RuleGenie: SIEM Detection Rule Set Optimization

Akansha Shukla, Parth Atulbhai Gandhi, Yuval Elovici and Asaf Shabtai

Beg-Gurion University of the Negev, Israel

Abstract. Security information and event management (SIEM) systems serve as a critical hub, employing rule-based logic to detect and respond to threats. Redundant or overlapping rules in SIEM systems lead to excessive false alerts, degrading analyst performance due to alert fatigue, and increase computational overhead and response latency for actual threats. As a result, optimizing SIEM rule sets is essential for efficient operations. Despite the importance of such optimization, research in this area is limited, with current practices relying on manual optimization methods that are both time-consuming and error-prone due to the scale and complexity of enterprise-level rule sets. To address this gap, we present RuleGenie, a novel large language model (LLM) aided recommender system designed to optimize SIEM rule sets. Our approach leverages transformer models' multi-head attention capabilities to generate SIEM rule embeddings, which are then analyzed using a similarity matching algorithm to identify the top-k most similar rules. The LLM then processes the rules identified, utilizing its information extraction, language understanding, and reasoning capabilities to analyze rule similarity, evaluate threat coverage and performance metrics, and deliver optimized recommendations for refining the rule set. By automating the rule optimization process, RuleGenie allows security teams to focus on more strategic tasks while enhancing the efficiency of SIEM systems and strengthening organizations' security posture. We evaluated RuleGenie on a comprehensive set of real-world SIEM rule formats, including Splunk, Sigma, and AQL (Ariel query language), demonstrating its platform-agnostic capabilities and adaptability across diverse security infrastructures. Our experimental results show that RuleGenie can effectively identify redundant rules, which in turn decreases false positive rates and enhances overall rule efficiency.

1 Introduction

In today's rapidly evolving digital landscape, enterprises must implement robust security measures to combat increasingly sophisticated cyber threats [20]. As organizations scale, their IT infrastructure correspondingly expands, encompassing a range of components such as networking devices, cloud computing environments, and endpoints. This expansion of IT infrastructure generates a diverse and vast stream of security events.

To cope with the increasing volume of security data, enterprises depend heavily on security information and event management (SIEM) systems [25]. SIEM systems are specifically designed to handle the challenges of processing diverse, high-volume data by aggregating events from multiple sources, each of which may use a unique vendor-specific schema. The ability of SIEM systems to standardize diverse data formats into a unified representation plays a vital role in modern enterprise security operations, enabling streamlined event analysis, storage, and incident response [9].

The operational core of SIEM architecture is anchored in its sophisticated rule-based system, where specialized rule engines serve as a critical bridge between raw security data and actionable intelligence [3]. This rule-based processing capability enables security operations center (SOC) analysts to detect and respond to threats more effectively. Therefore, the efficacy of a SIEM's detection and response capabilities depends on well-structured rule sets that eliminate redundancy, maximize coverage, and ensure maintainability.

Organizations continuously integrate new rules into their SIEM environment to adapt to emerging threats, accommodate new infrastructure components, and comply with regulatory requirements. While adopting an incremental approach to threat detection can enhance coverage, it also introduces significant operational challenges. As the amount of detection rules increases, overlapping logic, redundant conditions, and conflicting alerts may emerge, placing a burden on SOC analysts, who must filter through redundant alerts, a process consuming critical time and resources. The resulting alert fatigue can compromise analysts' ability to recognize genuine threats [2]. This increases the risk that serious security incidents will be undetected, threatening the organization's overall security posture.

Addressing theses challenges requires a systematic approach to optimizing rule sets and eliminating redundancies. Currently, SIEM system rule optimization relies heavily on manual efforts by analysts who must sift through entire rule sets to identify redundancies, merge similar rules, and remove unnecessary rules.

This process, which is both time-consuming and prone to human error, requires significant expertise and experience, resulting in the inefficient allocation of human resources and an increased workload for analysts. This ultimately affects the overall efficiency of cybersecurity operations. While both SQL and SIEM systems fundamentally deal with pattern matching and data filtering at scale, but the SIEM's operational contexts differ significantly. The extensive research performed on SQL query optimization underscores the significance of automated maintenance for optimized rule sets.

Traditional databases prioritize static, structured data, whereas SIEM systems manage real-time, heterogeneous security data streams, introducing complexities that extend beyond the scope of conventional SQL optimization methods. This unique technical distinction underscores the need for SIEM rule optimization, yet despite the critical operational importance of optimized SIEM rule sets in cybersecurity infrastructure, research in this domain remains notably limited.

To bridge this research gap, we present RuleGenie, a novel human-in-the-loop (HITL) system [21] that performs both retrospective and prospective analysis of SIEM rules. The proposed system retrospectively analyzes existing rule sets for potential redundancies while prospectively recommending optimizations when new rules are in-

troduced, thereby continuously ensuring rule set efficiency.

RuleGenie implements a robust pipeline to optimize and remove duplicate SIEM rules. It leverages an encoder-decoder transformer model [5] to generate embeddings of rules, enabling efficient syntactic representation. RuleGenie employs a distance-based similarity detection algorithm [23] to identify potentially redundant or overlapping rules in relation to a target rule in a rule set. The target rule may represent a newly proposed rule or an existing rule from the rule set being analyzed.

Each rule pair, consisting of the target rule and a potentially similar rule identified through similarity matching is then analyzed using a structured chain-of-thought (CoT) [29] reasoning process performed by an LLM. This CoT analysis is performed in three key steps. First, the LLM evaluates functional and semantic overlap between the rules. Second, it assesses platform specificity and hierarchical relationships between target environments. Finally, it compares the rules based on operational metrics including coverage, the false positive rate, and computational efficiency. By integrating these aspects of analysis, RuleGenie produces data-driven recommendations for rule set optimization, ensuring that no blind spots emerge in an organization's threat detection capabilities while maintaining SIEM systems' operational efficiency.

RuleGenie was assessed across diverse SIEM rule formats, including Sigma [4], Splunk [26], and AQL (Ariel query language) rule definitions. The evaluation encompassed 2,347 Sigma rules and 1,640 Splunk security content rules, with respectively 139 and 58 redundant rules demonstrating overlapping functionalities expressed via identical query structures or detection objectives. To validate the platform-agnostic nature of our approach, we converted the 2,347 Sigma rules into AQL security rules used by QRadar and examined RuleGenie's ability to identify redundant rule pairs across multiple SIEM environments. In addition, we evaluated RuleGenie using two LLMs: a proprietary model and a local open-source model. The proprietary model achieved over 90% recall and 65% precision across all SIEM rule formats. The local open-source model achieved over 80% recall and 75% precision across all SIEM rule formats, while also ensuring data privacy and reduced costs.

The main contributions of this paper can be summarized as follows:

- **A novel application of LLMs for SIEM rule set optimization:** We introduce a context-aware LLM-based approach for detecting similar rules and providing recommendations that reduces redundancies while maintaining high precision in evolving SIEM environments.
- **A code-based embedding technique for syntactic comparison of SIEM rules:** We propose the use of advanced code-based embedding techniques to enable syntactic comparisons of SIEM rules, ensuring more accurate and context-rich rule matching.
- **Real-world applicability:** We rigorously tested and validated RuleGenie in real-world SIEM environments, demonstrating its effectiveness in handling large-scale rule sets typical of enterprise deployments.

2 Related Work

To our knowledge, no prior work has directly addressed SIEM rule optimization, even though rule redundancy has long been recognized as a critical issue in cybersecurity.

Existing research in related areas such as firewall rule management, where research has shown that rule redundancy minimization significantly reduce noise and improve system efficiency [17, 18, 22]. Unlike firewall rules, which are relatively simple, SIEM rules involve intricate event aggregation, data transformation, and multi-source data correlation logic. Consequently, the optimization approaches that work well for firewall rules are often unsuitable for SIEM environments. But these studies reveal key insight that more security rules does not necessarily correlate with better security, but can instead introduce performance degradation and increase the likelihood of false positives. These methods, while effective for well-defined rule sets, face significant limitations when applied to the complex logic of SIEM rules.

Recent work has demonstrated the applicability of LLMs in a range of cybersecurity tasks, including log analysis [24, 15, 11, 19, 10, 6], and modeling threat behavior [8]. LLMs have shown the ability to understand and manipulate structured queries and configurations, including security rule formats like Sigma. However, their potential for automated rule optimization in operational security contexts particularly for identifying redundancy, semantic overlap, or logical inefficiencies in detection rules remains largely unexplored.

3 Methodology

3.1 RuleGenie Overview

The RuleGenie pipeline comprises three phases: rule embedding generation, similarity detection, and LLM analysis, as illustrated in Figure 1. In the first phase, SIEM rules are transformed into embedding representations in a high-dimensional vector to facilitate syntactical analysis. In the second phase, the vectorized embedding representations of target rule and existing rule set is compared using similarity matching algorithm. Once a target rule has been compared to an existing rule set, a cluster of potentially similar rules is generated. In the final phase, LLM-aided analysis is performed to evaluate both the target rule and the cluster of similar rules.

The RuleGenie pipeline leverages transformer models for SIEM rule vectorization and uses LLMs for semantic analysis.

3.2 Rule Embedding Generation

In the initial phase, RuleGenie utilizes a transformer-based architecture to generate rule embeddings. This embedding process forms the cornerstone of the pipeline, ensuring that downstream tasks benefit from comprehensive and context-rich representations of SIEM rules and enhancing the overall pipeline's effectiveness. In the subsections below, we describe the model selection and validation framework, implementation architecture, and overflow management pipeline, all of which are employed when generating transformer embeddings.

3.2.1 Model Selection and Validation

We conducted a comparative analysis of state-of-the-art transformer models to identify the most suitable architecture for encoding SIEM rule syntax. The evaluation included general-purpose language models (BERT [7]), security-specific model (SecBERT [13]), and code-oriented models (CodeT5 [28]).

The vector distance analysis between redundant Sigma rules demonstrated that code-based models achieve superior performance in maintaining vector proximity for syntactically similar rules. CodeT5 emerged as the optimal choice for our pipeline based on several key attributes:

1. **Architecture design:** Its pretrained encoder-decoder transformer architecture, derived from T5, excels at both comprehension and generation tasks.

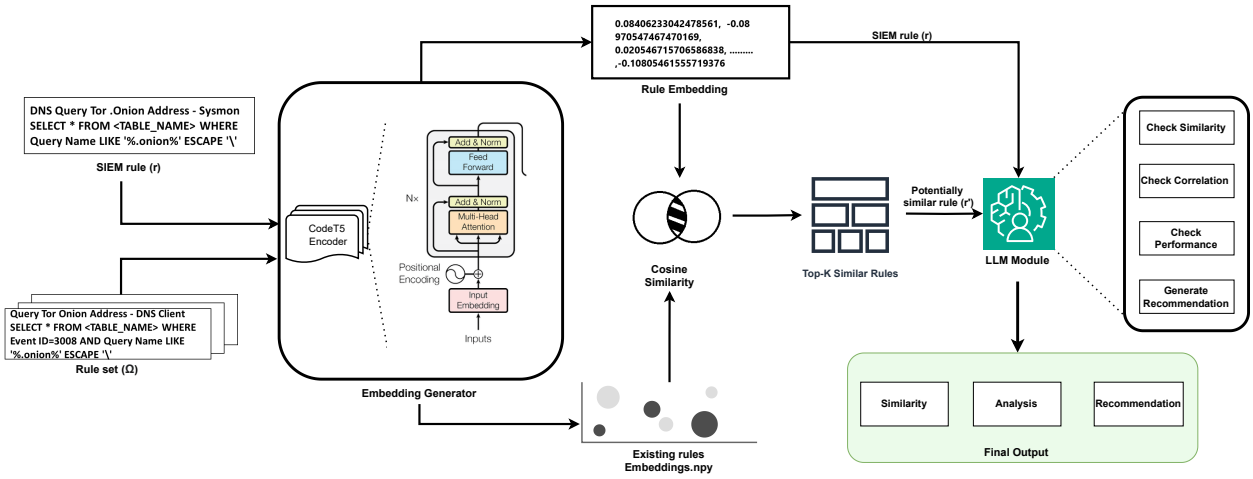

Figure 1. An overview of RuleGenie's three-phase pipeline.

2. **Token processing:** Its identifier-aware training objective enables precise token distinction and recovery.
3. **Dual-modality training:** Its training on both code and natural language facilitates effective alignment between natural and programming language constructs.

3.2.2 *Implementation Architecture*

Our embedding framework employs an adaptive segmentation strategy coupled with a conditional processing pipeline to accommodate SIEM rules of varying complexity while ensuring consistent vector representations. The architecture implements two processing mechanisms based on input dimensionality:

- *Standard processing mechanism.* The RuleGenie pipeline uses an advanced segmentation approach to handle SIEM rules that exceed standard token length limits (512-4096 tokens). The system employs a sophisticated chunking mechanism designed to preserve logical boundaries within each segment. After segmentation, each partition is processed independently through the encoder, which captures the salient features of the input. These encoded segments are then recombined to preserve the relationships between chunks and maintain the overall context of the rule.
- *Overflow management mechanism.* To maintain models' robustness when processing exceptionally complex SIEM rules (exceeding 4096 tokens), we employ a graceful degradation [12] protocol that initiates a controlled fallback to a near null-space representation. In critical security contexts, these instances are flagged for optional manual review.This overflow management mechanism ensures that system integrity is upheld even under extreme load conditions.

This dual-architecture for embedding generation ensures consistent processing capabilities across SIEM rules of varying complexity while preserving the dimensional structure of the vector space representation. The system design prioritizes both processing efficiency and representation uniformity, making it well-suited for security operations.

3.3 *Similarity Detection*

In this phase of the pipeline, cosine similarity [16] is used to quantify the syntactic relationships between SIEM rule embeddings.

Cosine similarity was chosen as the metric, because it is agnostic to vector magnitude, making it particularly robust for comparing embeddings of varying complexities.

To operationalize this similarity measure, we implement a top-k retrieval mechanism (in our evaluations we set $k = 5$) for each SIEM rule. The selection of k and its impact on precision and recall is systematically analyzed in Section 4.3, where we present empirical evidence supporting this choice through an ablation study.

Additionally, our experimental investigation of transformer selection reveals that the embedding space exhibits interesting topological properties, where syntactical related rules form well-defined clusters.

3.4 *Redundancy Analysis Using LLM*

The last phase of our framework leverages an LLM enhanced with CoT reasoning to perform fine-grained semantic and functional analysis of security rules. We initially used GPT-4o [14], which demonstrated exceptional performance in understanding complex security rule semantics and generating nuanced recommendations. However, operational constraints—particularly data privacy requirements and cost considerations necessitated the exploration of alternative solutions. Our subsequent evaluation of open-source models focused on three criteria: (1) context length capacity for handling extensive rule descriptions; (2) analytical capabilities for semantic analysis; and (3) computational efficiency and resource optimization.

After testing multiple candidates, including Llama 8B fine-tuned [27] and Qwen models [30], we selected Qwen-2.5-14B-Instruct as our primary LLM. This model demonstrated superior performance in our specific security rule analysis tasks, offering an optimal balance between computational efficiency and analytical capability.

To maximize the effectiveness of the Qwen-2.5-14B-Instruct model, we developed prompting strategies that leverage its context understanding capabilities and a standardized JSON response format that structures the model's outputs across all analytical stages while maintaining computational efficiency. This structured approach ensures consistent, machine-readable outputs while preserving the so-

phisticated analytical capabilities required for security rule optimization.

The LLM-driven redundancy analysis consists of four stages which are described below.

3.4.1 Deep semantic analysis and functional mapping

The process begins with the LLM evaluating a pair of rules (r, r'), where r represents the current rule and r' represents a candidate from the k-similar neighbors identified using the similarity matching algorithm. The LLM performs semantic analysis, considering both syntactic structure and functional intent. The rules are compared by LLM for structural matching, semantic equivalence and conditional overlap necessary to establish whether r and r' exhibit a significant degree of similarity. This analysis yields two key outputs: (a) a boolean classification indicating semantic and functional overlap; and (b) similarity score $s \in [0, 100]$, where 0 indicates no similarity and 100 indicates perfect similarity.

Through evaluation on Sigma rules, we established a minimum similarity threshold of 75, which effectively identifies significant rule overlaps while minimizing false positives as discussed in Section 4.3.4. Rules with $s \geq 75$ proceed to subsequent analysis stages, while others are pruned to maintain computational efficiency and ensure meaningful comparisons.

3.4.2 Hierarchical dependence recognition

For qualifying rule pairs, we employ a novel hierarchical relationship detection algorithm that examines both structural and semantic dependencies. The analysis focuses on identifying three primary relationship patterns:

1. **Platform-specific independence**, where rules serve similar security objectives but operate within distinct technological contexts.
2. **Generalization relationships**, where one rule subsumes or extends another functionality. For example, consider the Splunk rules Abnormally High AWS Instances Terminated by User (AWS-specific) and Abnormally High Cloud Instances Terminated by User (covers AWS, Azure, GCP, etc.). Here, the second rule generalizes the first by applying the same detection logic across multiple cloud providers.
3. **Cross-platform dependencies**, which reveal potential optimization opportunities across different security platforms.

This hierarchical mapping enables sophisticated rule organization strategies and forms the foundation for optimization decisions.

3.4.3 Performance and quality optimization

In this stage a comprehensive evaluation framework for dependent rule pairs is implemented to analyze their operational effectiveness. Comparative evaluation is conducted across three performance metrics:

1. **Coverage Analysis**: The scope and comprehensiveness of each rule's detection capabilities are assessed. Broader coverage is prioritized.
2. **Efficiency Metrics**: Execution times and resource consumption are evaluated to minimize operational overhead.
3. **False Positive Mitigation**: The potential for false positive generation is rigorously assessed based on three key aspects: conditional statement scope, filtering criteria, and validation logic implementation. Each rule is further analyzed for its effectiveness in minimizing blind spots and ensuring robust validation logic. After evaluating both the target rule r and candidate rule r' against these criteria, the results are compared to identify the superior rule, with a preference for rules that maintain high accuracy while reducing noise.

3.4.4 Rule recommendation for SIEM environments

Integrating insights from semantic analysis, hierarchical dependency mapping, and performance evaluation, the final stage of RuleGenie focuses on generating finely tuned recommendations. By carefully balancing trade-offs between conflicting metrics such as efficiency and accuracy, RuleGenie provides systematic and targeted strategies for optimizing SIEM rules.

For each rule pair analyzed, recommendations are generated based on their relationship:

1. **Keep superior rule:** If one rule demonstrates superior coverage and false positive handling, the recommendation is to retain the better-performing rule.
2. **Merge complementary rules:** If a rule pair is similar but not entirely redundant and could offer enhanced coverage when combined, RuleGenie recommends merging the rules.
3. **Keep both rules:** For rule pairs that are clearly distinct and serve different detection purposes, the recommendation is to retain both rules.

This structured framework enables precise reasoning that improves rule relevance and operational effectiveness, addressing the challenges of modern cybersecurity environments and resulting in a high-confidence rule deployment strategy designed to be robust, adaptable, and contextually aligned with operational requirements. This ensures that recommendations are not only balanced and actionable but also tailored to the dynamic demands of security operations, fostering advanced threat detection and mitigation.

4 Evaluation

4.1 Experimental Setup

RuleGenie was implemented using Python 3.11. We preprocess SIEM rules to extract key features. Embeddings for these rules are generated using the encoder component of Salesforce's CodeT5 model. We employed two large language models for rule similarity comparison, performance analysis, and recommendation generation: OpenAI's GPT-4o, accessed via the Azure OpenAI API, and a local deployment of Qwen-2.5-14B-Instruct, managed using the Hugging Face library. The locally deployed Qwen model can run efficiently on a single NVIDIA RTX 3080 GPU.[1]

4.2 Dataset

To address security considerations surrounding sensitive SIEM infrastructure, we limited our experimental analysis to detection rules sourced from public security repositories. Our data acquisition framework leveraged two prominent open-source security initiatives: the Sigma detection framework [4] and Splunk Security Content repository [26]. The integration of these complementary sources yielded 3,987 detection rules, which we methodically pre-classified into three distinct categories: (1) foundational detection rules for baseline security monitoring, (2) advance threat hunting rules, and

[1] Code will be made available once the paper is accepted.

(3) emerging threat rules targeted at time-critical vulnerabilities and exploits.

For the initial evaluation of RuleGenie pipeline Splunk and Sigma rules were used. Through rigorous analysis of 2,347 Sigma rules, 130 Windows-specific rules were identified as new, and the other 2,217 were incorporated in the existing rule set. The Splunk Security Content repository contributed an additional 100 new rules, complemented by 1,540 existing rules. To validate the platform-agnostic nature of our methodology, we converted all 2,347 Sigma rules into AQL queries. We utilized pySigma [1], a Python package maintained by the Sigma repository authors, to convert the Sigma rules into both AQL and Splunk query formats. which are methodically preclassified into three categories: (1) foundational detection rules for baseline security monitoring, (2) advance threat hunting rules, and (3) emerging threat rules targeted at time-critical vulnerabilities.

We partitioned the dataset to simulate real-world task of integrating new detections into an existing SIEM environment. We manually curated 230 rules (130 Sigma, 100 Splunk) representing recent additions typically seen in production environments. These form the *new* evaluation set, while the remaining 3,757 detections constitute the *existing* rule set for comparison. To assess platform-agnostic capabilities, all Sigma detections were programmatically converted into both AQL and Splunk query formats using pySigma [1], a Python package maintained by the Sigma repository authors.

Table 1. Comparison of SIEM rule sets in terms of redundancy and new rules.

SIEM Rule Set	Non-Redundant Existing Rules	Non-Redundant New Rules	Redundant Existing Rules	Redundant New Rules
Sigma	2167	41	50	89
Splunk	1517	65	23	35
AQL	2167	41	50	89

This comprehensive classification and conversion process facilitated the development of a robust input dataset analysis and experimentation. Table 1 categorizes SIEM rules into redundant and non-redundant groups across Sigma, Splunk, and AQL datasets, covering both newly introduced and existing rules.

4.3 Ablation Study

This subsection details the empirical evaluation used to optimize the configuration of RuleGenie. We present the selection process for key components, including: the embedding generation model, the similarity matching parameter (k), the operational similarity threshold, and the large language model for recommendations. We utilize standard precision and recall metrics throughout this evaluation since established SIEM rule optimization benchmarks are not available.

4.3.1 Embedding Model Selection

We evaluated three transformer models for generating SIEM rule embeddings: the general-purpose BERT, the security-focused SecBERT, and the code-focused CodeT5. Figure 2 illustrates the performance differences between these models in identifying redundant SIEM rules. The figure displays a 2D representation of the high-dimensional rule embeddings generated by each model, obtained using principal component analysis (PCA).The embedding space projections, has target rule under evaluation marked in red, while the rule that is redundant to target rule is represented in green.The visualizations show CodeT5 embeddings place the redundant rule significantly closer to target rule compared to BERT and SecBERT embeddings.

This closer proximity indicates that CodeT5 produces embeddings that better capture the similarities relevant for identifying redundancy specifically within SIEM rule syntax and structure. CodeT5's effectiveness likely stems from its pre-training on programming language structures and keywords, which align well with typical SIEM rule syntax, enabling it to represent rule features more effectively for this task than the general-purpose or security-text-focused models.

4.3.2 Value selection for k

The optimal value of k was empirically determined through analysis of precision-recall trade-offs across different k values for the Sigma rule set as shown in Figure 3. The results demonstrate that the recall metric values consistently improved as k increases from one to five, with recall values showing substantial improvement from 0.382 to 0.966, representing a 152.94% increase. Similarly, precision showed a generally positive trend up to $k = 5$, starting at 0.810 for $k = 1$ and reaching its peak of 0.887 at $k = 5$, representing a 9.5% improvement. This simultaneous enhancement in both precision and recall metrics up to $k = 5$ indicates strong classifier performance.

While the recall metric values plateau at 0.966 for $k \geq 5$, precision begins to deteriorate for $k > 5$, dropping to 0.843 at $k = 10$. This represents a 4.9% decrease in precision with no corresponding gain in recall, suggesting that $k = 5$ represents an optimal balance point. The deterioration in precision beyond $k = 5$ can be attributed to the inclusion of more distant neighbors in the classification decision, which may introduce noise and reduce the classifier's ability to maintain clear decision boundaries.

4.3.3 LLM model selection

While our analysis confirms that embeddings generated by CodeT5 effectively capture syntatic patterns and locality in SIEM rules with rules sharing similar syntax clustering together (see Figure 2), relying solely on embedding similarity presents several limitations. This syntactic matching lacks a deeper understanding of rule semantics and operational context, risking false positive recommendations or the removal of rules with unique defensive value, potentially creating security blind spots.

To overcome these limitations, we leverage LLMs trained on both natural language and cybersecurity domain knowledge. These models offer a more nuanced interpretation of SIEM rule semantics, considering technical implementation details alongside the security implications of potential modifications. This approach aims to enhance the quality and safety of rule recommendations.

We evaluated three candidate LLMs:

First, we assessed OpenAI's GPT-4o via the Azure API. While demonstrating high accuracy in identifying redundancies and providing contextually relevant recommendations, its operational cost was prohibitive. Processing our complete rule set incurred an estimated cost of $142.5 per analysis iteration (based on standard API pricing of $0.03/1K input tokens and $0.06/1K output tokens). Additionally, the average processing time of 70 minutes made it unsuitable for frequent or large-scale use.

Given the cost constraints of GPT-4o, we next investigated smaller models to assess if a more cost-effective open-source solution could be viable. We experimented with Llama 3.1 8B as a lightweight open-source alternative. However, the base model exhibited a high hallucination rate (70%) when interpreting rule semantics and operational context. Subsequent fine-tuning yielded only marginal performance

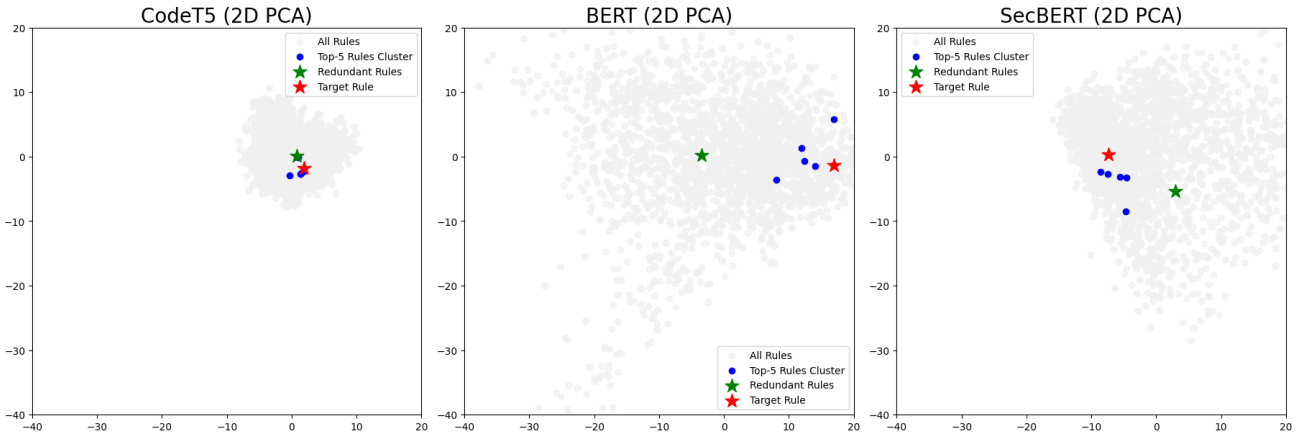

Figure 2. An example of a randomly selected rule. Sigma rule *"New Service Uses Double Ampersand in Path"* embedding using transformer models.

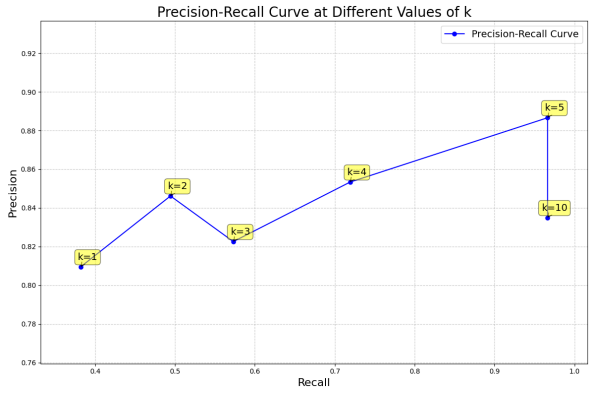

Figure 3. Precision-recall curve for k in Sigma rule set.

improvements. The model's non linear time complexity proved computationally inefficient for enterprise-scale deployments, particularly when processing large rule sets. This computational constraint, coupled with persistent hallucination issues, rendered the model unsuitable for production environments requiring reliable rule analysis.

Finally, our evaluation focused on Qwen-2.5-14B-Instruct deployed locally. This model demonstrated performance comparable to GPT-4o in identifying relevant rules while satisfying crucial constraints of data privacy via on-premises deployment and operational cost. Specifically, in our rule recommendation task, Qwen achieved a minimum precision of 72%, and recall of 80% (see Table 4). These performance metrics, combined with the practical advantages of local deployment, establish Qwen-2.5-14B-Instruct as the most viable LLM solution for RuleGenie within an enterprise context.

4.3.4 *Similarity Threshold Selection*

To determine the optimal similarity threshold for identifying redundant SIEM rules, we evaluated multiple threshold values using the Sigma rule set, analyzing the resulting precision and recall for each candidate.

As detailed in Table 2, the analysis revealed the expected trade-off: lower thresholds (e.g., 65) increased recall at the cost of precision, while higher thresholds (e.g., 85) improved precision but reduced recall. Based on these results, we selected a threshold of 75 as providing the optimal balance. This value is therefore used as the operational similarity threshold in RuleGenie.

Table 2. Performance of RuleGenie for different similarity score.

Similarity Threshold	Precision	Recall
65	0.454	0.733
75	0.818	0.733
85	0.840	0.635

4.4 *Evaluation Metrics*

Our evaluation assesses the performance of RuleGenie pipeline in two stages: (a) the identification of potentially similar or redundant rules, and (b) the quality of the optimization recommendations provided for the identified rules. We evaluate the system's output against a ground truth established through expert analyst review.

To quantify RuleGenie's performance, we utilize metrics that reflect both its ability to accurately detect redundancy and the correctness of its recommendations. While we adopt the terminology of precision and recall for evaluating our method, we define these metrics in a task-specific manner due to the absence of prior baselines.

First, we measure the RuleGenie's ability to identify truly redundant rules using Recall, consistent with its standard definition in classification tasks. Recall quantifies the proportion of actual redundant rules in the dataset that our system successfully identifies:

$$\text{Recall} = \frac{\text{\# of correctly identified redundant rules}}{\text{\# of correctly identified redundant rules} + \text{\# of missed redundant rules}} \tag{1}$$

Second, to quantify the quality of the optimization recommendations provided by RuleGenie for the rules it identifies, we define Precision. This metric specifically measures the proportion of recommendations that were validated as correct by human analysts based on three critical matrices: (a) detection coverage; (b) false positive rate reduction; and (c) computational efficiency, out of all the rules the system flagged as potentially redundant:

$$\text{Precision} = \frac{\text{\# of correct recommendations}}{\text{\# of correct recommendations} + \text{\# of incorrect recommendations}} \tag{2}$$

By quantifying both Precision and Recall, we aim to provide a comprehensive view of the pipeline's effectiveness in supporting human analysts in SIEM rule set optimization.

5 Results

5.1 Adding new rules to an existing rule set

Our experimental evaluation, the results of which are summarized in Table 3, examines the performance of various LLMs in generating security rule recommendations for newly implemented SIEM rules. The results show that Qwen-2.5-14B-Instruct outperforms GPT-4o in terms of precision across all evaluated rule sets, with comparable recall performance between the two models.

Table 3. Adding new rules to an existing rule set. Comparison of models on different rule sets in terms of precision and recall. Bold numbers indicate best performance and underlined numbers indicate second-best performance.

Model	SIEM Rule Set	Precision	Recall
GPT-4o	Sigma	0.886	**0.966**
	Splunk	0.673	**1.000**
	AQL	0.886	**0.966**
Qwen-2.5-14B-Instruct	Sigma	**0.941**	0.910
	Splunk	**0.795**	0.886
	AQL	**0.941**	0.910
Llama 3.1 8B-Instruct Fine-tuned	Sigma	0.450	0.483
	Splunk	0.315	0.398
	AQL	0.450	0.483

5.2 Optimizing existing rule set

Based on the cost analysis presented in Section 4.3, which revealed significant operational expenses associated with GPT-4o, and the high hallucination rates observed in Llama 3.1 8B, we selected Qwen-2.5-14B-Instruct to optimize the existing SIEM rule set. The results of our evaluation of Qwen-2.5-14B-Instruct's performance in rule set optimization are presented in Table 4 and demonstrate its effectiveness as a cost-efficient and reliable alternative.

Table 4. Optimizing existing rule set. Performance of Qwen-2.5-14B-Instruct on different rule sets.

Model	SIEM Rule Set	Precision	Recall
Qwen-2.5-14B-Instruct	Sigma	0.947	0.850
	Splunk	0.726	0.804
	AQL	0.947	0.850

6 Discussion

6.1 Impact of Embedding Generation and Similarity Matching

This study aimed to investigate the effectiveness and efficiency of RuleGenie for identifying similar rules within a rule set. A key objective is to quantify the contribution of each phase.

To evaluate the impact of our embedding-based pre-processing strategy, we conducted a comparative analysis between two workflows as summarized in Table 5:

- **Baseline Workflow:** Brute-force analysis of SIEM rules, requiring pairwise comparisons across the entire rule set without initial filtering or dimensionality reduction.
- **Proposed Pipeline:** Incorporates the following pre-processing steps: (a) generation of vector embeddings for each rule, (b) retrieval of the top-5 most similar rules via cosine similarity and (c) run downstream analysis exclusively on these top candidate rules

Table 5. Performance comparison between baseline and proposed pipeline (Qwen-2.5-14B-Instruct).

Workflow	Sample Size	Total Time (min)	Avg. Time per Rule (min)	Throughput (rules/min)	Speed-up (per-rule)
Baseline (brute force)	7	224	32.0	0.22	–
Proposed pipeline (embedding + top-5)	100	40	0.4	2.50	80×

The baseline experiment was limited to a sample size of 7 rules due to the prohibitive computational cost of a full 100-rule brute-force analysis. An estimated $100 \times 32.0 = 3200$ min (approximately 53.3h) would have been required, rendering it infeasible within practical time constraints.

The performance improvements shown in Table 5 derive from embedding-based dimensionality reduction, which compacts the search space, and from similarity pruning, which restricts comparisons to the top five most similar rules thereby mitigating the quadratic time complexity of brute-force, all-pairs evaluation.

6.2 Enhancing Analysis Quality with Chain-of-Thought

The second study evaluated the impact of integrating a CoT prompting strategy compared to a naive single-prompt baseline. The results shown in Table 6 shows the dramatic increase in evaluation metrics of recall and precision discussed in Section 4.4.

The stepwise reasoning inherent in the CoT methodology enables the model to better understand nuanced relationships between rules, particularly where redundancy stems from semantic equivalence rather than syntactic similarity, thus providing better recommendations. This effectiveness stems from the principle discussed by [29] that decomposing complex tasks, into simpler steps allows LLMs to process information more reliably and accurately.

Table 6. Performance of baseline and chain-of-thought prompts (Qwen-2.5-14B-Instruct).

Prompt Type	Precision	Recall
Single prompt	0.250	0.533
CoT prompt	0.818	0.733

In conclusion, the combined contributions of the efficient preprocessing and the sophisticated reasoning are essential components of the RuleGenie pipeline. The first phase addresses the critical challenge of scalability and computational cost while, second phase addresses the equally important challenge of analytical accuracy and the quality of the recommendations. Together, these phases create a robust and effective system for large-scale rule optimization.

7 Conclusion and Future Work

In this paper, we introduced RuleGenie, a novel HITL method that leverages LLMs generated recommendations to optimize SIEM rule sets. Our experimental results demonstrate that RuleGenie can streamline SOC workflows by minimizing multiple redundant alerts through intelligent rule recommendation.

Building on our findings, future research will focus on two key dimensions: automated implementation of LLM-generated recommendations and optimization of rule execution sequences. This dual approach aims to enhance both the accuracy and computational efficiency of SIEM infrastructure. By developing intelligent orchestration mechanisms for rule execution, we anticipate achieving im-

proved detection capabilities while minimizing system resource utilization. This advancement could significantly enhance an organization's security posture by establishing a more responsive and resource-efficient security monitoring framework.

References

[1] Sigma Rules pySigma documentation, 2021. URL https://sigmahq-pysigma.readthedocs.io/en/latest/. Last accessed 05-05-2025.

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

[3] S. Bhatt, P. K. Manadhata, and L. Zomlot. The operational role of security information and event management systems. *IEEE security & Privacy*, 12(5):35–41, 2014.

[4] S. contributors. Sigma – generic signature format for siem systems. https://github.com/SigmaHQ/sigma/, 2023. Accessed: 2023-01-06.

[5] E. Egonmwan and Y. Chali. Transformer-based model for single documents neural summarization. In A. Birch, A. Finch, H. Hayashi, I. Konstas, T. Luong, G. Neubig, Y. Oda, and K. Sudoh, editors, *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 70–79, Hong Kong, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5607. URL https://aclanthology.org/D19-5607/.

[6] P. A. Gandhi, P. N. Wudali, Y. Amaru, Y. Elovici, and A. Shabtai. Shield: Apt detection and intelligent explanation using llm, 2025. URL https://arxiv.org/abs/2502.02342.

[7] P. Ganesh, Y. Chen, X. Lou, M. A. Khan, Y. Yang, H. Sajjad, P. Nakov, D. Chen, and M. Winslett. Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association for Computational Linguistics*, 9:1061–1080, 2021.

[8] E. Garza, E. Hemberg, S. Moskal, and U.-M. O'Reilly. Assessing large language model's knowledge of threat behavior in mitre att&ck. KDD, 2023.

[9] G. González-Granadillo, S. González-Zarzosa, and R. Diaz. Security information and event management (siem): analysis, trends, and usage in critical infrastructures. *Sensors*, 21(14):4759, 2021.

[10] H. Guo, S. Yuan, and X. Wu. Logbert: Log anomaly detection via bert. In *2021 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2021.

[11] X. Han, S. Yuan, and M. Trabelsi. Loggpt: Log anomaly detection via gpt, 2023. URL https://arxiv.org/abs/2309.14482.

[12] M. P. Herlihy and J. M. Wing. Specifying graceful degradation. *IEEE Transactions on Parallel and Distributed Systems*, 2(1):93–104, 1991.

[13] H. Huang and Y. Wang. Secbert: Privacy-preserving pre-training based neural network inference system. *Neural Networks*, 172:106135, 2024.

[14] R. Islam and O. M. Moushi. Gpt-4o: The cutting-edge advancement in multimodal llm. *Authorea Preprints*, 2024.

[15] Y. Lee, J. Kim, and P. Kang. Lanobert : System log anomaly detection based on BERT masked language model. *CoRR*, abs/2111.09564, 2021. URL https://arxiv.org/abs/2111.09564.

[16] B. Li and L. Han. Distance weighted cosine similarity measure for text classification. In *Intelligent Data Engineering and Automated Learning–IDEAL 2013: 14th International Conference, IDEAL 2013, Hefei, China, October 20-23, 2013. Proceedings 14*, pages 611–618. Springer, 2013.

[17] A. X. Liu and M. G. Gouda. Complete redundancy removal for packet classifiers in tcams. *IEEE Transactions on Parallel and Distributed Systems*, 21(4):424–437, 2008.

[18] A. X. Liu, C. R. Meiners, and Y. Zhou. All-match based complete redundancy removal for packet classifiers in tcams. In *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, pages 111–115. IEEE, 2008.

[19] Y. Liu, S. Tao, W. Meng, J. Wang, W. Ma, Y. Zhao, Y. Chen, H. Yang, Y. Jiang, and X. Chen. Interpretable online log analysis using large language models with prompt strategies, 2024. URL https://arxiv.org/abs/2308.07610.

[20] H. Ltd. Cyber readiness report 2024. Technical report, Hiscox Ltd., 2024. URL https://www.hiscoxgroup.com/sites/group/files/documents/2024-10/HSX245%20%E2%80%93%20%202024%20CRR.pdf.

[21] D. S. Nunes, P. Zhang, and J. S. Silva. A survey on human-in-the-loop applications towards an internet of all. *IEEE Communications Surveys & Tutorials*, 17(2):944–965, 2015.

[22] F. Persson. Optimization of rulesets with reinforcement learning, 2020.

[23] G. Qian, S. Sural, Y. Gu, and S. Pramanik. Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1232–1237, 2004.

[24] D. Saha, S. Tarek, K. Yahyaei, S. K. Saha, J. Zhou, M. Tehranipoor, and F. Farahmandi. Llm for soc security: A paradigm shift. *IEEE Access*, 2024.

[25] M. Sheeraz, M. A. Paracha, M. U. Haque, M. H. Durad, S. M. Mohsin, S. S. Band, and A. Mosavi. Effective security monitoring using efficient siem architecture. *Hum.-Centric Comput. Inf. Sci*, 13:1–18, 2023.

[26] Splunk. Splunk security content homepage, n.d. URL https://github.com/splunk/security/content.

[27] R. Vavekanand and K. Sam. Llama 3.1: An in-depth analysis of the next-generation large language model, 2024.

[28] Y. Wang, W. Wang, S. Joty, and S. C. Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*, 2021.

[29] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[30] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.