SOCRATES: An AI-Powered SOC Agent for Automated Network Alert Triage

Abstract

Modern Security Operations Centers (SOCs) are increasingly overwhelmed by a deluge of network security alerts, the vast majority of which are false positives. This phenomenon, known as alert fatigue, severely degrades the efficiency of security analysts and increases the risk of overlooking genuine threats. While Large Language Models (LLMs) have shown promise in semantic understanding, their direct application to real-world alert triage is hindered by prohibitive inference latency and high false positive rates due to a lack of enterprise-specific context. In this paper, we propose SOCRATES, an AI-powered SOC agent designed for automated and scalable network alert triage. SOCRATES employs a staged, coarse-to-fine processing pipeline that integrates three core components: (1) a lightweight aggregation and filtering module to consolidate redundant alerts, (2) a business logic self-learning module that leverages XGBoost to suppress recurring benign activities, and (3) a context-enhanced LLM investigation module that performs evidence-grounded reasoning by retrieving internal asset metadata and external threat intelligence. Experimental results demonstrate that SOCRATES achieves a significant workload reduction ratio of 95.22% while maintaining high detection integrity, retaining 99.63% of true attacks with an F1-score of 82.48%. Furthermore, real-world deployment over one year confirms that SOCRATES reduces manual investigation effort by over 80%, providing a scalable, interpretable, and effective solution for modern security operations.

1 Introduction

With the increasing sophistication and automation of cyber-attacks, modern enterprises are facing an ever-growing and increasingly complex threat landscape. To effectively detect and mitigate malicious activities, organizations have deployed a wide range of security devices, including Web Application Firewalls (WAFs) [25], Intrusion Detection Systems (IDSs) [6], and Intrusion Prevention Systems (IPSs) [22]. While these security devices significantly enhance defensive visibility, they paradoxically give rise to a severe problem known as **alert fatigue** [5, 13, 15, 33]. In large-scale enterprise environments, they routinely generate millions of alerts on a daily basis, far exceeding the processing capacity of human analysts, with over 99% of these alerts ultimately identified as false positives [1, 3, 41]. To alleviate this burden, industry practices have extensively adopted heuristic-based detection techniques as well as traditional machine learning and deep learning paradigms [4, 34, 36]. However, these approaches remain constrained by inherent limitations [31]. Rule-based systems are rigid and costly to maintain, requiring continuous manual updates to keep pace with evolving attack patterns [38]. Data-driven learning models, on the other hand, often operate as black boxes, lacking the interpretability necessary for analysts to trust their decisions. Moreover, such models typically fail to capture the deep semantic structure of alert logs or to correlate attacker behaviors across contextual dimensions, resulting in missed detections of sophisticated and stealthy threats.

Recently, Large Language Models (LLMs) have demonstrated remarkable capabilities in semantic understanding and contextual reasoning [14, 17], enabling them to interpret complex data with a level of flexibility previously unattainable by traditional learning models. Their ability to process heterogeneous inputs and reason over long textual contexts makes them especially attractive for security analysis tasks [28], where alerts often contain rich but fragmented semantic information distributed across logs, payloads, and metadata. Beyond general natural language tasks, LLMs have already shown promising results across multiple cybersecurity domains [40], such as automated penetration testing [8, 30], vulnerability discovery and analysis [19, 24, 42], and security policy auditing. LLMs have been increasingly explored as a potential paradigm for automating security workflows and bridging the gap between low-level alert data and high-level human reasoning.

Research Gaps. Despite the significant potential of LLMs, directly applying them to real-world alert classification and

triage scenarios remains challenging [39]. Through empirical investigation and practical consideration of industrial Security Operations Centers (SOCs) environments, we identify two critical research gaps that fundamentally hinder the deployment of LLM-based solutions in production settings: **excessive false positives** and **prohibitive inference latency**. On the one hand, standalone LLMs suffer from persistently high false positive rates when operating without sufficient contextual knowledge [9]. Enterprise environments often exhibit business-specific behaviors that closely resemble genuine attack patterns, such as authorized vulnerability scanning, internal testing, or customized application workflows. In the absence of enterprise-aware context, LLMs struggle to disambiguate these benign activities from real threats. Moreover, LLMs tend to adopt a conservative decision-making bias, frequently defaulting to classifying ambiguous alerts as malicious. This behavior significantly amplifies false positives, undermining the primary objective of alert triage, which is to reduce analyst workload while preserving detection effectiveness. On the other hand, the inference latency of LLMs poses a severe scalability bottleneck [2, 21]. Even under optimized deployment reminders, evaluating a single alert typically requires several seconds of processing time, especially when advanced reasoning strategies are employed. Such per-alert latency is fundamentally incompatible with operational SOC environments, where security infrastructures routinely generate alerts at the scale of millions per day. As a result, directly applying LLMs to perform deep reasoning over every incoming alert would incur unacceptable delays and computational costs, rendering naive LLM-based triage approaches impractical for large-scale, high-throughput settings.

To address these challenges, we propose SOCRATES, an AI-powered SOC agent designed to automate the network alert triage process. SOCRATES integrates techniques like lightweight filtering and aggregation, enterprise-aware behavior modeling, and context-enhanced reasoning to effectively manage the alert overload in SOCs. By intelligently reducing alert volume and incorporating contextual knowledge, SOCRATES helps mitigate false positives while enhancing the precision of alert classifications, providing a scalable solution to the growing problem of alert fatigue in modern enterprises.

Technical Challenges. There are three major technical challenges in developing SOCRATES:

① *How can computationally expensive LLM reasoning be scaled to handle massive volumes of alerts?* Modern SOC infrastructures generate alerts at an extremely high rate, often reaching millions of events per day. Directly applying LLM-based deep reasoning to every incoming alert is computationally infeasible due to strict latency constraints and prohibitive resource costs. The challenge lies in designing a scalable processing pipeline that can efficiently reduce alert volume while preserving critical security signals, ensuring that intensive LLM reasoning is reserved only for high-value and contextually meaningful alerts.

② *How can the system accurately capture enterprise-specific business semantics?* A significant portion of false positives in alert triage originates from legitimate enterprise activities that closely resemble malicious behavior, such as authorized scanning, internal testing, or customized application workflows. These behaviors are often highly specific to individual organizations and evolve over time, making them difficult to model using static rules or generic threat intelligence. Without an effective mechanism to learn and adapt to enterprise-specific business logic, an automated triage system risks systematically misclassifying benign activities as attacks, thereby failing to reduce alert fatigue.

③ *How can the system equip LLMs with sufficient alert-centric context to support reliable triage decisions under uncertainty?* Security alert triage requires more than accurate pattern recognition; it fundamentally depends on the availability of relevant contextual information surrounding an alert. When operating in isolation, standalone LLMs are forced to reason over incomplete and fragmented evidence, making them prone to conservative bias and speculative judgments. In real-world SOC environments, a single alert is rarely self-contained and often needs to be interpreted in conjunction with related alerts, asset context, and external knowledge sources. Without access to such alert-centric context, LLMs struggle to assess intent, scope, and severity, resulting in unreliable decisions and elevated false positives. Designing a system that can systematically retrieve, integrate, and reason over relevant contextual information, while producing interpretable verdicts grounded in concrete evidence, remains a critical technical challenge.

Key Insights. To address these challenges, SOCRATES incorporates several key insights and techniques:

- **Insight 1: Coarse-to-Fine Alert Reduction via Lightweight Filtering and Aggregation.** Applying deep reasoning to every alert is neither necessary nor feasible at scale. An important insight is that the majority of raw alerts are low-fidelity, redundant, or semantically similar. By employing lightweight filtering and aggregation techniques to remove noise and cluster related alerts, the system can drastically reduce alert volume while preserving critical security signals. This coarse-to-fine processing strategy ensures that computationally expensive LLM reasoning is reserved only for a small set of high-value candidate alerts, effectively overcoming scalability and latency constraints.
- **Insight 2: Enterprise-Aware Behavior Modeling through Self-Learning.** Rather than relying on static rules or generic threat intelligence, effective alert triage requires an understanding of enterprise-specific operational semantics. A key insight is that benign business behaviors that trigger alerts are often repetitive and exhibit stable patterns over time. By continuously learning from historical alert data and traffic characteristics, the system can construct adaptive baselines of normal enterprise

behavior. This enables the identification and suppression of alerts originating from legitimate but attack-like activities, directly addressing the challenge of high false positive rates caused by the lack of enterprise context.

- **Insight 3: Context-Grounded Reasoning through Alert-Centric Context Enrichment.** Accurate triage decisions depend on more than the content of an isolated alert; they require access to relevant contextual evidence. A central insight is that LLMs can make more reliable judgments when equipped with alert-centric context, such as correlated alerts, asset information, and external knowledge sources. By explicitly retrieving and integrating such context into the reasoning process, the system enables evidence-grounded analysis that reduces speculative inference.

Contributions. In this work, we propose SOCRATES, an AI-powered SOC agent designed to automate and optimize the network alert triage process. SOCRATES addresses the core challenges identified in the research gaps by introducing a modular and scalable framework. The system incorporates three key components: a lightweight alert aggregation and filtering module, a business logic self-learning module, and a context-enhanced LLM investigation module. The alert aggregation and filtering module consolidates redundant and similar alerts, reducing overall volume and ensuring that only high-value alerts are processed further. The business logic self-learning module continuously adapts to the enterprise's specific operations, learning from historical data to suppress alerts triggered by benign, recurring business activities. This reduces false positives without relying on static rules. Finally, the context-enhanced LLM investigation module enriches alert analysis by retrieving and integrating relevant contextual information, allowing SOCRATES to make evidence-grounded decisions. Together, these components ensure a balance between efficiency, adaptability, and accuracy, enabling SOCRATES to effectively manage large-scale SOC environments and alleviate alert fatigue while preserving high detection performance.

Evaluation. We evaluate SOCRATES on a large-scale industry arena dataset comprising over 70 million log entries collected over 21 days from a production enterprise environment. The experimental results demonstrate that SOCRATES achieves a significant workload reduction ratio of 95.22%, effectively suppressing 1,846,068 alerts while maintaining high detection integrity. Compared to traditional alert throttling strategies which incur a catastrophic attack loss rate of up to 84.36%, SOCRATES retains 99.63% of true attacks with a negligible loss rate of only 0.37% (187 alerts). Furthermore, by leveraging enterprise-aware context, our system reduces the false positive rate to 19.35% and achieves an f1-score of 82.48%, significantly outperforming standalone LLM baselines and existing state-of-the-art frameworks. These results confirm that SOCRATES provides accurate, scalable, and trustworthy triage decisions grounded in comprehensive evidence, bridging the gap between automated analysis and human expertise.

2 Background & Related Work

2.1 Network Alert Triage

Network alert triage is a fundamental workflow in modern SOCs, whose objective is to progressively reduce massive volumes of raw alerts into a small set of confirmed security incidents. In large-scale enterprise environments, this process is typically organized as a multi-stage, coarse-to-fine pipeline, as illustrated in Figure 1, which balances scalability and analytical depth. At the initial stage, SOCs continuously ingest massive raw alerts generated by heterogeneous security devices such as IDSs and WAFs. These alerts are highly redundant and noisy, as detection mechanisms are often configured to favor coverage over precision [3]. To cope with this scale, SOC platforms first apply coarse-grained filtering, such as whitelist and blacklist matching, alert suppression, and deduplication, to rapidly eliminate obvious noise with minimal overhead [26]. Alerts that survive this stage remain unverified and require further analysis. Subsequently, the alert triage stage focuses on semantic understanding of alert payloads and contextual metadata to assess potential malicious intent. This stage aims to distinguish genuinely suspicious activities from benign anomalies that resemble attacks, producing a smaller set of highly suspicious alerts. These alerts are then escalated to the investigation and validation stage, where correlation across multiple data sources, such as historical alerts, asset context, and threat intelligence, is performed to determine whether an alert constitutes a confirmed security incident. This progressive triage paradigm reflects a key operational principle in SOCs: early stages prioritize throughput and noise reduction, while later stages emphasize accuracy and contextual reasoning. However, traditional alert triage pipelines rely heavily on static rules and manual analysis, limiting their ability to adapt to enterprise-specific behaviors and motivating more intelligent and context-aware triage mechanisms.

2.2 LLM Applications in Cybersecurity

Most LLMs are built on the transformer architecture [37]. The key innovation of this architecture lies in the self-attention mechanism, which allows the model to evaluate the relative importance of different words in the input text and capture long-range dependencies and complex contextual relationships. These models are pre-trained on massive and diverse text corpora to learn the task of predicting the next word in a sequence. As a result, they develop a strong ability to understand meaning in language [43]. This emerging capacity for deep semantic understanding enables LLMs to interpret the implications and intentions behind semi-structured texts, such as the payloads in network alerts. The powerful

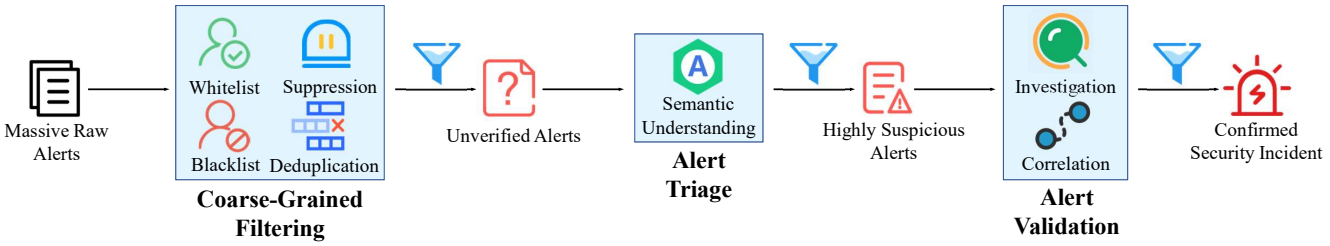

Figure 1: A typical network alert triage pipeline in SOCs, illustrating the coarse-to-fine processing stages from massive raw alerts to confirmed security incidents.

text understanding and reasoning capabilities of LLMs have catalyzed a wave of innovation within the cybersecurity domain [11, 40]. Beyond simple text classification, LLMs are now being explored for a wide range of complex cybersecurity tasks, including threat intelligence analysis [23, 32], malware code dissection [10, 16], penetration testing [8, 29], and incident response plan formulation [18]. Recognizing this potential, several cybersecurity and technology companies have begun developing their own security-specific LLMs. For instance, Microsoft has introduced Security Copilot [27], Google has developed Sec-PaLM as part of its Security AI Workbench [12], and CrowdStrike has launched Charlotte AI [7]. These models are typically trained on massive proprietary cybersecurity datasets and provide more accurate security analysis.

3 Exploratory Study

To investigate the inherent boundaries of LLMs in network alert triage and establish a baseline for subsequent comparison, we first conducted an exploratory study. This evaluation, as illustrated in Figure 2, focuses on the performance of LLMs operating in a standalone mode, where the models lack access to external context such as threat intelligence and asset databases, or specialized security tools. This baseline assessment addresses the following two Research Questions (RQs):

- **RQ1(Performance):** How effective are standalone LLMs in performing network alert triage?
- **RQ2(Bottlenecks):** What are the primary limitations of LLMs during the alert triage process?

We first introduce the curation and preprocessing of our real-world alert dataset in § 3.1. Next, we outline the experimental configuration, including the specific LLMs and prompting strategies tested, in § 3.2. Finally, we present the empirical results and provide an in-depth analysis of the performance bottlenecks in § 3.3.

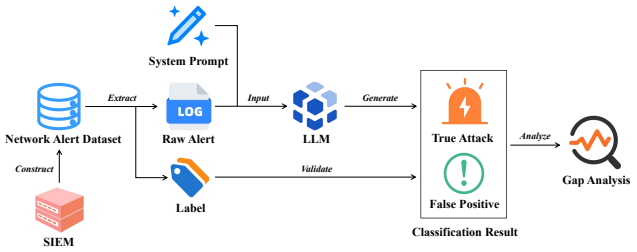

Figure 2: The experimental pipeline of our exploratory study and gap analysis.

3.1 Data Preparation

We collaborated with a partner enterprise to curate a benchmark dataset derived from a production IDS. The raw data collection spans a continuous period of six months and originally comprised millions of alert logs generated within the SOC. Through rigorous preprocessing and deduplication, we extracted a high-quality subset of alerts that preserves essential attributes for triage, such as timestamps, source and destination IP addresses, protocols, and raw payload content, as illustrated in Table 1. To establish a reliable ground truth for evaluation, senior security analysts were engaged to manually inspect and cross-validate each alert instance. We strictly balanced the collection to eliminate class bias, yielding a final dataset of 4,000 distinct samples. This corpus contains exactly 2,000 confirmed malicious incidents labeled as "True Attack" and 2,000 benign anomalies labeled as "False Positive".

3.2 Evaluation Setup

Model Selection. Regarding model selection, we prioritized state-of-the-art open-source models suitable for privacy-sensitive enterprise environments. To rigorously investigate the impact of parameter scaling and underlying model structures on triage performance, we focused specifically on the Qwen3 family. This selection includes both standard *Dense* models across different scales (e.g., `Qwen3-14B`, `Qwen3-32B`) and efficient *Mixture-of-Experts (MoE)* variants (e.g., `Qwen3-30B-A3B`).

Prompting Strategies. Recognizing that the performance of LLMs is heavily contingent upon input formulation, we

Table 1: Structure and content of a representative preprocessed alert instance.

Timestamp	**Sip**	**Dip**	**Proto**
1747037759	10.16.3.237	10.115.50.29	http
Rule Name		**Sport**	**Dport**
Directory Traversal Attack		46144	8081
Package Data			
Request Header		**Request Body**	
GET /manage/log/view?filename=/etc/passwd&base=../../../../../../../../ HTTP/1.1		N/A	
Response Header		**Response Body**	
HTTP/1.1 400 Bad Request Content-Type: text/plain; charset=utf-8		400 Bad Request	
Payload			
filename=/etc/passwd&base=.. /../../../../../../../			
Label			
False Positive			

evaluated four distinct prompting strategies: *Zero-shot*, *Few-shot*, *Chain-of-Thought (CoT)*, and *Expertise*. In the *Few-shot* configuration, we strictly ensured that no samples from the evaluation dataset were utilized to prevent data leakage. We randomly selected exactly two demonstration instances from a separate held-out dataset, consisting of one verified "True Attack" and one "False Positive" to maintain class balance. To ensure reproducibility and transparency, the complete prompt templates and detailed configurations are provided in Appendix C.

Implementation & Metrics. We utilized the Ollama framework for local model deployment and inference management. All models were executed using their default hyperparameter configurations, specifically setting the temperature to 0 to ensure deterministic outputs. To ensure a strictly controlled variable for efficiency evaluation, all inference tasks were conducted on a standardized hardware environment equipped with a single 80GB NVIDIA A100 GPU. We assess model efficacy using standard classification metrics, including True Positive Rate, False Positive Rate, and F1-score. Additionally, we record the Average Processing Time per alert to gauge operational efficiency.

3.3 Results

Table 2 summarizes the performance of three representative Qwen3 variants (14B, 30B-A3B, 32B) under different prompting strategies.

Answer to RQ1: Performance of Standalone LLMs. From a high-level perspective, standalone LLMs demonstrate strong detection capability in terms of recall. Averaged across all evaluated models and prompting strategies, the TPR reaches 91.08%, indicating that most malicious alerts are successfully identified. This observation holds consistently across different architectures, suggesting that modern LLMs possess sufficient

Table 2: Comprehensive evaluation results across different prompting strategies. global best and global worst values (maximum for TPR and F1, minimum for FPR and Time). Within each model block, **bold** indicates the best result and underlining denotes the worst result.

Model	**Strategy**	**TPR** ↑	**FPR** ↓	**F1** ↑	**Time**♣ ↓
Qwen3-32B	Zero-shot	**99.3%**	83.7%	70.2%	**20.36**
	Expertise	95.2%	72.0%	71.3%	26.65
	Few-shot	98.4%	**58.6%**	**76.6%**	23.62
	CoT	94.9%	69.1%	72.7%	107.63
Qwen3-30B-A3B	Zero-shot	93.6%	63.7%	72.8%	16.75
	Expertise	**95.8%**	75.7%	70.6%	**11.15**
	Few-shot	63.7%	**26.6%**	66.9%	21.68
	CoT	90.6%	40.7%	**78.3%**	131.28
Qwen3-14B	Zero-shot	92.1%	68.2%	70.8%	**5.67**
	Expertise	83.8%	67.2%	66.8%	10.88
	Few-shot	**93.7%**	**47.0%**	**77.8%**	7.13
	CoT	91.8%	69.6%	70.3%	80.87
Avg	–	91.08%	61.83%	72.08%	38.64

♣ **Time** denotes the average per-alert triage latency of the LLM, measured in seconds.

semantic reasoning ability to recognize attack-like patterns in raw network alerts. However, this high recall comes at the cost of poor precision. As shown in the Avg row of Table 2, the average FPR remains as high as 61.83%, with several configurations exceeding 60%. In the context of SOCs, where the primary goal of alert triage is to reduce analyst workload rather than maximize detection coverage, such a high false positive rate is operationally unacceptable. These results indicate that while standalone LLMs are effective at flagging suspicious activity, they fail to reliably distinguish true threats from benign anomalies, limiting their practical utility for alert triage.

Answer to RQ2: Bottlenecks in Alert Triage with LLMs. The experimental results, combined with a manual inspection of model outputs, reveal two fundamental bottlenecks that limit the applicability of standalone LLMs for alert triage: high false positive rates and prohibitive inference latency. From the perspective of false positives, our analysis shows that LLMs struggle to accurately distinguish malicious attacks from benign enterprise-specific activities that closely resemble attack behaviors. Through manual examination of the CoT reasoning traces, we observe that many misclassifications arise from the absence of essential contextual knowledge about enterprise environments. Certain legitimate business operations, such as authorized vulnerability scanning, internal testing, or customized application workflows, exhibit traffic patterns that are syntactically and semantically similar to real attacks. Lacking access to enterprise-specific context, LLMs are unable to reliably disambiguate such cases and tend to default to conservative judgments. This bias toward classifying ambiguous alerts as attacks systematically inflates the false positive rate.

In terms of efficiency, the inference latency of LLMs presents a significant operational bottleneck. Even under op-

timized local deployment, the inference latency per alert is substantial, ranging from approximately 5 seconds to over 130 seconds depending on the reasoning strategy. This latency is fundamentally incompatible with real-world SOC environments, where millions of alerts may be generated on a daily basis. As a result, directly applying LLMs to perform per-alert deep reasoning at scale would introduce unacceptable delays and computational costs, rendering standalone LLM-based triage impractical in high-throughput settings.

4 Methodology

This section describes the overall methodology of SOCRATES, a modular and scalable alert triage framework designed to address the practical challenges of excessive false positives and limited reasoning capacity in real-world SOC environments. Instead of applying computationally expensive LLM reasoning to all incoming alerts, SOCRATES adopts a staged, coarse-to-fine processing pipeline that progressively reduces alert volume while incrementally enhancing semantic and contextual understanding. As illustrated in Figure 3, the methodology consists of three sequential modules. First, a lightweight alert aggregation and filtering module (§ 4.1) efficiently consolidates redundant alerts and suppresses low-value noise at scale, serving as the system's front-line throughput control. Second, a business self-learning module (§ 4.2) models enterprise-specific benign behavior patterns from historical alerts, enabling the system to identify and suppress business-induced false positives that are difficult to capture with static rules. Finally, an alert context–enhanced LLM investigation module (§ 4.3) performs evidence-grounded reasoning over a small subset of high-value alerts by integrating enriched contextual information, producing interpretable and analyst-aligned triage decisions.

4.1 Lightweight Alert Aggregation and Filtering

In large-scale SOC environments, raw alerts generated by heterogeneous security devices are often highly redundant and temporally correlated, resulting in substantial low-value noise. Directly forwarding such high-volume alert streams to downstream learning or reasoning components is impractical due to scalability and efficiency constraints. To address this challenge, we introduce a ***lightweight alert aggregation and filtering module*** as the first stage of the pipeline, which performs low-cost consolidation and coarse-grained prioritization to regulate alert throughput for subsequent analysis.

4.1.1 Alert Normalization

To enable efficient alert aggregation and filtering, raw alerts from heterogeneous security devices are first converted into a unified and consistent representation through normalization. We define an alert event model and map all raw alerts into this common schema, capturing shared structural attributes such as source and destination identifiers, transport protocol, ports, rule identifiers, and timestamps, while preserving protocol-specific semantics via dedicated sub-fields (e.g., `method`, `uri`, and `status_code` for HTTP alerts, or protocol-relevant flags for non-HTTP traffic). Optional fields such as `severity` and `confidence` are retained to accommodate device-specific attributes. To reduce superficial syntactic variability among alerts that exhibit the same underlying behavior, content-bearing fields are further normalized by abstracting instance-specific elements, including numeric values, timestamps, session identifiers, random tokens, hashes, and variable-length strings, into canonical placeholders (e.g., `<NUM>`, `<TIME>`, `<SESSION>`, `<TOKEN>`, `<HASH>`) using lightweight pattern-based rules. For HTTP-related alerts, request paths and query strings are transformed into URI templates by preserving stable path structures while masking variable arguments (e.g., `/download/tmp/ab93f7c21e4f9a8.log` → `/download/tmp/<TOKEN>.log`). Similar normalization is applied to payload fragments and headers, ensuring that protocol-relevant keywords are retained while dynamic values are abstracted.

4.1.2 Alert Aggregation

After normalization, alerts are aggregated to consolidate redundant events triggered by identical or highly similar activities. This submodule groups alerts based on stable structural attributes and temporal proximity, producing compact aggregated representations that capture repeated behaviors without retaining individual alert instances.

Aggregation key construction. Each alert is assigned to an aggregation bucket based on a composite key constructed from stable structural attributes in the unified event model. Specifically, we define the aggregation key as a tuple of $(\texttt{sip}, \texttt{dip}, \texttt{proto}, \texttt{rule\_name}, [\texttt{uri\_template}])$, where `sip` and `dip` denote the source and destination identifiers, `proto` captures the transport protocol, `rule_name` represents the triggering detection rule, and `uri_template`, when present, corresponds to the normalized content template. This key groups alerts that originate from the same source, target the same destination, and exhibit equivalent behavioral patterns.

Field Consolidation. Alerts that share the same aggregation key are merged within a bounded time interval, such that multiple occurrences of the same alert pattern observed within a day are represented as a single aggregated alert. During aggregation, we retain a compact representation of the merged alerts and introduce an explicit reference field that records the identifiers of all constituent raw alerts. This reference enables efficient backtracking from an aggregated alert to its underlying alert instances, allowing on-demand retrieval of detailed metadata during downstream analysis or investigation.

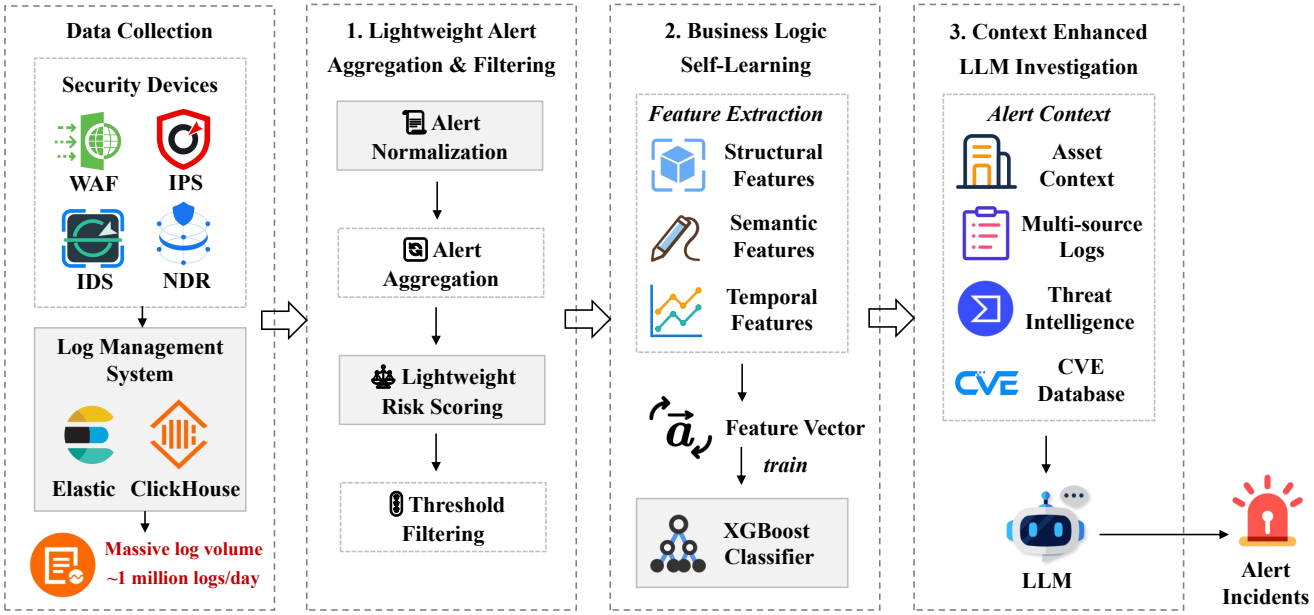

Figure 3: System architecture of SOCRATES for automated and context-aware alert triage.

4.1.3 Lightweight Risk Scoring and Threshold Filtering

Following aggregation, the resulting alerts differ in their relative importance and investigative priority. This submodule assigns coarse-grained risk scores to aggregated alerts and applies threshold-based filtering to determine which alerts should be forwarded to downstream analysis stages.

Lightweight risk scoring For each aggregated alert, we compute a coarse-grained risk score by combining several lightweight and interpretable scoring components. Each component captures a distinct aspect of alert salience:

- **Frequency score** (S_{freq}): Quantifies the repetition intensity of an alert pattern within the aggregation interval. Alerts with higher occurrence counts or denser inter-arrival patterns yield larger S_{freq} values.
- **Rule and metadata score** (S_{rule}): Derived from detection metadata provided by upstream security devices, such as rule category or `severity`. Alerts associated with higher-risk rules contribute larger S_{rule} values.
- **Contextual heuristic score** (S_{ctx}): Encodes coarse contextual cues based on readily available metadata, such as external exposure of the source or sensitivity of the target asset, without requiring additional context retrieval.
- **Historical rarity score** (S_{rare}): Reflects how uncommon an alert pattern is relative to recent history (the past 14 days). Rarely observed patterns receive higher S_{rare} values, whereas frequently recurring alerts are down-weighted.

Threshold-based filtering and alert forwarding. The individual scoring components are combined to produce a unified threat priority score for each aggregated alert. Let $\mathcal{K} = \{\text{freq}, \text{rule}, \text{ctx}, \text{rare}\}$ denote the set of scoring dimensions, where each dimension $k \in \mathcal{K}$ corresponds to a normalized score $\hat{S}_k$. The aggregated priority score is computed as

$$\mathcal{S} = \sigma\left(\sum_{k \in \mathcal{K}} w_k \hat{S}_k\right)$$

where w_k is a configurable weight for score k, and $\sigma(\cdot)$ is a monotonic squashing function used to bound the score range. Importantly, $\mathcal{S}$ represents a coarse-grained *priority score* rather than a definitive threat verdict. The score is used solely to rank and filter aggregated alerts: alerts with $\mathcal{S}$ exceeding a predefined threshold are forwarded to downstream modules for further analysis, while lower-scoring alerts are deprioritized or suppressed.

4.2 Business Logic Self-Learning

A primary source of false positives in enterprise SOC environments stems from legitimate business activities that closely resemble malicious behaviors, such as authorized vulnerability scanning, internal testing, and customized application workflows. These business-induced alerts are typically persistent, repetitive, and organization-specific, making static rules and generic threat intelligence largely ineffective. To mitigate this, we introduce a business logic self-learning module designed to explicitly model enterprise-specific benign patterns from historical data. As detailed in the following sections, the module primarily operates through two tightly coupled stages: (i) *Business Baseline Induction*, which learns stable behavior prototypes from historical false positives, and (ii) *Business-Aware Alert Matching*, which leverages trained models to identify and suppress recurring benign alerts in real-time.

4.2.1 Business Baseline Induction

In the offline stage, we aim to induce a business baseline that captures recurring benign alert patterns specific to enterprise environments and supports automated suppression of business-induced false alerts. By learning from historically confirmed false positives, the induced baseline encodes normal business behavior and provides a foundation for training a discriminative model to distinguish benign alerts from other events.

Feature Extraction. Analysis of real-world SOC data shows that business-induced false alerts exhibit stable regularities in structure, semantics, and timing. Motivated by these observations, we extract structural, semantic, and temporal features to represent each alert.

① *Structural Features.* Structural features capture the stable and interpretable attributes of alerts that reflect how detection rules are triggered in enterprise environments. In our setting, we consider a set of commonly available structural fields, including the source IP address (`sip`), destination IP address (`dip`), source and destination ports (`sport`, `dport`), transport or application-layer protocol (`proto`), the triggering detection rule identifier (`rule_name`), and the accessed resource or endpoint (`uri`) when applicable. These fields jointly describe the communication pattern, protocol context, and detection semantics of an alert. For business-induced false alerts, such structural attributes tend to remain highly stable over time, as they are often tied to fixed services, assets, or automated workflows, making them particularly informative for distinguishing benign business behavior from other alerts. We apply lightweight and field-specific encoding strategies that preserve discriminative information while controlling feature sparsity. Categorical fields such as `proto` and `rule_name` are encoded using frequency-based representations, allowing the model to learn rule- or protocol-specific patterns. IP-related fields (`sip`, `dip`) are transformed into hashed identifiers, to reduce cardinality while retaining structural consistency. Port numbers (`sport`, `dport`) are encoded as raw numerical. For URI fields, we normalize and abstract variable components (e.g., identifiers or random tokens) into templates and encode them as categorical features.

Formally, for an alert a_i, we denote its encoded structural representation as

$$x_i^{\text{str}} = \phi_{\text{str}}(\texttt{sip}_i, \texttt{dip}_i, \texttt{sport}_i, \texttt{dport}_i, \texttt{proto}_i, \texttt{rule\_name}_i, \texttt{uri}_i).$$

where $\phi_{\text{str}}(\cdot)$ represents the collection of field-specific normalization and encoding functions described above. Together, these encodings yield a compact and structured representation that enables efficient learning of recurring business-induced alert patterns.

② *Semantic Features.* Although business-induced false alerts may vary significantly in instance-specific parameters, the core behavioral intent reflected in the alert-triggering content is often semantically consistent. To capture this consistency, we focus semantic feature extraction on the *alert payload*, i.e., the key protocol messages or traffic fragments that directly trigger detection rules.

For each alert a_i, we extract its normalized payload $payload_i$, which preserves protocol-level grammatical structure while masking volatile and instance-specific fields. The payload is then encoded using a lightweight pre-trained sentence embedding model to obtain a fixed-length dense representation:

$$x_i^{\text{sem}} = \phi_{\text{sem}}(payload_i),$$

where $\phi_{\text{sem}}(\cdot)$ denotes the semantic encoding function and $payload_i$ represents the normalized alert payload associated with a_i.

③ *Temporal Features.* Temporal features provide lightweight calendar-level cues that describe when an alert occurs, which can help distinguish routine business workflows from irregular activities. In the temporal feature extraction process, we expand the features derived from the timestamp t_i by incorporating both basic and more detailed calendar attributes. Specifically, we derive the following features: the hour-of-day $h_i \in \{0,\ldots,23\}$, indicating the specific hour when the alert was generated; the day-of-week $d_i \in \{0,\ldots,6\}$, representing the day of the week (Monday–Sunday); the weekend indicator $w_i \in \{0,1\}$, which is set to 1 if the day is Saturday or Sunday, and 0 otherwise; the business-hours indicator $b_i \in \{0,1\}$, which is set to 1 if the hour falls within business hours (default $[9,18)$) and 0 otherwise. In addition to these, we introduce month-of-year $m_i \in \{1,\ldots,12\}$, quarter-of-year** $q_i \in \{1,2,3,4\}$, an is-holiday indicator $h_i^{\text{holiday}} \in \{0,1\}$, and time since last alert t_i^{last}. To ensure a comprehensive temporal representation, we also include the raw timestamp $@timestamp$, which allows for accurate chronological mapping.

The complete set of temporal features is then encoded into a high-dimensional feature vector as follows:

$$x_i^{\text{tmp}} = \phi_{\text{tmp}}(t_i) = \left[h_i, d_i, w_i, b_i, m_i, q_i, h_i^{\text{holiday}}, t_i^{\text{last}}, @\text{ts}\right]$$

This extended encoding captures both basic and advanced time-related information, enriching the model's temporal awareness and improving its ability to differentiate between normal operational patterns and potential anomalies.

Business Pattern Prototype Induction. The extracted feature vectors for structural (x_i^{str}), semantic (x_i^{sem}), and temporal (x_i^{tmp}) modalities differ significantly in dimensionality and scale. Direct concatenation of these raw vectors may lead to model bias, where high-dimensional semantic embeddings dominate the learning process while low-dimensional structural or temporal cues are overshadowed. To mitigate this heterogeneity, we employ a projection-based alignment strategy. We map each distinct feature vector into a shared latent space of uniform dimension D using modality-specific linear

transformations followed by non-linear activation functions. Formally, the aligned representations are computed as:

$$\mathbf{h}_i^{\text{str}} = \text{ReLU}(\mathbf{W}_s x_i^{\text{str}} + \mathbf{b}_s),$$
$$\mathbf{h}_i^{\text{sem}} = \text{ReLU}(\mathbf{W}_e x_i^{\text{sem}} + \mathbf{b}_e),$$
$$\mathbf{h}_i^{\text{tmp}} = \text{ReLU}(\mathbf{W}_t x_i^{\text{tmp}} + \mathbf{b}_t),$$

where $\mathbf{W}_{\{s,e,t\}}$ and $\mathbf{b}_{\{s,e,t\}}$ are the learnable weight matrices and bias vectors for the respective modalities. By projecting disparate inputs to a uniform dimensionality, we ensure that each information source contributes equally to the representation capacity. Finally, these aligned vectors are concatenated to form the unified alert embedding x_i:

$$x_i = [\mathbf{h}_i^{\text{str}} \parallel \mathbf{h}_i^{\text{sem}} \parallel \mathbf{h}_i^{\text{tmp}}].$$

This composite vector x_i serves as the balanced and comprehensive input for the downstream classification model.

We employ XGBoost as the core learner due to its efficiency in modeling heterogeneous features and robustness to noisy labels without requiring manual normalization. To address class imbalance, we utilize the `scale_pos_weight` mechanism to prioritize the identification of minority business-induced false alerts, ensuring conservative pattern induction. Furthermore, we adopt a 14-day window for training data selection, which offers an optimal balance between stability and adaptability; this duration is sufficient to capture recurring weekly workflows while excluding stale patterns associated with outdated configurations.

4.2.2 Business-Aware Alert Matching

After the lightweight alert denoising and aggregation stage, the input to this module consists of aggregated alerts, each of which represents a group of highly similar raw alerts compressed within a bounded time window. For each aggregated alert, we leverage its reference field to retrieve the corresponding raw alert instances and recover their original structural, semantic, and temporal information. Each raw alert is then independently encoded using the same feature extraction pipeline described in the business pattern induction stage and evaluated by the trained XGBoost model, yielding a per-instance likelihood score that reflects how strongly the alert matches learned business-induced false alert patterns.

To determine whether an aggregated alert should be classified as a business-induced false alert, we aggregate the instance-level model outputs into a unified score. Specifically, given an aggregated alert containing a set of raw alerts $\{a_1, \ldots, a_n\}$ with predicted false-alert probabilities $\{s_1, \ldots, s_n\}$, we compute a weighted aggregate score that accounts for both the consistency and prevalence of business-like behavior within the group. The aggregated alert is classified as a false alert only if its weighted score exceeds a predefined threshold, ensuring that suppression decisions are conservative and robust to occasional outliers. Aggregated alerts that fail to meet this criterion are forwarded to downstream investigation modules for further analysis.

4.3 Context Enhanced LLM Investigation

After the lightweight aggregation and business logic self-learning stages, the remaining alerts represent a small set of high-value candidates that require in-depth analysis. However, individual alerts are often insufficient to support reliable security judgments in isolation, as critical evidence is distributed across assets, historical behaviors, host and network telemetry, and external threat intelligence. To address this limitation, we introduce a context-enhanced LLM investigation module, which systematically retrieves and integrates alert-centric contextual information before performing evidence-grounded reasoning. By coupling large language models with structured context retrieval and tool-assisted queries, this module emulates the investigative workflow of human SOC analysts and produces interpretable triage decisions supported by concrete evidence.

4.3.1 Alert-Centric Context Retrieval

Effective alert triage requires reasoning over contextual information that extends beyond the content of a single alert. In SOCRATES, we organize alert-centric context into two complementary categories: external intelligence and internal enterprise intelligence, which together provide the factual foundation for downstream LLM-based investigation.

External Intelligence. External intelligence captures security knowledge that is independent of a specific enterprise environment but critical for interpreting potential threats. This includes vulnerability information such as associated CVEs and affected software versions, IP and domain reputation data from threat intelligence feeds, as well as adversary tactics, techniques, and procedures (TTPs) aligned with the MITRE ATT&CK framework. Such information helps contextualize whether an observed alert pattern is consistent with known exploitation behaviors or threat campaigns.

Internal Enterprise Intelligence. Internal intelligence reflects organization-specific context that is essential for disambiguating benign business activities from genuine attacks. This category includes asset metadata such as asset groups, exposure level, business criticality, and ownership, as well as correlated telemetry from multiple internal data sources. In particular, we incorporate logs from heterogeneous security and monitoring systems, including web application logs, endpoint detection and response (EDR) telemetry, authentication records, and network flow data. These sources provide behavioral and environmental evidence that cannot be inferred from external knowledge alone.

Context Retrieval Interface to LLMs. To operationalize the access to both external and internal intelligence, we implement a unified retrieval layer developed in Python that encap-

sulates heterogeneous data sources into a set of structured, callable tools. Rather than embedding massive raw datasets directly into the prompt, this interface exposes specific investigative capabilities—such as querying asset ownership, searching historical logs within a time window, or validating IP reputation—as standardized API functions. We construct a comprehensive tool definition registry that is provided to the LLM within its system prompt. This registry details the name, functional description, and parameter schema for each available tool, enabling the model to semantically understand the utility of each interface. By parsing this definition, the LLM can autonomously invoke the necessary tools to dynamically gather evidence, ensuring that its reasoning process is grounded in real-time, retrievable data.

4.3.2 Investigation Workflow

The investigation workflow in SOCRATES is designed to effectively query and retrieve relevant data from multiple sources, both internal and external, while ensuring that the process remains efficient and scalable. The workflow consists of several key steps:

Retrieving Multi-source Logs. Our logs are centrally collected and managed via the Elastic platform, where each log type is assigned a distinct index. To begin the investigation, we provide sufficient contextual information related to the logs, which allows the LLM to construct appropriate query parameters. Based on these parameters, the LLM can query the corresponding logs from Elastic, retrieving the relevant entries that align with the specific conditions defined in the alert. This approach ensures that all internal data sources are efficiently queried and integrated into the investigation process.

Retrieving External Knowledge. To enhance the investigation, we illustrate the process of retrieving external knowledge using threat intelligence as an example. We leverage platforms such as Virustotal, a globally recognized threat intelligence aggregation service. Virustotal offers an API interface that allows SOCRATES to send queries related to known threats, domain reputation, or IP addresses. The LLM constructs the necessary parameters based on the alert's context and sends a request to the Virustotal API. The query results are then returned and used to enrich the investigation with up-to-date external threat intelligence data.

Overall Process. Upon receiving an alert, the LLM first uses predefined prompts to assess which tools are available for the investigation. Based on this assessment, the LLM initiates the plan phase, in which it outlines the general course of action for the investigation, identifying which data sources and tools will be used and in what sequence. After finalizing the plan, the LLM proceeds with executing the queries as defined in the plan. To prevent context overload and ensure efficient reasoning, once a tool is queried and results are returned, the LLM first independently processes the results, converting the data into a natural language format. This allows the results to be meaningfully integrated back into the ongoing context. The process continues iteratively, where the LLM repeatedly calls relevant tools, interprets the results, and updates the context, building on each piece of evidence. Finally, after all relevant information is gathered and analyzed, the LLM synthesizes the results into a comprehensive understanding of the alert, making a final judgment based on the accumulated evidence. This iterative, evidence-driven workflow ensures that the investigation remains systematic and accurate while keeping computational overhead in check.

Table 3: Overview of the industry arena dataset.

Category	Log ID	Log Type	#Logs	Size	Span
Alerts	Log-A	IDS Alerts	9,481,751	25.74 GB	
Context	Log-B	IDS Logs	6,203,400	17.19 GB	21 d
	Log-C	WAF Logs	38,970,536	200.72 GB	
	Log-D	Web Server Logs	15,362,361	41.73 GB	
	Log-E	Network Threat Logs	80,551	531.40 MB	
	Log-F	Endpoint Logs	128,675	245.02 MB	
Total	–	–	**70,227,274**	**286.14 GB**	–

5 Evaluation

In this section, we evaluate SOCRATES to assess its effectiveness and practicality. The evaluation is designed to answer the following research questions:

- **RQ3 (Effectiveness):** How does the SOCRATES system perform in network alert triage?
- **RQ4 (Comparison):** How does the performance of SOCRATES compare to existing alert triage systems?
- **RQ5 (Industrial Deployment):** How does SOCRATES perform in real-world industrial deployment scenarios?

5.1 Evaluation Setup

5.1.1 Industry Arena Dataset

Most existing alert datasets focus on a single type of security log or alert source, overlooking the fact that real-world security operations typically rely on correlating signals across multiple heterogeneous data sources. To reflect the operational reality of SOCs where decisions rely on cross-source correlation, we constructed an Industry Arena Dataset collected from a production enterprise environment. Unlike existing single-source datasets, ours comprises six distinct log streams: one primary IDS alert stream serving as the triage target, and five auxiliary sources (network, web, and endpoint logs) used exclusively for contextual enrichment. The data spans a 21-day window, a duration selected to balance the high cost of manual labeling with the need to capture recurring business behaviors and periodic attack patterns. All alerts within this period were

Table 4: Comprehensive analysis of SOCRATES performance across multiple modules.

Date	Total Alerts	Module 1				Module 2				Module 3		
		Type	Count	Agg. Total	ARR	Type	Count	Agg. Total	ARR	TPR↑	FPR↓	F1↑
Day1	273,824	Attack	7,945	16,333	94.04%	Attack	7,941	434	97.34%	93.90%	17.41%	84.38%
		False Alarm	265,879			False Alarm	5,291					
Day2	310,511	Attack	9,347	18,092	94.17%	Attack	9,194	1,912	89.43%	89.40%	19.08%	81.99%
		False Alarm	301,164			False Alarm	7,767					
Day3	318,788	Attack	8,142	15,910	95.01%	Attack	8,122	422	97.35%	90.12%	20.00%	81.11%
		False Alarm	310,646			False Alarm	5,850					
Day4	312,102	Attack	8,291	16,003	94.87%	Attack	8,286	226	98.59%	93.02%	20.00%	82.47%
		False Alarm	303,811			False Alarm	6,344					
Day5	273,395	Attack	5,333	16,093	94.11%	Attack	5,331	234	98.55%	92.04%	14.88%	88.51%
		False Alarm	268,062			False Alarm	5,377					
Day6	228,327	Attack	5,743	12,826	94.38%	Attack	5,741	75	99.42%	86.21%	17.39%	80.65%
		False Alarm	222,584			False Alarm	5,073					
Day7	221,778	Attack	6,075	12,144	94.52%	Attack	6,074	75	99.38%	90.00%	26.67%	78.26%
		False Alarm	215,703			False Alarm	6,266					
Total	**1,938,725**	/	/	**107,401**	**94.44%**	/	/	**3,378**	**97.15%**	**90.67%**	**19.35%**	82.48%

rigorously inspected by security analysts to establish reliable ground truth.

An overview of the resulting dataset is summarized in Table 3. In total, the dataset comprises over 70 million log entries collected over a 21-day period, occupying approximately ~286 GB of raw data and spanning heterogeneous security domains, including intrusion alerts, network traffic inspection, web application activity, and endpoint security events. The primary alert stream alone contains more than 9.4 million alerts, reflecting the high-volume and high-noise nature of real-world SOC environments. In contrast, the auxiliary logs collectively contribute over 60 million additional records, providing rich and diverse contextual signals.

5.1.2 Metrics

We evaluate SOCRATES using four complementary metrics: True Positive Rate (TPR), False Positive Rate (FPR), F1-score, and Alert Reduction Ratio (ARR).

- **True Positive Rate (TPR):** $\mathrm{TPR} = \mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$, measuring the proportion of true attacks that are correctly identified.
- **False Positive Rate (FPR):** $\mathrm{FPR} = \mathrm{FP}/(\mathrm{FP}+\mathrm{TN})$, indicating the fraction of benign alerts incorrectly classified as attacks.
- **F1-score:** $\mathrm{F1} = 2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}/(\mathrm{Precision}+\mathrm{Recall})$, providing a balanced measure of triage accuracy.
- **Alert Reduction Ratio (ARR):** $\mathrm{ARR} = 1 - N_{\mathrm{triaged}}/N_{\mathrm{total}}$, representing the proportion of alerts filtered out before final triage decisions.

5.2 RQ3: Effectiveness of SOCRATES

We first evaluate the overall effectiveness and efficiency of SOCRATES in real-world alert triage. The results demonstrate that SOCRATES is able to substantially reduce alert noise while preserving detection capability, and achieve practical efficiency suitable for large-scale SOC environments.

Alert Reduction. The primary objective of SOCRATES' alert triage system is to significantly reduce the number of alerts without sacrificing the ability to detect genuine threats. As shown in Table 4, the aggregation rate of Module 1 is consistently high, ranging from 94.04% to 95.01% across different days, reflecting the system's efficiency in consolidating redundant alerts. Additionally, the noise reduction rate achieved by Module 2 further demonstrates SOCRATES' capability to filter out irrelevant or benign alerts, with reductions varying between 89.43% and 99.42%. For instance, on Day 6, Module 2 managed to reduce the alert count from 12,826 to just 75, showcasing its strength in eliminating non-critical alerts, thereby enabling SOC analysts to focus on high-value events.

Accurate Identification of Business False Positives. SOCRATES excels in distinguishing between legitimate business activities and actual attacks, addressing a significant challenge in modern SOCs. As seen in the data, Module 2 effectively suppresses business-induced false positives, leading to a marked decrease in false alarm alerts. The number of attack alerts remains stable across both Module 1 and Module 2, with minimal variation in the attack count (e.g., Day 1: 7,945 vs. 7,941). However, Module 2 substantially reduces the false positive alerts, with the number of false alarms dropping from 265,879 to 5,291 on Day 1. This indicates that the system accurately identifies benign business behaviors and removes them without mistakenly flagging real attacks, ensuring high

Table 5: Performance comparison of different alert throttling configurations.

Exp. ID	Throttling Parameters			Suppression Results (Count & Percentage)		
	Trigger Window	Threshold	Suppression Window	Total Suppressed	Attack Loss	Benign Reduction
Exp 1	10 min	5	30 min	803,970 (41.47%)	40,354 (79.32%)	763,616 (40.45%)
Exp 2	10 min	3	30 min	1,038,367 (53.56%)	42,921 (84.36%)	995,446 (52.73%)
Exp 3	10 min	5	10 min	668,463 (34.48%)	35,010 (68.81%)	633,453 (33.55%)
Exp 4	5 min	5	30 min	705,237 (36.38%)	36,858 (72.45%)	668,379 (35.40%)
SOCRATES	/			**1,846,068 (95.22%)**	**187 (0.37%)**	**1,845,881 (97.78%)**

detection reliability.

Effectiveness of Module 3. With Module 2 effectively filtering out business-induced false positives that are challenging for large models to identify, and Module 3 providing the remaining alerts with enriched contextual information, SOCRATES significantly enhances the accuracy of its triage decisions. The addition of relevant context allows the LLM to better interpret ambiguous alerts, leading to a substantial reduction in the false positive rate. Notably, the TPR remains stable across all days, indicating that the system consistently identifies the vast majority of actual attacks. For instance, on average, the FPR across all days is reduced to 19.35%, with Day 5 exhibiting a notably low FPR of just 14.88%. This demonstrates how SOCRATES balances high detection rates with a significant reduction in false alarms by leveraging both reduced alert volume and enhanced context in Module 3.

5.3 RQ4: Comparison with Existing Technologies

In this section, we compare the performance of SOCRATES with existing approaches in the field of network alert triage.

Comparison with Alert Throttling Alert throttling is a commonly used noise-reduction technique in enterprise environments. This method works by suppressing repeated alerts from the same source within a specific time window, typically after a predefined threshold of occurrences. The goal is to prevent the system from continuously flagging the same event and reduce the alert volume. The main advantage of alert throttling is its simplicity and quick effectiveness in minimizing alert overload. However, its drawback lies in the possibility of missing actual attacks that exhibit similar patterns but occur at irregular intervals, as throttling may inadvertently suppress legitimate alerts in such cases.

To quantify the advantages of our semantic-aware approach over traditional volume-based heuristics, we compared SOCRATES with four standard alert throttling configurations (Exp 1-4). As detailed in Table 5, heuristic throttling exhibits a fundamental conflict between noise reduction and detection recall. For instance, the aggressive configuration in Exp 2 achieved a moderate benign reduction of 52.73%, but this came at a catastrophic cost: suppressing 84.36% of genuine attack alerts. In contrast, SOCRATES effectively breaks this trade-off by leveraging coarse-to-fine semantic filtering. As shown in the bottom row of Table 5, our system achieves a superior benign reduction rate of 97.78%, filtering out nearly 1.85 million false positives. Crucially, this massive noise reduction does not compromise security visibility: SOCRATES retains 99.63% of true attacks, with an attack loss rate of only 0.37% (187 alerts). These results demonstrate that SOCRATES can distinguish between benign repetitive business behaviors and malicious high-frequency attacks, providing a level of precision that simple frequency-based throttling cannot match.

Table 6: Performance comparison between the standalone LLM baseline and SOCRATES.

Model	Strategy	TPR↑	FPR ↓	F1-score ↑
Standalone LLM	Zero-shot	93.60%	63.70%	72.80%
	Expertise	**95.80%**	75.70%	70.60%
	Few-shot	63.70%	26.60%	66.90%
	CoT	90.60%	40.70%	78.40%
SOCRATES	/	90.67%	**19.35%**	**82.48%**

Comparison with Standalone LLMs. To demonstrate the necessity of our architectural design, we compare SOCRATES with the standalone LLM architecture evaluated in § 3. As summarized in Table 6, compared to the standalone baseline, our system achieves a superior balance between detection recall and precision. While the standalone model suffers from a prohibitive FPR of 40.70%, SOCRATES drastically reduces this to 19.35% while maintaining a high TPR of 90.67%, resulting in an improved F1-score of 82.48%. This performance leap highlights the distinct contributions of our modules: Module 2 effectively identifies enterprise-specific benign patterns to suppress noise, while Module 3 equips the LLM with retrieved evidence to support robust, context-grounded reasoning.

Comparison with State-of-the-Art Approaches. To evaluate the standing of SOCRATES in the current research landscape, we conduct a comparative analysis against three representative state-of-the-art triage systems: DEEPCASE [36], CALLEE [20], and the stream-supervised approach [35]. It is critical to note that since these frameworks were evaluated on distinct, proprietary industrial datasets (e.g., the Lastline dataset or private ISP logs) that are not publicly accessible, a direct experimental reproduction under a unified environment is infeasible. Therefore, we evaluate SOCRATES by cross-referencing the performance metrics reported in their respective original publications and qualitatively assessing their architectural capabilities, as summarized in Table 7. While ex-

Table 7: Comparison of SOCRATES with state-of-the-art alert triage frameworks.

Method	Core Technique	Workload Reduction	Detection Performance	Context Scope	Semantic Understanding	Natural Language Explanation
DEEPCASE [36]	Attention-based DL	90.53%	Acc: 94.34%	Local Seq.	×	×
Stream-Supervised [35]	Stream Clustering + RF	Not Report	F1: ≈0.91	Local Attr.	×	×
CALLEE [20]	HIN Embedding + DBSCAN	99.76%	Not Report	Local Graph	×	×
SOCRATES	XGBoost + LLM	95.22%	F1: 91%	Global + Ext.	✓	✓

isting methods achieve significant workload reduction through statistical pattern matching or deep learning, they primarily operate as "black boxes" limited to local context. DEEPCASE utilizes temporal sequences but lacks semantic depth, and CALLEE achieves the highest volume reduction (99.76%) via graph clustering but assumes that structural similarity equates to malicious intent, which may fail in complex enterprise environments. In contrast, SOCRATES bridges this gap by leveraging LLMs for deep semantic understanding of alert payloads. Our system is the only one capable of incorporating global context, such as external threat intelligence and internal asset sensitivity, into the reasoning process. This allows SOCRATES to maintain a high F1-score of 0.91 while providing human-readable explanations, effectively reducing the cognitive load on SOC analysts beyond simple alert suppression.

5.4 RQ5: Industrial Deployment

By the time of this writing, SOCRATES has been deployed within the SOC of a large-scale enterprise for over one year, monitoring a network infrastructure consisting of more than 10,000 endpoint devices. During this extended deployment, the system has demonstrated exceptional stability under high-throughput conditions, processing an average of over one million raw alerts daily. By effectively filtering benign anomalies and aggregating redundant events, SOCRATES achieves a remarkable alert noise reduction rate of 99.6%, which has translated into a reduction in manual investigation effort by more than 80%. These real-world operational results confirm that SOCRATES successfully alleviates the critical issue of "alert fatigue", significantly enhancing both the velocity of threat response and the overall operational efficiency of the SOC.

6 Discussion

Architectural Paradigm and Strategic Value. The design of SOCRATES underscores a critical shift in automated security operations: the transition from static alert filtration to dynamic, context-aware investigation. Our research reveals that the primary bottleneck in applying LLMs to cybersecurity is not their inherent reasoning capability, but the scarcity of enterprise-specific context and the prohibitive cost of processing high-volume noise. By decoupling the suppression of repetitive business behaviors—handled efficiently by lightweight machine learning—from the semantic analysis of complex threats, we establish a scalable paradigm for AI-human collaboration. This hybrid architecture demonstrates that effective automation requires more than just powerful models; it demands an architectural alignment with the operational reality of SOCs, where understanding legitimate business logic is often a prerequisite for identifying genuine anomalies. By mimicking the tiered workflow of human analysts, the system bridges the gap between the speed of heuristic detection and the depth of cognitive reasoning.

Limitations. Despite these advancements, SOCRATES is subject to certain limitations that warrant future investigation. The efficacy of the Business Self-Learning module is contingent upon the availability of high-quality historical data and reliable analyst verdicts. In newly established SOC environments or rapidly evolving network infrastructures, the system may initially suffer from a "cold start" problem until a sufficient baseline of business behaviors is induced. Additionally, while the evidence-driven reasoning mechanism enhances interpretability, the upper bound of the LLM's decision-making quality is strictly defined by the completeness and freshness of the retrieved context. Fragmented logs or outdated threat intelligence can still lead to inconclusive or conservative judgments.

7 Conclusion

In this paper, we propose SOCRATES, an AI-powered SOC agent designed for automated and scalable network alert triage in large-scale enterprise environments. By innovatively utilizing a staged, coarse-to-fine pipeline that combines adaptive business logic learning with context-grounded LLM investigation, SOCRATES effectively minimizes alert fatigue while preserving high detection integrity. The successful long-term deployment in a production SOC and its proven track record in substantially reducing manual investigation effort further attest to its robustness and practical scalability. In conclusion, SOCRATES offers a promising paradigm for enhancing the efficiency and intelligence of modern security operations against the increasingly complex threat landscape.

A Ethical Considerations

This work investigates automated network alert triage in enterprise SOCs, a setting where errors and misuse may have operational and privacy implications. We adopt a stakeholder-based ethical analysis, considering SOC analysts, enterprise users, and partner organizations as primary stakeholders. The study relies on real-world enterprise security telemetry; all data are handled under strict access control and de-identification policies, and only aggregated statistics and sanitized examples are reported. To mitigate potential harm from misclassification, the proposed system is designed as a decision-support tool rather than a fully autonomous response system, with conservative filtering, evidence-grounded reasoning, and explicit support for human oversight. We do not make judgments about individuals, and the system's outputs are limited to alert-level assessments. Overall, we believe the benefits of reducing alert fatigue and improving SOC scalability outweigh the residual risks, given the conservative design and deployment assumptions.

B Open Science

To support reproducibility and future research, we have released the source code, data preprocessing scripts, and model configurations used in this study. However, due to strict privacy regulations and security non-disclosure agreements, the raw enterprise datasets, including network alert logs, detailed asset inventories, and historical incident tickets, cannot be publicly disclosed. To address this limitation while preserving the utility of our artifacts, we provide sanitized samples and mock-up data that reflect the schema and structure of the original dataset. All related materials are available at https://anonymous.4open.science/r/socagent.

C Prompt Templates

To ensure reproducibility and transparency, we provide the detailed prompt designs used in our experimental evaluation. We adopted a modular prompting framework, utilizing a fixed **Zero-Shot Baseline** as the foundation. Other strategies were implemented by injecting specific functional modules into this baseline:

- **Expertise Strategy**: Injects a domain knowledge module into the system prompt to define specific analysis markers.
- **Few-Shot Strategy**: Inserts authentic demonstration examples (one True Attack and one False Positive) before the user input to leverage in-context learning.
- **Chain-of-Thought (CoT)**: Modifies the output instruction to enforce a step-by-step reasoning process before the final verdict.

Figure 4 illustrates this incremental design, clearly marking the baseline components and the optional modules added for each strategy.

Modular Prompt Structure Construction

[System Prompt (Base: Zero-Shot)]

```
You are a senior network security analyst. Your task is
to analyze the following network alert logs.
```

+ Module A: Expertise (Injects Domain Rules)

```
# Simplified Markers for Analysis:
1. Attack Signatures: Look for known patterns like
SQL injection ('UNION SELECT'), directory traversal
('../')...
2. Benign Patterns: Identify standard business
behaviors...
3. Contextual Indicators: Consider response codes...
```

+ Module B: Few-Shot (Injects Demonstrations)

```
Here are two examples of how to analyze the alerts:
Example 1 (True Attack): Input: {...} Output: {...}
Example 2 (False Positive): Input: {...} Output: {...}
```

+ Module C: CoT (Modifies Output Requirement)

```
# Reasoning Requirement:
Before providing the classification, explicit
step-by-step analysis is required:
1. Analyze payload content... 2. Check status...
Output Format: <Step-by-step Reasoning> followed by
JSON.
```

[Output Instruction (Base)]

```
Determine if this is a "True Attack" or a "False
Positive". Return the result in JSON format.
```

[User Input (Base)]

```
Input: A JSON object containing alert metadata
(timestamp, source/dest IP, payload, etc.).
```

Figure 4: The modular prompt template design. The text directly on the background represents the Zero-Shot Baseline. The inner boxes represent the pluggable modules for Expertise, Few-Shot, and CoT strategies, which are inserted into the baseline sequence before the User Input.

References

[1] Bushra A Alahmadi, Louise Axon, and Ivan Martinovic. 99% false positives: A qualitative study of {SOC} analysts' perspectives on security alarms. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2783–2800, 2022.

[2] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, et al. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In

SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, pages 1–15. IEEE, 2022.

[3] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):186–205, 2000.

[4] Tao Ban, Ndichu Samuel, Takeshi Takahashi, and Daisuke Inoue. Combat security alert fatigue with ai-assisted techniques. In *Proceedings of the 14th Cyber Security Experimentation and Test Workshop*, pages 9–16, 2021.

[5] Mohan Baruwal Chhetri, Shahroz Tariq, Ronal Singh, Fatemeh Jalalvand, Cecile Paris, and Surya Nepal. Towards human-ai teaming to mitigate alert fatigue in security operations centres. *ACM Transactions on Internet Technology*, 24(3):1–22, 2024.

[6] Cisco. Snort. https://www.snort.org/, 2025.

[7] CrowdStrike. Powering the next evolution of the soc. https://www.crowdstrike.com/en-us/platform/charlotte-ai/, 2025.

[8] Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. {PentestGPT}: Evaluating and harnessing large language models for automated penetration testing. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 847–864, 2024.

[9] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 1107–1128, 2024.

[10] Chongzhou Fang, Ning Miao, Shaurya Srivastav, Jialin Liu, Ruoyu Zhang, Ruijie Fang, Ryan Tsang, Najmeh Nazari, Han Wang, Houman Homayoun, et al. Large language models for code analysis: Do {LLMs} really do their job? In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 829–846, 2024.

[11] Mohamed Amine Ferrag, Fatima Alwahedi, Ammar Battah, Bilel Cherif, Abdechakour Mechri, and Norbert Tihanyi. Generative ai and large language models for cyber security: All insights you need. *Available at SSRN 4853709*, 2024.

[12] Google. Supercharging security with generative ai. https://cloud.google.com/blog/products/identity-security/rsa-google-cloud-security-ai-workbench-generative-ai/, 2023.

[13] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. Nodoze: Combatting threat alert fatigue with automated provenance triage. In *network and distributed systems security symposium*, 2019.

[14] Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In *Findings of the association for computational linguistics: ACL 2023*, pages 1049–1065, 2023.

[15] Fatemeh Jalalvand, Mohan Baruwal Chhetri, Surya Nepal, and Cecile Paris. Alert prioritisation in security operations centres: A systematic survey on criteria and methods. *ACM Computing Surveys*, 57(2):1–36, 2024.

[16] Hamed Jelodar, Samita Bai, Parisa Hamedi, Hesamodin Mohammadian, Roozbeh Razavi-Far, and Ali Ghorbani. Large language model (llm) for software security: Code analysis, malware analysis, reverse engineering. *arXiv preprint arXiv:2504.07137*, 2025.

[17] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[18] Xihuan Lin, Jie Zhang, Gelei Deng, Tianzhe Liu, Xiaolong Liu, Changcai Yang, Tianwei Zhang, Qing Guo, and Riqing Chen. Ircopilot: Automated incident response with large language models. *arXiv preprint arXiv:2505.20945*, 2025.

[19] Peiyu Liu, Junming Liu, Lirong Fu, Kangjie Lu, Yifan Xia, Xuhong Zhang, Wenzhi Chen, Haiqin Weng, Shouling Ji, and Wenhai Wang. Exploring {ChatGPT's} capabilities on vulnerability management. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 811–828, 2024.

[20] Yu Liu, Tong Li, Runzi Zhang, Zhao Jin, Mingkai Tong, Wenmao Liu, Yiting Wang, and Zhen Yang. A context-aware clustering approach for assisting operators in classifying security alerts. *IEEE Transactions on Software Engineering*, 2024.

[21] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. Towards efficient generative large language model serving: A survey from algorithms to systems. *ACM Computing Surveys*, 58(1):1–37, 2025.

[22] OISF. Suricata. https://suricata.io/, 2025.

[23] Shuva Paul, Farhad Alemi, and Richard Macwan. Llm-assisted proactive threat intelligence for automated reasoning. *arXiv preprint arXiv:2504.00428*, 2025.

[24] Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, and Brendan Dolan-Gavitt. Examining zero-shot vulnerability repair with large language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2339–2356. IEEE, 2023.

[25] OWASP Modsecurity Project. Modsecurity. https://modsecurity.org/, 2025.

[26] Reza Sadoddin and Ali Ghorbani. Alert correlation survey: framework and techniques. In *Proceedings of the 2006 international conference on privacy, security and trust: bridge the gap between PST technologies and business services*, pages 1–10, 2006.

[27] Microsoft Security. Microsoft security copilot. https://www.microsoft.com/en-us/security/business/ai-machine-learning/microsoft-security-copilot/, 2025.

[28] Mahmood Sharif, Pubali Datta, Andy Riddle, Kim Westfall, Adam Bates, Vijay Ganti, Matthew Lentzk, and David Ott. Drsec: Flexible distributed representations for efficient endpoint security. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3609–3624. IEEE, 2024.

[29] Xiangmin Shen, Lingzhi Wang, Zhenyuan Li, Yan Chen, Wencheng Zhao, Dawei Sun, Jiashui Wang, and Wei Ruan. Pentestagent: Incorporating llm agents to automated penetration testing. *arXiv preprint arXiv:2411.05185*, 2024.

[30] Xiangmin Shen, Lingzhi Wang, Zhenyuan Li, Yan Chen, Wencheng Zhao, Dawei Sun, Jiashui Wang, and Wei Ruan. Pentestagent: Incorporating llm agents to automated penetration testing. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, pages 375–391, 2025.

[31] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[32] Sanchana Srikanth, Mohammad Hasanuzzaman, and Farah Tasnur Meem. Evaluating the usability of llms in threat intelligence enrichment. *arXiv preprint arXiv:2409.15072*, 2024.

[33] Shahroz Tariq, Mohan Baruwal Chhetri, Surya Nepal, and Cecile Paris. Alert fatigue in security operations centres: Research challenges and opportunities. *ACM Computing Surveys*, 57(9):1–38, 2025.

[34] Ankit Thakkar and Ritika Lohiya. A review on machine learning and deep learning perspectives of ids for iot: Recent updates, security issues, and challenges. *Archives of computational methods in engineering*, 28(4), 2021.

[35] Risto Vaarandi and Alejandro Guerra-Manzanares. Stream clustering guided supervised learning for classifying nids alerts. *Future Generation Computer Systems*, 155:231–244, 2024.

[36] Thijs Van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten Van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna. Deepcase: Semi-supervised contextual analysis of security events. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 522–539. IEEE, 2022.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[38] Mathew Vermeer, Michel Van Eeten, and Carlos Gañán. Ruling the rules: Quantifying the evolution of rulesets, alerts and incidents in network intrusion detection. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 799–814, 2022.

[39] Yiran Wu, Mauricio Velazco, Andrew Zhao, Manuel Raúl Meléndez Luján, Srisuma Movva, Yogesh K Roy, Quang Nguyen, Roberto Rodriguez, Qingyun Wu, Michael Albada, et al. Excytin-bench: Evaluating llm agents on cyber threat investigation. *arXiv preprint arXiv:2507.14201*, 2025.

[40] HanXiang Xu, ShenAo Wang, Ningke Li, Kailong Wang, Yanjie Zhao, Kai Chen, Ting Yu, Yang Liu, and HaoYu Wang. Large language models for cyber security: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 2024.

[41] Limin Yang, Zhi Chen, Chenkai Wang, Zhenning Zhang, Sushruth Booma, Phuong Cao, Constantin Adam, Alexander Withers, Zbigniew Kalbarczyk, Ravishankar K Iyer, et al. True attacks, attack attempts, or benign triggers? an empirical measurement of network alerts in a security operations center. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1525–1542, 2024.

[42] Xin Yin, Chao Ni, and Shaohua Wang. Multitask-based evaluation of open-source llm on software vulnerability. *IEEE Transactions on Software Engineering*, 2024.

[43] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.