving interpretability. Overall, providing context improves attack-alert identification over the zero-shot baseline while maintaining strong false positive detection. Combining context with few-shot examples further improves performance beyond either approach alone. While the two-shot setting achieves the best overall attack-alert classification, the one-shot setting provides more balanced detection and the best false positive results.

Table 2. Average success rate of alert categorization for context experiments

Experiment	Attack	No-attack	ChatGPT	Gemini	Total
Server Config	40.33%	92.58%	70.33%	62.25%	66.30%
User Config	39.42%	91.58%	70.75%	60.25%	65.50%
Textual Description	36.17%	89.92%	68.08%	58.00%	63.04%
Keyword Description	39.83%	88.08%	68.33%	59.50%	63.92%
Combination	37.92%	91.08%	70.25%	58.75%	64.50%
Combination with One-Shot	62.00%	95.33%	79.58%	77.92%	78.67%
Combination with Two-Shot	72.83%	86.42%	81.42%	77.83%	79.63%

These trends are reflected in the metric values shown in Fig. 4. In context-only settings, accuracy ranges between 58–75%, while F1-scores improve by roughly 20% over the zero-shot baseline. Although TP-rates vary substantially across experiments (20–71%), precision remains high (64–100%), suggesting that contextual prompting mainly improves classification by reducing false alarms. Wazuh additionally shows a broader precision range than AMiner (64–100%

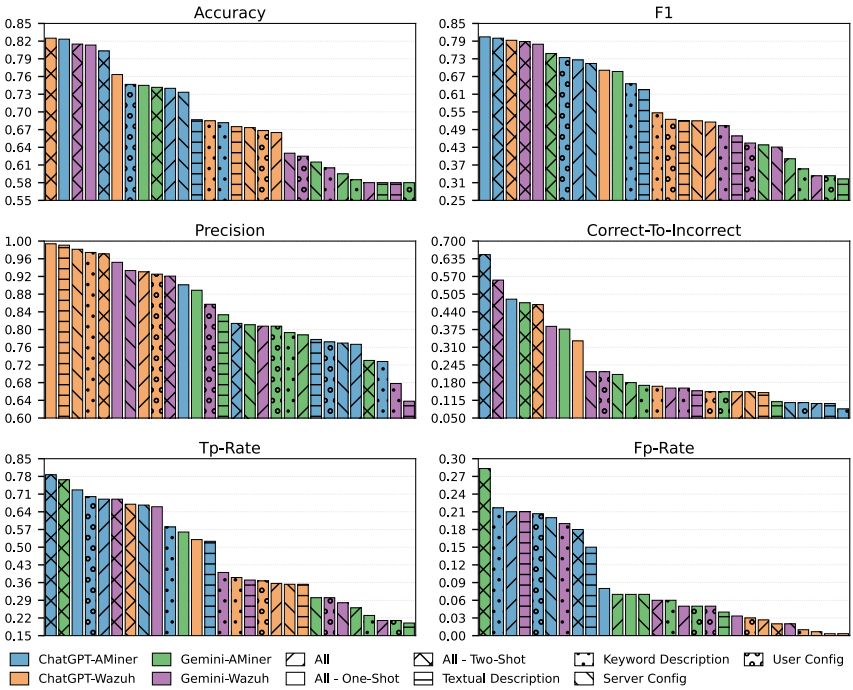


Fig. 4. Comparison of different context experiments.

vs. 77–90%), indicating greater sensitivity to the prompting strategy. The best overall performance is achieved by combining context with few-shot examples, increasing accuracy to 74–84%. Compared to plain few-shot prompting, TP-rates and F1-scores become more consistent across setups, while FP-rates decrease from up to 40% to at most 28% which Wazuh having a consistently lower rate than AMiner alerts. This indicates improved robustness against FP misclassifications by using contextual information. For attack type classification, context has mixed effects. In the few-shot setting, adding context slightly reduces the correct-to-incorrect ratio (33–66% vs. 35–70%) while it provides clear benefits in the zero-shot setting, improving the correct-to-incorrect ratio from 2–14% to 10–22%.

5.3 Results of Simple Alert and Log Line Experiments

When increasing the number of supplementary alerts or log lines with each run, we observe that adding context primarily improves performance within the first 5–9 additions, after which results stabilize and may slightly decline once around 15 sources are included (Fig. 5). This indicates diminishing returns, where a small amount of additional evidence supports interpretation, while too many sources introduce noise. Across both datasets, additional alerts are on average more beneficial than log lines, resulting in slightly higher accuracy (70.81% vs. 65.43%)

and TP-rates (up to 84% vs. 78%). In contrast, log-line experiments maintain slightly higher precision (around 79% vs. 77%) and lower FP-rates (up to 17% vs. 47%), making them less prone to overly confident positive predictions.

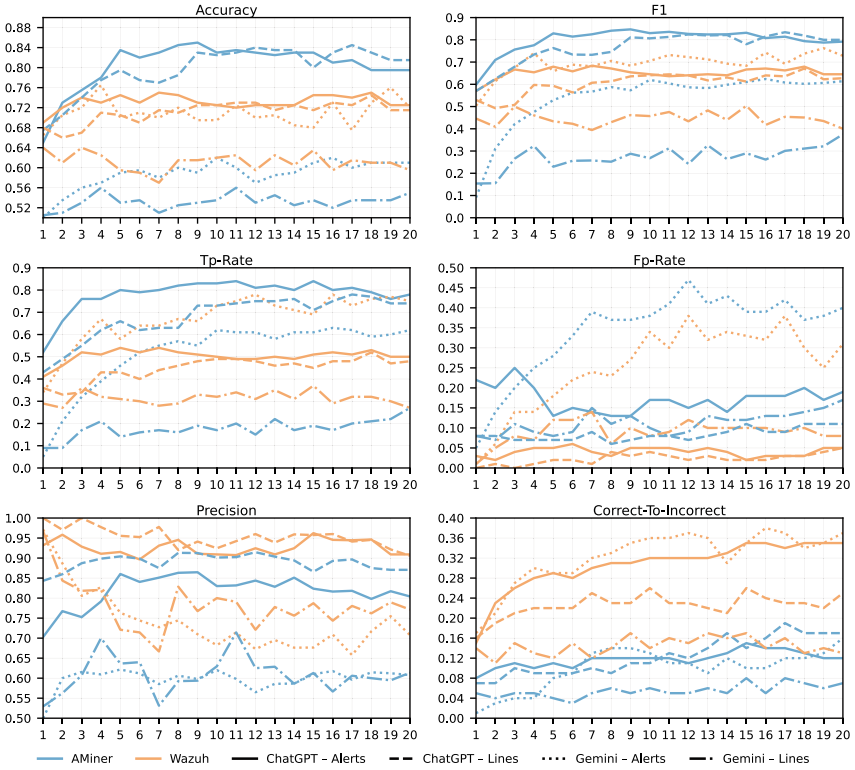


Fig. 5. Comparison of additional alert and log data experiments in simple setting.

Compared to few-shot prompting, supplementary sources improve classification performance over the zero-shot setting but do not reach the performance of the one- and two-shot variants. Across datasets, Wazuh alerts are generally classified with higher precision than AMiner alerts (67–100% vs. 53–91%) and tend to produce lower FP-rates, particularly when adding alerts. Overall, we find that AMiner alerts benefit more strongly from the first additions but also show clearer performance degradation when adding too many sources. For attack type classification, Wazuh alerts achieve higher correct-to-incorrect ratios than AMiner alerts (11–39% vs. 1–19%). Moreover, additional alerts have a greater impact on the classification performance compared to log lines. Both approaches outperform the zero-shot baseline, although absolute performance remains limited.

5.4 Results of Time-Based Alert and Log Line Experiments

The time-based setting produces more stable results across runs than the simple setting while maintaining a comparable overall performance, as shown in Fig. 6. Accuracy remains within a similar range as in the simple setting (42–82% vs. 50–85%), suggesting that additional context from a specific time frame improves consistency rather than detection capability. We also observe model- and IDS-specific effects. ChatGPT performs similarly on alert- and log-based additional contexts and generally handles AMiner better than Wazuh alerts, whereas Gemini achieves better results on alert-based than log-based experiments and interprets Wazuh alerts more accurately than AMiner alerts. Compared to the simple setting, the time-based runs slightly improve precision (58.8–100% vs. 50–100%) and reduces false-positive rates (0–22% vs. 0–47%). In general, Wazuh are interpreted more precisely than AMiner alerts, while TP-rates and F1-scores remain comparable across the experiment settings. Attack-type classification performance also remains similar, although with a narrower performance range.

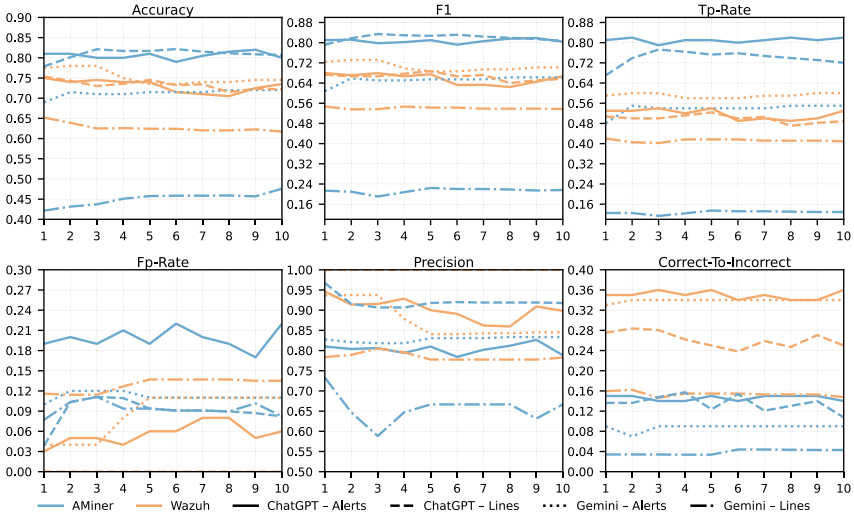


Fig. 6. Comparison of additional alert and log data experiments in time-based setting.

6 Feasibility of Integration in the SOC Context

This section summarizes the results, explores explanations for observed trends, discusses integration into SOC workflows and associated challenges, and examines result consistency, reproducibility, and limitations.

6.1 Factors Influencing LLM Classification Performance

Overall, zero-shot experiments achieve modest performance for attack alerts (31%) and attack types (∼9%), but good results for no-attack alerts (92%). Few-shot examples substantially improve attack alert (∼75%) and attack type classification (35–70%), while slightly reducing no-attack results (84%). Adding context, such as configuration data and system descriptions, mitigates no-attack performance degradation to near zero-shot levels, with a slight decrease in attack alert classification. The two-shot context setting achieves the best overall attack-alert performance (∼73%), whereas one-shot with context offers slightly lower attack-alert (∼62%) but better no-attack results (∼95%) at lower token cost, making it preferable. Even without few-shot examples, context improves classification performance over the zero-shot baseline (62% vs. 79%). Including additional log lines or alerts yields similar improvements, with time-based producing more stable results than simple settings. Across experiments, both LLMs consistently classify no-attack alerts more accurately than attack alerts, and binary attack/no-attack classification proves more reliable than attack-type classification. Wazuh alerts show higher precision and fewer false positives, whereas AMiner alerts achieve higher TP-rates, accuracy, and correct attack types. To further analyze these effects, we examine individual alerts. Figure 7 presents a heatmap of attack-type classifications by IDS and attack type across experiments, comparing ChatGPT (first rows) with Gemini (second rows).

Influence of Few-Shot Examples. LLM predictions closely follow the few-shot examples. This is most evident for AMiner Dirb alerts and Wazuh Suricata alerts, which closely resemble the examples and thus achieve strong classification. In contrast, alerts that differ substantially from the examples perform worse, especially in the one-shot setting, indicating overfitting. This suggests that few-shot examples implicitly convey attack tactics and constrain the considered techniques. No-attack false positives and DNSteal alerts are largely unaffected.

Degrading Performance on Long Inputs. Additional log line, alert, and combined context experiments show degrading performance with increasing input length, consistent with findings by Modarressi et al. [19] linking long prompts to reduced LLM output quality. While excluding low-semantic fields such as IDs may mitigate this issue, our results emphasize the importance of carefully selected few-shot examples, especially when combined with contextual information.

Training Data Availability. The heatmap shows that Wazuh alerts outperform AMiner alerts, likely because LLMs are more extensively trained on Wazuh and Suricata data. Moreover, AMiner's anomaly-based alerts provide more limited context, as they are frequently expressed as high-level indicators like frequency-based deviations (e.g. *CPU value deviates from average in monitoring logs.*).

Relevance Indicators. High-value relevance indicators (e.g. confidence scores), improve classification of anomaly-based and protocol violation alerts (e.g., *SURICATA TLS invalid record/traffic*). Aggregated alerts (e.g., *Multiple web server*

400 error codes from same source IP.) are also classified more accurately, showing that repeated occurrences aid LLM classification. Special URL characters (e.g., $+!*) further increase the likelihood of an alert being classified as malicious.

Tool Names. Tool names (e.g. *Nmap Scripting Engine*, *WPScan v3.8.20*) strongly influences classifications e.g. for the scan and WPScan attack types as shown by the heatmap. Similarly, privilege escalation alerts with root-level activity (e.g., *Successful sudo to ROOT executed*) are reliably identified.

Time-Based Labeling. Both heatmaps show that many attack-labeled alerts (e.g. DNSteal and *dovecot login*) are classified as benign by the LLMs. This occurs because their descriptions (e.g., *dnsmasq[14460]: query[A] youtu.be*) lack malicious indicators but coincide with attack simulation windows, causing our methodology to count correct classifications as errors. Conversely, some alerts labeled benign by time-based labeling contain malicious activity (e.g., DNSteal activity in AMiner alerts) and are classified accordingly. Future work should use event-based labeling and exclude unrelated alerts within attack windows.

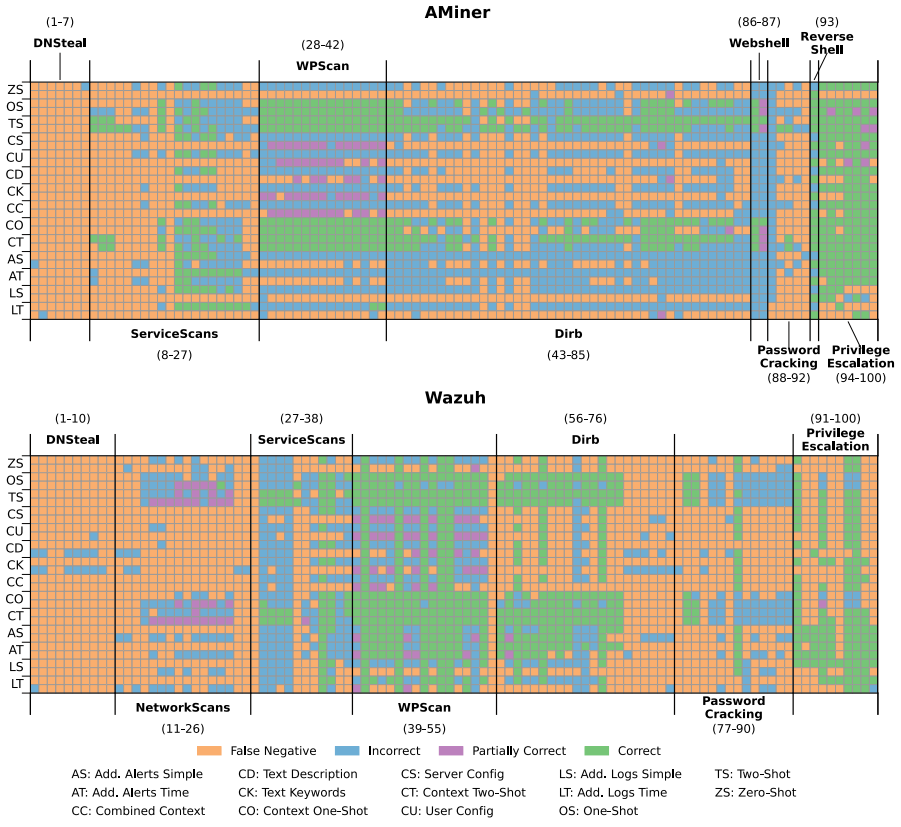


Fig. 7. Distribution of attack type classification correctness for AMiner and Wazuh.

6.2 Use Case: Automatic Alert-to-CTI Mapping

To enhance SOC workflows, our approach can be integrated into a larger pipeline beyond the initial IDS alert classification. Because our method already maps alerts to MITRE ATT&CK techniques, it enables easy Cyber Threat Intelligence (CTI) enrichment to accelerate investigations. As illustrated in Fig. 8, CTI extraction tools such as AttacKG by Li et al. [16] or TTPDrill by Husari et al. [8] align CTI reports with ATT&CK IDs, either directly or after preprocessing. With both alerts and CTI sources represented as ATT&CK IDs, querying alert IDs within indexed CTI data efficiently matches them to accelerate triage.

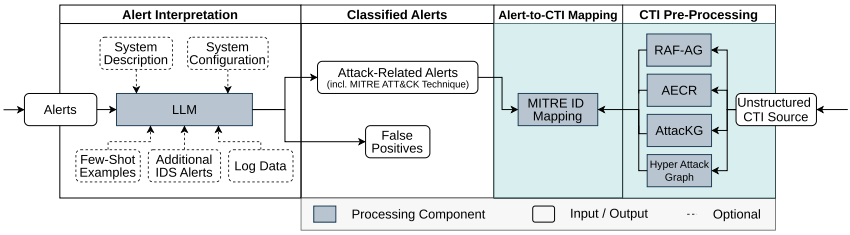


Fig. 8. Alert-to-CTI matching pipeline based on LLM interpretation

Despite its effectiveness, several limitations affect the approach's applicability in the SOC context. First, token costs and API rate limits may limit the economic feasibility of cloud-based LLMs in high-volume environments. In our experiments, ChatGPT incurred total token costs of approx. 400, while Gemini costs could not be determined due to the use of a trial version. Third-party data sharing also raises privacy and security concerns regarding sensitive system information. However, deploying self-hosted LLMs can mitigate these infrastructure and data exposure issues. Additionally, the evolving threat landscape requires ongoing maintenance, including updated model versions, performance re-evaluation, and current few-shot examples. Finally, API processing failures as observed with Gemini can disrupt automation, and limited explainability reduces suitability for auditing or compliance. However, because the framework supports rather than replaces SOC analysts, this lack of explainability is less critical.

6.3 Result Consistency

To assess reproducibility, we evaluate LLM interpretations consistency by repeating the zero-shot and one-shot experiments on attack-related alerts over 40 runs, except for the one-shot Wazuh setting, which was limited to 25 runs due to Gemini processing errors. Focusing on attack-related alerts reduces token costs while preserving the data for alert categorization and attack type classification. As in Sect. 4.2, each run produces TP/FN labels and, when applicable, attack-type assignments. We assess consistency over two majority states, namely a simple

state (TP/FN) and an extended state that including attack-type correctness (e.g., tp:68, fn:32, cc:30, pc:6, ic:32). Assuming statistical independence across conversation contexts of the LLM requests, we model consistency as a Bernoulli process, where runs matching the majority state represent successful trials. From these, we compute 95% Wilson confidence intervals [2], illustrated alongside their observed proportions in Fig. 9.

The figure shows that Gemini exhibits consistently high reproducibility, with 95% confidence intervals spanning 68–100% across all variants. ChatGPT is more variable, intervals ranging from 6–77%, and shows higher consistency for binary than for attack type classification. Few-shot examples improve ChatGPT's consistency in the simple but degrade it in the extended state. For Gemini, they have a slight negative effect, widening the interval for Wazuh alerts. Overall, Wazuh runs remain more consistent than AMiner runs for both LLMs, likely due to greater training data representation. Ultimately, these variations align with trends seen in the other experiments and known sources of LLM nondeterminism such as GPU concurrency and batch invariance failures [7].

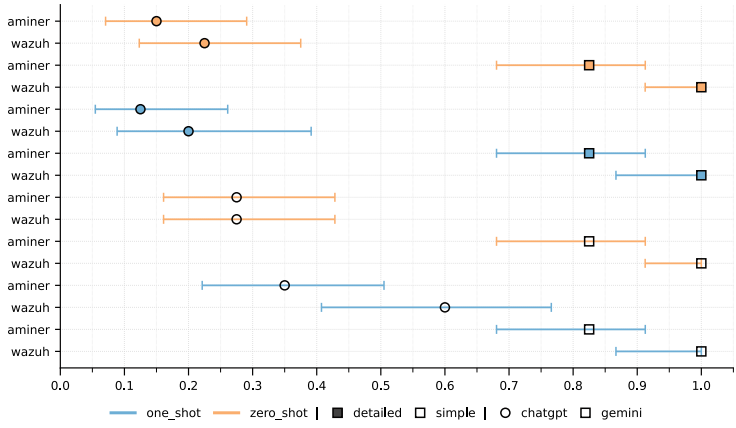


Fig. 9. Wilson Confidence Intervals.

6.4 Limitations

Several factors condition the broader generalizability of our findings. First, the small sample and evaluation dataset sizes necessitate cautious interpretation. Time-based labeling also introduces misclassifications for alerts incorrectly falling into attack simulation windows. Furthermore, since few-shot examples were derived from the same attack execution types as the interpreted alerts, performance may not generalize to other tools, environments, or attack variants. Finally, despite deterministic configurations, reproducibility remains subject to minor variability quantified using Wilson confidence intervals.

7 Conclusion

This work evaluates an approach for interpreting and classifying IDS alerts using the LLMs ChatGPT and Gemini, specifically focusing on effective distinguishing of attack-related alerts from false positives and assigning MITRE ATT&CK technique IDs. We systematically analyze how different prompt additions such as few-shot examples and various contextual information (additional logs, system configurations, textual descriptions) impact classification performance. Results show that a zero-shot approach is insufficient, achieving only 31% accuracy for attack-related alert identification and 9% for attack-type classification. Incorporating few-shot examples substantially improves these metrics to 75% and 3370%, respectively, though at the expense of no-attack accuracy which drops from 92% to 84%. Contextual data mitigates this false-positive degradation. Balancing performance, attack-type accuracy, and token costs, the one-shot variant with context emerges as the most practical setup for operational SOC environments. Beyond these quantitative results, our findings show that LLMs are effective for semantic alerts (e.g., including tool names) but unreliable for statistical or anomaly-based alerts, especially in the absence of confidence scores. Finally, real-world deployment of the approach remains bounded by operational challenges, including high token costs, API availability constraints, data privacy risks, limited explainability, and the need for continuous maintenance of few-shot examples and prompts as new attack types and LLM model versions emerge.

Acknowledgments. Funded by the European Union under the Horizon Europe Research and Innovation programme (GA no. 101168144 - MIRANDA) and under the European Defence Fund (GA no. 101121403 - NEWSROOM, GA no. 101168092 - ECYSAP EYE, and GA no. 101121418 - EUCINF). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them. Co-funded by the Austrian FFG Kiras project ASOC (FO999905301).

Data Availability. All code, data, and figures are publicly available on GitHub to reproduce the results.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alahmadi, B.A., et al.: 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms. In: USENIX Security 22, pp. 2783–2800 (2022)
2. Brown, L.D., et al.: Interval estimation for a binomial proportion. Stat. Sci. 101–133 (2001)
3. Brown, T.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems, pp. 1877–1901 (2020)

4. Butun, I., et al.: A survey of intrusion detection systems in wireless sensor networks. IEEE Commun. Surv. Tutorials 266–282 (2013)
5. Du, D.: Mad-llm: a novel approach for alert-based multi-stage attack detection via LLM. In: 2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA), pp. 2046–2053 (2024)
6. Habibzadeh, A.: Large language models for security operations centers: A comprehensive survey. arXiv preprint arXiv:2509.10858 (2025)
7. He, H.: Defeating nondeterminism in llm inference. In: Thinking Machines Lab: Connectionism, (2025). https://doi.org/10.64434/tml.20250910
8. Husari, G., et al.: Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In: ACSAC 2017, pp. 103–115 (2017)
9. Jüttner, V., et al.: Chatids: Advancing explainable cybersecurity using generative ai. Int. J. Adv. Secur. 2 (2024)
10. Khraisat, A., et al.: Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 1–22 (2019)
11. Killcrece, G., et al.: Organizational models for computer security incident response teams (csirts). Carnegie Mellon University, SEI Handbook (2003)
12. Landauer, M.: Have it your way: Generating customized log datasets with a model-driven simulation testbed. IEEE Trans. Reliab. 402–415 (2021)
13. Landauer, M.: Aminer: A modular log data analysis pipeline for anomaly-based intrusion detection. In: Digital Threats: Research and Practice, pp. 1–16 (2023)
14. Landauer, M.: Maintainable log datasets for evaluation of intrusion detection systems. IEEE TDSC 3466–3482 (2023)
15. Landauer, M.: Introducing a new alert data set for multi-step attack analysis. In: Cyber Security Experimentation and Test Workshop, pp. 41–53 (2024)
16. Li, Z., Zeng, J., Chen, Y., Liang, Z.: AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) Computer Security – ESORICS 2022. ESORICS 2022. Lecture Notes in Computer Science, vol. 13554. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17140-6_29
17. Loukas, L.: Breaking the bank with chatgpt: Few-shot text classification for finance. arXiv preprint arXiv:2308.14634 (2023)
18. Mavroeidis, V., Bromander, S.: Cyber threat intelligence model: an evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence. In: EISIC 2017, pp. 91–98. IEEE (2017)
19. Modarressi, A.: Nolima: Long-context evaluation beyond literal matching. arXiv preprint arXiv:2502.05167 (2025)
20. Nguyen, L., Nguyen, MD., Bui, A.H., Cavalli, A.R., de Oca, E.M.: LLMs in Security Testing and Monitoring: An Initial Study. In: Skopik, F., Naessens, V., De Sutter, B. (eds.) Availability, Reliability and Security. ARES 2025. Lecture Notes in Computer Science, vol 15999. Springer, Cham (2025). https://doi.org/10.1007/978-3-032-00644-8_9
21. Nir, D., et al.: Labeling network intrusion detection system (nids) rules with mitre att&ck techniques: Machine learning vs. large language models. BDCC 23 (2025)
22. Ouyang, S.: An empirical study of the non-determinism of chatgpt in code generation. ACM TOSEM 1–28 (2025)
23. Raiaan, M.A.K.: A review on large language models: Architectures, applications, taxonomies, open issues and challenges. IEEE Access 26839–26874 (2024)
24. Singh, R.: Llms in the soc: An empirical study of human-ai collaboration in security operations centres. arXiv preprint arXiv:2508.18947 (2025)

25. Singh, Y.: Enhancing security operations center efficiency through multi-model integration of large language models and siem systems (2024)
26. Sun, D.: From alerts to intelligence: A novel llm-aided framework for host-based intrusion detection. arXiv preprint arXiv:2507.10873 (2025)
27. Sundaramurthy, S.C.: A human capital model for mitigating security analyst burnout. In: SOUPS 2015, pp. 347–359 (2015)
28. Yang, W., Lam, K.-Y.: Automated Cyber Threat Intelligence Reports Classification for Early Warning of Cyber Attacks in Next Generation SOC. In: Zhou, J., Luo, X., Shen, Q., Xu, Z. (eds.) ICICS 2019. LNCS, vol. 11999, pp. 145–164. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-41579-2_9
29. Zimmerman, C.: Ten strategies of a world-class cybersecurity operations center. mitre 2014 (2022). https://www.mitre.org/sites/default/files/publications/pr-13-1028-mitre-10-strategies-cyberops-center.pdf

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

