GARNET: GoT-Based Alert Reduction and Narrative Event Tracing

Yiru Gong[1,2], Song Liu[1,2], Changzhi Zhao[1,2], Junrong Liu[1,2], Tian Tian[1,2], Xiaobo Yang[1,2], Bo Jiang[1,2*], Zhigang Lu[1,2]

[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{gongyiru, liusong1106, zhaochangzhi, liujunrong, tiantian, yangxiaobo, jiangbo, luzhigang}@iie.ac.cn

Abstract

Alerts generated by Security Operations Centers (SOCs) are often numerous and scattered, requiring significant effort from security analysts to manage, which severely slows response times. While recent alert correlation graph methods can effectively reduce alert volume, these graphs are often too complex for analysts to understand. As a result, analysts are increasingly seeking ways to automatically correlate alerts and generate concise, human-readable attack path summaries. Recently, Large Language Models (LLMs) have demonstrated superior performance due to their advanced capabilities in knowledge reserve and reasoning. In this work, we propose GARNET, a framework that uses LLMs for reasoning on alert correlation graphs. GARNET addresses three key technical challenges: 1) modality alignment between alert graphs and logs; 2) semantic alignment between alert graphs and logs; 3) enabling LLMs reasoning along graph paths. Specifically, we first project the embeddings of the graph and logs into the same vector space using contrastive learning. Then, we design self-supervised graph-log instructions to bridge the semantic gap between the graph and logs by training a novel LLM. Finally, GARNET uses a novel Graph-of-Thought (GoT)-based interaction reasoning approach to guide LLM reasoning along graph paths, ultimately generating structured, concise, and human-readable attack path summaries. Experimental results across six attack scenarios show that GARNET reduces false positives by an average of 80%, lowering the false positive rate to below 0.0037. It outperforms the latest approaches and provides more explainable attribution.

Introduction

In modern Security Operations Centers (SOCs), analysts have to handle a large volume of security alerts, many of which are false positives. A qualitative study (Alahmadi, Axon, and Martinovic 2022) reports that approximately 99% of alerts are false positives, leading to alert fatigue from the constant need to verify which alerts are from real attacks, severely delaying the response to the attack. Alert reduction is a tough task for several reasons. First, as different security tools detect various aspects of the same event, an attack can trigger a cascade of scattered alerts. Correlating these alerts to focus on a single attack is difficult. Second, alert analysis is labor-intensive. Evaluating alerts and providing coherent explanations consumes a significant amount of time and effort from analysts.

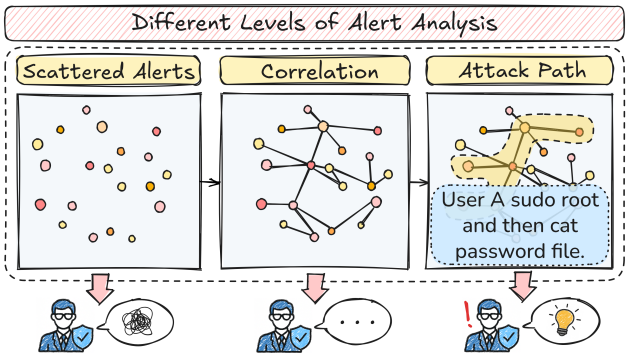


Figure 1: The numerous and scattered alerts generated by different detectors often confuse security analysts. Recent methods use graphs to associate alerts, providing a clearer attack causality. However, these graphs remain overly complex. If the attack paths could be clearly described, it would offer significant assistance to analysts.

Recent research has focused on correlating scattered alerts using graph-based structures (Hassan, Bates, and Marino 2020; Hossain et al. 2017; Milajerdi et al. 2019; Hassan et al. 2019). They establish causal relationships between alerts by constructing alert correlation graphs, providing a global view of attack activities. However, these graphs are usually complex, with numerous security entities and interaction dependencies, requiring additional manual attribution analysis and explanation. How to automatically correlate alerts and generate concise, human-readable attack path summaries becomes an urgent need, as shown in Figure 1.

Recently, Large Language Models (LLMs) have made success in reasoning tasks across various domains (Jin et al. 2023; Ullah et al. 2024; Chen et al. 2024). It offers powerful knowledge and reasoning capabilities, demonstrating a strong grasp of context. Additionally, LLMs can generate human-readable explanations in natural language, enabling non-security experts to find out the reasoning behind alerts.

*Corresponding author
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

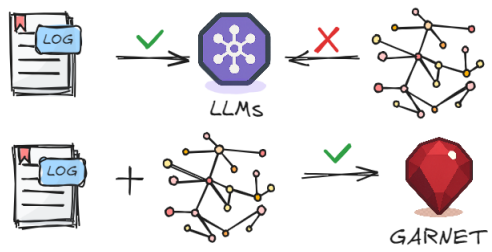


Figure 2: Since LLMs are designed to process natural language, they cannot directly understand graphs. Our framework, GARNET, is capable of understanding both log text and graph structure.

In this work, we intend to use a framework that uses LLMs for reasoning on alert correlation graphs. Nevertheless, this introduces three key technical challenges:

- **CH1**: How to handle the different modalities of alert correlation graphs and alert context logs, given that they are represented in separate vector spaces?
- **CH2**: How to let LLMs understand the semantic meanings of the alert correlation graph's nodes in relation to the security logs (Guo et al. 2023)?
- **CH3**: How to enable LLMs to reason effectively along the alert correlation graph's paths?

To address these challenges, we propose **GARNET**, a method of **G**oT-based **A**lert **R**eduction and **N**arrative **E**vent **T**racing. GARNET proposes the concept of GoT (Graph of Thoughts), using graph structures to guide LLMs in alert reasoning, while providing human-readable explanations. GARNET solves the three challenges mentioned above. For **CH1**, we design a graph-log multimodal alignment task based on contrastive learning, projecting the graph and the log embeddings into the same vector space. For **CH2**, we design a self-supervised graph-log instruction tuning task, guiding the LLM to align the semantic meaning of graph nodes and their corresponding security entities. For **CH3**, GARNET uses a novel GoT-based interaction reasoning approach, which breaks down graph-based alert reduction problems into a series of path-to-path logical reasoning. The LLM's responses will update the graph structure accordingly, ultimately generating the simplest alert paths with a human-readable explanation for attribution.

In summary, the contributions of this paper are as follows:

- We propose a novel GoT-based LLMs reasoning method for alert reduction. This method can automatically reduce false alerts, producing structured, concise, and human-readable attack path summaries.
- We align the graph and the log modalities on both the vector space and the semantic meanings. Our model is capable of understanding graph nodes and matching them with the corresponding security entities in logs.
- Evaluations on six attack scenarios show that our method outperforms the latest approaches in alert reduction and can provide more explainable attribution for security analysts.

Related Work

In this section, we provide an overview of the existing alert reduction methods, which can be categorized into three classes: methods based on knowledge, methods based on context, and methods based on graphs.

Knowledge-based Alert Reduction Methods. Knowledge-based methods introduce external knowledge through approaches such as attack pattern (Patil et al. 2025), knowledge graphs (Kaiser et al. 2023), threat intelligence (Qamar et al. 2017), and Retrieval-Augmented Generation (RAG) (Zhang et al. 2024). These methods provide sufficient evidence to assist analysts in making informed judgments, but they are highly dependent on the collection and mapping of external knowledge, failing to handle the unknown.

Context-based Alert Reduction Methods. Context-based methods completely eliminate the reliance on external knowledge. Liu et al. argue that context provides a more accurate representation of network alerts compared to knowledge-based methods. They propose Context2Vector (Liu et al. 2022), which utilizes language models to learn the representation of event contexts. DEEPCASE (Ede et al. 2022) uses the context of alerts to determine which events require further investigation. The method constructs and clusters attention vectors for each input sequence, enabling security operators to label clusters rather than individual events. Despite its effectiveness, DEEPCASE is vulnerable to evasion tactics and adaptive attacks (Tariq et al. 2025). DISTDET (Dong et al. 2023) is more universally applicable. It designs a ranking algorithm that takes into account the anomaly scores of alert events as well as their ancestors and descendants. Context-based analytical methods do not require prior knowledge and are capable of adapting to new, unknown threats. Despite these advantages, their deep learning models lack interpretability and fail to provide tangible reasoning for decision-making.

Graph-based Alert Reduction Methods. Graph-based methods address the limitations of context-based approaches by constructing alert correlation graphs (Mao et al. 2021; Hassan, Bates, and Marino 2020; Hossain et al. 2017; Milajerdi et al. 2019; Hassan et al. 2019). For a given alert, MORSE (Hossain, Sheikhi, and Sekar 2020) associates it with the subject or object that triggered the alert and performs a search in the graph. If other alerts are found, the weight of the target alert is increased, enabling alert prioritization. KRYSTAL (Kurniawan et al. 2022) identifies the root cause of an alert through backward-forward connections and then reconstructs the entire attack sequence. Similar to MORSE, they perform backward searches on the graph and add the scores of each preceding alert along the path to prioritize the alerts. However, these graphs are usually large and complex, which is hard to analyze. To solve this problems, NoDoze (Hassan et al. 2019) simplifies the dependency of the alert correlation graphs. Their graphs receive anomaly scores based on event frequency, where the total score ranks true alerts above false alerts, effectively reducing false positives. Although simplified, these graphs only contains anomaly scores and lack further explanations for analysts.

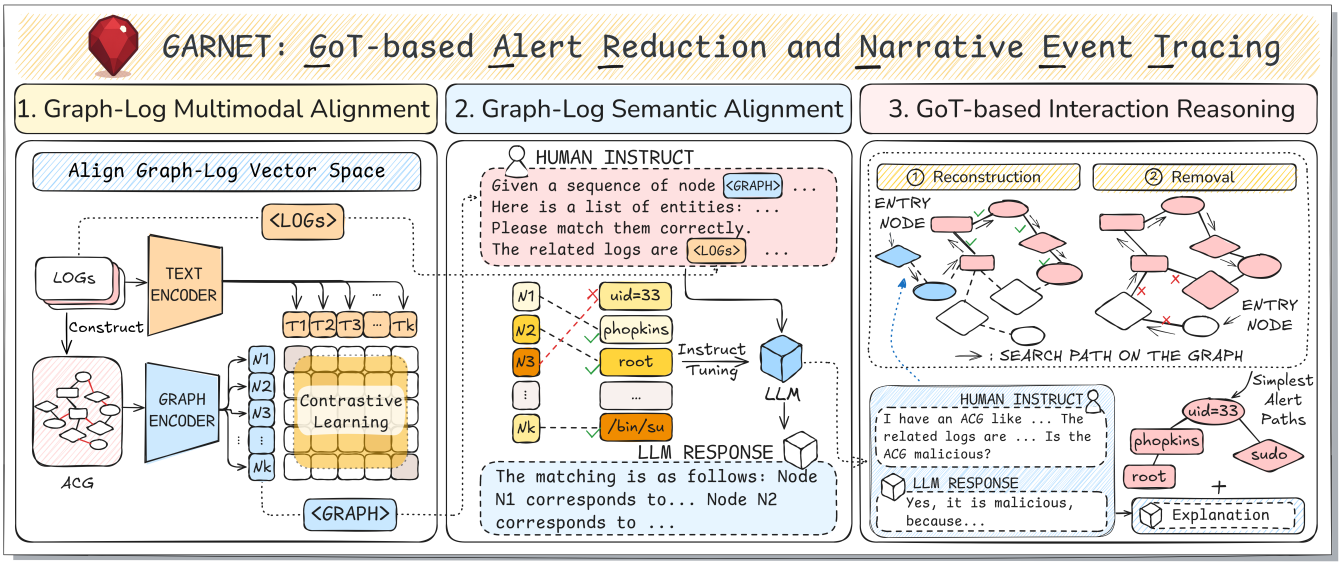


Figure 3: The overview of GARNET that uses LLMs for reasoning on alert correlation graphs. First, GARNET uses contrastive learning to project alert correlation graphs and logs into the same vector space. Then, GARNET generates instructions to guide the LLM match the semantic meanings of each graph nodes. Finally, a GoT-based interaction reasoning method is used to reduce the alerts and generate the explainable simplest attack path summaries that help security analysts trace attack events.

Methodology

In this section, we describe the details of GARNET, which includes three stages: graph-log multimodal alignment, graph-log semantic alignment and GoT-based interaction reasoning, as shown in Figure 3.

Graph-Log Multimodal Alignment

In this stage, we integrate the log context for scattered alerts and correlate them with Alert Correlation Graph (ACG). To enable the LLM to learn both graphs and logs modalities, we align the vector space of graphs' and logs' representations.

Log Correlation. The log context of alerts provides additional information for analysis. To obtain the context of given alerts, we correlate logs based on process groups around alerts. For a process p, from its start to its end, we consider this to be a process life cycle. For a process p_0, if there are n processes $\{p_1, p_2, ..., p_n\}$ that complete their life cycle during p_0's life cycle, these processes are grouped into an event E. We filter out log groups with the given alerts as their context.

Alert Correlation Construction. Based on the alerts' context, we construct alert correlation graphs to describe the relationships between security entities and their temporal interactions. For each alert context E, we build $ACG = (V, E, X)$, which is an undirected attributed graph. Here, V represents the set of all nodes, denoting entities in the network (including user ids, terminals, accounts, etc.). E is the set of all edges, representing heterogeneous correlation relationships between entities. X is the set of all node and edge attributes.

For a node $u_i \in V$, if there exists a log entry that correlates it with node $v_i \in V$, we connect these two nodes with an edge $e_i = (r_i, w_i, T_i) \in E$ of relationship r_i. When n log entries correlate u_i and v_i, the weight w_i of edge e_i is set to n, and its timestamp sequence is recorded as $T_i = \{t_{i1}, t_{i2}, ..., t_{in}\}$. For edge attributes, we combine the edge type r_i, edge weight w_i, and edge timestamp sequence T_i to form the edge feature $x_{ei} \in X$. For node attributes, we compute the weighted average of all connected edge attributes as the node feature $x_{vi} \in X$.

Graph-Log Vector Space Alignment. Since LLMs are designed for natural language, and the representation of graphs exists in a different vector space, LLMs cannot directly process graphs. Therefore, we align the representations of these two modalities and project them into the same vector space. We design a graph-log alignment task based on contrastive learning. The goal is to make the graph nodes and log embedding close to each other in the same vector space. Specifically, the embedding of node v on the graph and its corresponding log description t_v should be similar.

For graphs, we use a GCN (Kipf and Welling 2017) encoder to learn the embedding of each node:

$$h_v^{(l+1)} = \sigma\Big(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{|\mathcal{N}(v)|}\sqrt{|\mathcal{N}(u)|}} W^{(l)} h_u^{(l)}\Big)$$

where $h_v^{(l)}$ is the embedding of node v at the l-th layer; $\mathcal{N}(v)$ is the set of neighbors of v; $W^{(l)}$ is the trainable weight matrix of the l-th layer; σ is the activation function.

For logs, we use a Transformer-based text encoder (Radford et al. 2021) to generate embedding. For node v, if there are n correlated logs $L_v = (l_1, l_2, ..., l_n)$, we concatenate the log sequence L_v into text T_v in time order and use the text encoder to generate a fixed-dimensional representation z_{t_v}.

We use a contrast loss during training (Radford et al. 2021). For a set of nodes and log samples, we use the following formula to calculate the contrast loss:

$$\mathcal{L}_{CL} = \frac{1}{2}\left[\frac{1}{k}\sum_{i=1}^{k}\log\left(\frac{\exp\left(\mathbf{N}_i^\top \mathbf{T}_i/\tau\right)}{\sum_{j=1}^{k}\exp\left(\mathbf{N}_i^\top \mathbf{T}_j/\tau\right)}\right) + \frac{1}{k}\sum_{i=1}^{k}\log\left(\frac{\exp\left(\mathbf{T}_i^\top \mathbf{N}_i/\tau\right)}{\sum_{j=1}^{k}\exp\left(\mathbf{T}_i^\top \mathbf{N}_j/\tau\right)}\right)\right]$$

where $\mathbf{N_i} \in \mathbb{R}^{k\times D}$ represents the embedding matrix of graph node v_i, $T_i \in \mathbb{R}^{k\times D}$ represents the log sequence text L_v related to the security entity corresponding to v_i, k is the number of nodes and D is the dimension of the embedding. τ is the temperature parameter used to scale the similarity. This formula calculates the contrast loss between nodes and log text, and minimizes the similarity gap between them through cross-entropy loss.

Graph-Log Semantic Alignment

To enable the LLMs to understand the semantic meaning of the graph nodes as they relate to the security entities, we design a graph-log instruction tuning task, which leverages self-supervised signals derived from graph structures as instructions for fine-tuning. Specifically, we devise a structure-aware node classification task that guides the LLM to classify nodes using natural language-like logs.

Our instruction comprises four components: (1) graph data, (2) relevant logs, (3) human queries, and (4) LLM responses. For each instruction instance, we first randomly select a central node from the ACG and perform n-hop neighbor sampling to construct a subgraph. We use the aligned node embedding generated from the trained graph encoder as the graph data. Then, the logs related to the subgraph are added to the instruction. In human queries, we provide a list of security entities correlating to the subgraph's nodes. The LLM is then required to establish correct mappings between entities and nodes. This design necessitates a contextual understanding of logs for accurate node alignment.

During training, we directly put the node embedding into the instruction text after tokenization. Mathematically, for the node embedding $X_{\mathcal{N}} = f_{\mathbf{P}}(\hat{\mathbf{H}})$ and text embeddings $X_{\mathcal{I}} = tokenizer(instruction)$, we compute the probability of generating target output $X_{\mathcal{O}}$ for a sequence of length L as follows (Tang et al. 2024):

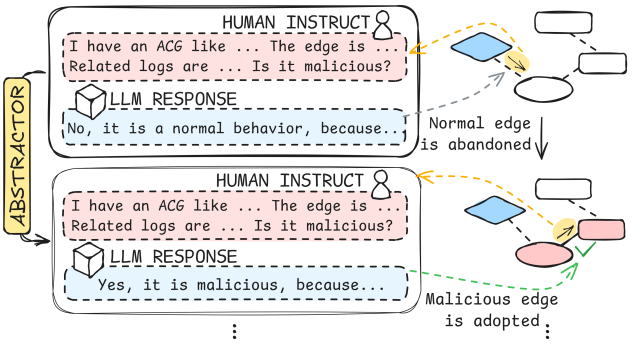
$$p(X_{\mathcal{O}}|X_{\mathcal{N}}, X_{\mathcal{I}}) = \prod_{i=1}^{L} p_\theta(x_i|X_{\mathcal{N}}, X_{\mathcal{I},<i}, X_{\mathcal{O},<i})$$

where θ is a learnable parameter of the LLM.

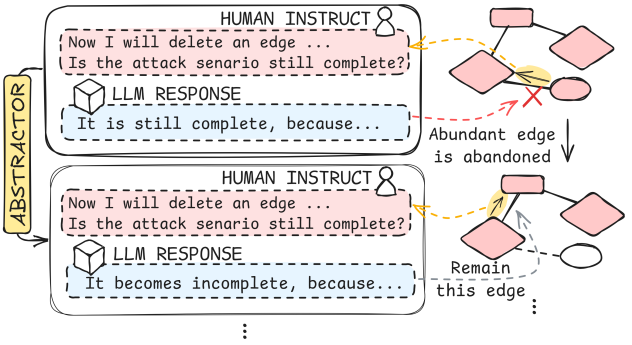
Finally, the LLM will calculate the cross-entropy loss between the given LLM responses and the output X_O to feed back into the training.

GoT-based Interaction Reasoning

Inspired by the CoT (Chain-of-Thought), which is a reasoning method that breaks down complex problems into a series of step-by-step logical reasoning to reach a conclusion,



(a) Attack paths reconstruction.



(b) Abundant paths removal.

Figure 4: The GoT-based interaction reasoning steps in GARNET, including attack paths construction and abundant paths removal. We utilize the structure of the graph to guide the reasoning of the LLM, and then use the response to determine the changes in the graph structure.

we propose the idea of GoT (Graph-of-Thought) to solve the alert reduction problem. Specifically, we guide the LLM to reason along the paths on the ACG, breaking down alert reduction problems into a series of path-to-path logical reasoning. The responses of LLM will also change the graph structure as interaction, as shown in Figure 4. Additionally, in each reasoning step, the LLM will give an explanation of its judgment, which helps the security analysts understand the alert path's details. Our GoT-based interaction reasoning approach consists of knowledge distillation, attack paths reconstruction and abundant paths removal.

Knowledge Distillation. To guide the LLM in more effectively analyzing, we introduce knowledge distillation technology. We use commercial LLM APIs with reasoning capabilities as the teacher model to evaluate and explain a subset of alert data, generating labels that serve as training data for our LLM. This allows our model to maintain performance close to that of the teacher model.

Attack Paths Reconstruction. Based on the ACG structure, we reconstruct the attack paths through the path-to-path reasoning process of LLM, which is described in Figure 4(a). We first randomly select an entry node and then perform a depth-first search (DFS) on the ACG, adding new paths to reconstruct a complete series of alert paths. Each time a new

edge is added, we input the changed alert paths and the logs associated with the new edge to the LLM, asking it to determine whether the current paths contain malicious behavior. When the LLM identifies malicious behavior on these paths, we select a new edge and ask the LLM whether adding this edge completes the attack scenario. If it does, the edge is added to the existing paths; if not, the edge is abandoned. Once the traversal is complete, the reconstruction alert paths will be able to describe the alerted attack completely.

In the GoT process, each step of LLM reasoning requires the previous conclusion. Therefore, we call LLM in the form of multi-round dialogues. In order to prevent the context length from being too long, we construct an abstractor that uses the LLM to summarize the content after each round of dialogue, providing high-quality contextual information for the next conversation.

Abundant Paths Removal. To simplify the attack paths, we also apply the GoT-based approach to remove abundant paths, which is described in Figure 4(b). We again randomly choose a starting node and use DFS to search through the attack paths. For each edge, we ask the LLM to determine whether the attack paths remain complete if that node is removed. If it remains complete, we consider the edge redundant and remove it. This process continues until the traversal is complete, resulting in the simplest attack paths. In this part, we also use an abstractor to extract the context of multiple rounds of dialogue as the input for the next round. The structural changes and interpretations of each dialogue round are recorded in the abstractor, providing automated analysis for security analysts.

Experiments

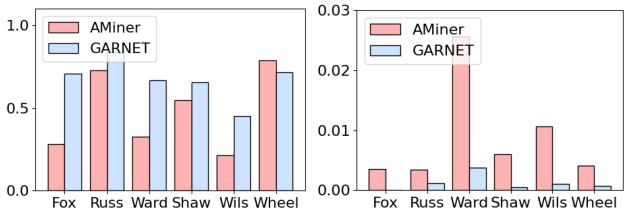
In this section, we outline the experimental setup, followed by the presentation of results and analysis in response to the following research questions.

RQ1: How effective is GARNET in alert reduction?
RQ2: Can GARNET provide explainable attack paths for the security analysts?
RQ3: What is the impact of varying parts in GARNET?

Experiment Settings

Dataset In our experiment, we utilize eight attack scenarios in the AIT Log Dataset V2.0 and the related AIT Alert Dataset (Landauer et al. 2021, 2022a) collected from the testbeds that are built at the Austrian Institute of Technology (AIT). We construct the alert correlation graphs from the audit and auth logs in the datasets which include privilege escalation and data leakage. The alert data is chosen from the AMiner detection method in the AIT Alert Dataset. Since we need some ground truth to generate instructions for the teacher model in the knowledge distillation stage, we select the logs and alerts in the scenarios 'harrison' and 'santos' as training sets. We evaluate GARNET and the baselines on the six remaining scenarios.

Baselines We compare with 6 of the latest alert reduction methods. First, we select two context-based methods: DEEPCASE (Ede et al. 2022) and aecid (Landauer et al. 2022b). Then, we choose one graph-based method (Hassan et al. 2019). Finally, we use one commercial LLM API with reasoning process (DeepSeek-r1, DS-r1 for short (DeepSeek-AI et al. 2025)) and two without reasoning (GPT-4.1 (OpenAI 2025) and DeepSeek-chat, DS-chat for short).



(a) F1-score comparison. (b) FPR comparison.

Figure 5: Performance comparison of alerts generated by GARNET and AMiner.

Implementation Details The experiments are conducted on a machine equipped with an Intel(R) Xeon(R) Silver 4210 CPU 2.20GHz and two NVIDIA Tesla P100 16 GB GPUs. We choose DeepSeek-R1-Distill-Qwen-14B (DeepSeek-AI 2025) as the basic model of GARNET and DeepSeek-r1 as the teacher model. We use the LoRA method (Hu et al. 2021) to fine-tune the basic model.

For metrics, we use F1-score, True Positive Rate (TPR) and False Positive Rate (FPR) to evaluate the effect of real alert classification. To highlight the performance of the alert reduction task, we introduce the Error Correction Rate (ECR) metric, which is calculated as follows:

$$ECR = \frac{FP_{origin} - FP_{reduction}}{FP_{origin}}$$

where FP_{origin} is the number of original false positive alerts and $FP_{reduction}$ is the false positive alerts after alert reduction. When $FP_{origin} \leq FP_{reduction}$, the ECR is 0.

Comparison Experiments (RQ1)

The comparison results of the alert reduction on 6 attack scenarios of datasets are shown in Table 1. It can be observed that our method, GARNET, achieves strong performance across all attack scenarios, outperforming all baseline methods in F1-score and surpassing most methods in both FPR and ECR.

It is worth noting that, compared to graph-based and LLM-based methods, context-based methods perform worse in alert reduction. This may be because these methods are unable to conduct an in-depth analysis of the root cause of the alert. As a result, they struggle to distinguish between genuine fundamental abnormalities and superficial symptoms, making them more prone to omission errors. For LLM methods, GPT-o1 and DS-chat are more likely to classify alerts as benign, while DS-r1 can identify more true alerts, indicating that adding the reasoning mechanism can improve the alert analysis performance. However, due to the lack of causal relationship, the effectiveness of these methods is still

Datasets	Metrics	Deepcase	Aecid	NoDoze	GPT-o1	DS-chat	DS-r1	**GARNET**
Fox	F1(↑)	0.1053	0.0159	0.4364	0.2222	0.2000	0.4545	**0.7083**
	FPR(↓)	0.0015	0.4271	0.0035	0.0204	0.2000	0.1786	**0.0000**
	ECR(↑)	0.5833	0.0000	0.0000	**0.9167**	0.8333	0.1667	0.8333
Russellmitchell	F1(↑)	0.1739	0.1436	0.6154	0.0000	0.2500	0.6250	**0.8333**
	FPR(↓)	0.0012	0.0507	0.0003	0.0233	**0.0000**	0.0930	0.0012
	ECR(↑)	0.6364	0.0000	0.9091	0.9091	**1.0000**	0.6364	0.6364
Wardbeck	F1(↑)	0.1455	0.0911	0.4706	0.1818	0.4000	0.2222	**0.6667**
	FPR(↓)	0.0066	0.1288	**0.0027**	0.0205	0.0068	0.7792	0.0037
	ECR(↑)	0.7403	0.0000	0.8961	0.9610	**0.9870**	0.1164	0.8571
Shaw	F1(↑)	0.1818	0.0818	0.5333	0.0000	0.0000	0.4348	**0.6531**
	FPR(↓)	0.0022	0.1388	**0.0005**	0.0127	0.0127	0.5600	**0.0005**
	ECR(↑)	0.6400	0.0000	0.9200	**0.9600**	**0.9600**	0.1392	0.5200
Wilson	F1(↑)	0.1200	0.0219	0.4528	0.0000	0.0000	**0.6316**	0.4490
	FPR(↓)	0.0034	0.6915	0.0018	0.0172	0.0172	0.0862	**0.0010**
	ECR(↑)	0.6829	0.0020	0.8293	**0.9756**	**0.9756**	0.8780	0.9024
Wheeler	F1(↑)	0.1935	0.0160	0.6316	0.0000	0.2222	0.7368	**0.7500**
	FPR(↓)	0.0010	0.5843	0.0003	0.0244	0.0244	0.0488	**0.0000**
	ECR(↑)	0.7500	0.0000	0.9167	0.9167	0.9167	0.8333	**1.0000**

Table 1: Compared with baselines in the alert reduction task. The best scores are in boldface, and the second-best scores are underlined.

unsatisfactory. Compared with them, NoDoze uses graphs to construct a causal relationships between alerts, resulting in better performance in decreasing false alerts, which proves the superiority of using graphs.

To show the accuracy of GARNET in alert reduction, we compare the alert hits generated by GARNET with the original alerts produced by AMiner, as shown in Figure 5. It can be observed that the alerts generated by GARNET have a higher F1 score, indicating that these alerts are closer to those generated by real attacks. Additionally, GARNET significantly reduces the false positives produced by the AMiner detector.

From the overall perspective of the comparison evaluation, GARNET combines the advantages of graph and log context, providing a large number of true alerts while reducing the original false alerts, verifying the effectiveness of our proposed framework.

Case Study (RQ2)

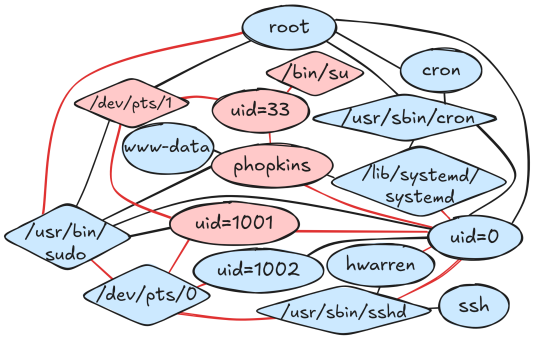
To demonstrate GARNET's ability in providing explainable attack paths, we select the attack scenario 'fox' on the AIT V2.0 dataset as a case study, which is shown in Figure 6.

In this scenario, the attacker initially leverages the *su* command to switch from the user *www-data* to the user *phopkins*, successfully initiating a session under the latter's credentials. Following the establishment of the session, the attacker utilizes *sudo* to escalate privileges, first attempting to list the contents of the *root*'s directory. The attacker executes the command *ls -laR /root/*, successfully enumerating the files within the root directory, thereby gaining root-level access. Then, the attacker executes the command *cat /etc/shadow*, which allowed him to read the contents of the */etc/shadow* file, a critical system file that contains the hashed passwords of all user accounts.

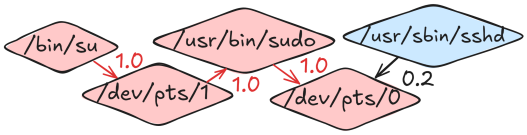
We build an alert correlation graph for the alerts generated by the Aminer detector on the AIT Alert Dataset, as shown in Figure 6(a). The red nodes represent entities involved in real attacks in the ground truth, the blue nodes represent entities related to non-real attacks, and the red edges represent the paths alerted by the AMiner detector. It can be seen that the original alerts identify many interactions between non-threatening entities as abnormalities, such as the process */usr/sbin/sshd*, the non-malicious user *hwarren*, and the normal *sudo* behavior of *root*. These alerts are redundant and incomprehensible, leaving security analysts confused.

Graph-based alert reduction methods can generate dependency graphs for real alerts. In Figure 6(b), we show the alert dependency graph generated by the NoDoze approach after alert aggregation and ranking. It can be seen that NoDoze has simplified the alert dependency, allowing security analysts to focus on specific processes. However, since Nodoze builds a process-level graph, it is difficult to capture permission changes during the privilege escalation process. In addition, the alert dependency graph provided by NoDoze only shows the anomaly score of the path, but does not include the specific cause. When security analysts see Figure 6(b), they still need to check the relevant logs and combine security knowledge to analyze the alerts.

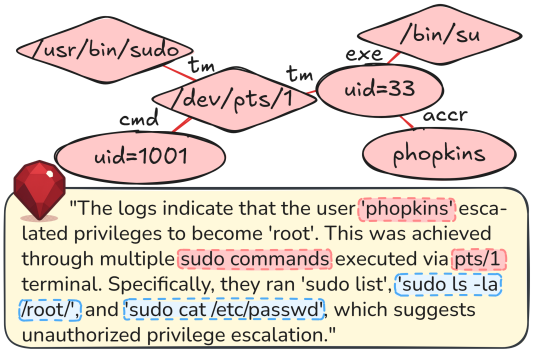
Our approach, GARNET, can both form concise and precise attack paths and provide understandable explanations for security analysts, which is shown in Figure 6(c). GARNET makes optimizations in the graph structure and can present the interaction types between security entities, such as the command execution operation of *uid=1001* on the



(a) The ACG constructed from the origin alerts generated by Aminer detector.



(b) NoDoze's simplified alert dependency graph.



(c) GARNET's simplest alert paths with explanation.

Figure 6: The case study on the fox attack scenario, where the attacker used the account "phopkins" to escalate privileges and successfully accessed the sensitive file /etc/shadow that stores all user encrypted passwords in the system.

process */dev/pts/1* and the association between *uid=33* and the account *phopkins*. In this way, with only the simplified attack paths, security analysts can initially discover the problem in the system: the *phopkins* account may have successfully performed multiple privilege escalation operations through a series of *sudo* and *su* commands. The explanation generated by LLM further restored the attack details: there were multiple privilege escalation instructions from the *pts/1* terminal in the security log, attempting to list files in the root directory and access sensitive files that stored system user password information.

This case study demonstrates that GARNET can generate structured, visually-based and explainable attack path summaries, helping security analysts quickly locate real alerts and provide analysis assistance.

Ablation	F1(↑)	ECR(↑)	TPR(↑)	FPR(↓)
(1) w/o multimodal align	0.4308	0.0000	0.4516	0.0058
(2) w/o semantic align	0.5333	0.8333	0.3871	0.0006
(3) w/o reasoning	0.1714	0.9167	0.3871	0.0003
(4) w/o reconstruction	0.3768	0.0000	0.4375	0.4194
(5) w/o removal	0.4854	0.0000	**0.8065**	0.0137
GARNET	**0.7083**	**1.0000**	0.5484	**0.0000**

Table 2: Ablation study on the fox scenario.

Ablation Study (RQ3)

For a clear understanding of GARNET's architecture design, we conduct a comprehensive ablation study on the 'fox' attack scenario. We design 5 sets of ablation experiments, which include: (1) not aligning multimodal of graphs and logs; (2) not aligning semantic meanings of graph and entities in logs; (3) replacing the reasoning basic model with LLM without reasoning; (4) removing the attack paths reconstruction stage; (5) removing the abundant paths removal stage. The ablation results are shown in Table 2.

It can be seen that removing the reasoning stage has the greatest impact on GARNET's performance, which is the kernel of our method. Removing alignment will affect the accuracy of GARNET to some extent, because it involves LLM's ability to understand graphs and logs. We observe that when removing the abundant path removal stage, the TPR will be higher, but the FPR will also increase at the same time. It may be because path reconstruction aims to complete suspicious causal chains, naturally prioritizing TPR, including edges that are less certain but "plausible." Redundant path removal aims for minimal sufficient explanation, naturally prioritizing precision (reducing FPR/increasing ECR). These steps "pull" on the objective function: the former is more "aggressive," the latter more "conservative." Thus, the observed metric changes are expected and consistent with the design.

Conclusion

In this paper, we propose GARNET, a GoT-based alert reduction and narrative event tracing method. GARNET reduces false positive alerts and provides clear attack scenarios by correlating alert context with graphs and reasoning them with the LLM. In order to guide the LLM to correctly perform alert reasoning on the graphs, we first align the modalities of graphs and security logs, projecting the graphs' and logs' representations into the same vector space. Then, we use a self-supervised graph-log instruction tuning task to let the LLM understand the semantic meaning of the nodes in the graphs as they relate to the security entities in logs. Finally, we design a novel GoT-based interaction reasoning approach to use the alert context graph to guide the LLM to perform reasoning and judgment. In the end, our method GARNET can generate the simplest attack scenario graph with a human-readable explanation, which helps security analysts tackle the alert fatigue. In future work, we will focus on utilizing LLMs agent to generate automated solutions based on these alerts.

Acknowledgments

This research is supported by National Key Research and Development Program of China (No.2023YFC2206402), and the Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDA0460100). This work is also supported by the Program of Key Laboratory of Network Assessment Technology, the Chinese Academy of Sciences, Program of Beijing Key Laboratory of Network Security and Protection Technology.

References

Alahmadi, B. A.; Axon, L.; and Martinovic, I. 2022. 99% False Positives: A Qualitative Study of {SOC} Analysts' Perspectives on Security Alarms. In *31st USENIX Security Symposium (USENIX Security 22)*, 2783–2800. ISBN 978-1-939133-31-1.

Chen, Y.; Xie, H.; Ma, M.; Kang, Y.; Gao, X.; Shi, L.; Cao, Y.; Gao, X.; Fan, H.; Wen, M.; Zeng, J.; Ghosh, S.; Zhang, X.; Zhang, C.; Lin, Q.; Rajmohan, S.; Zhang, D.; and Xu, T. 2024. Automatic Root Cause Analysis via Large Language Models for Cloud Incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*, 674–688. Athens Greece: ACM. ISBN 9798400704376.

DeepSeek-AI. 2025. DeepSeek-R1-Distill-Qwen-14B. https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B. Accessed: 2025-07-21.

DeepSeek-AI; Guo, D.; Yang, D.; and et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

Dong, F.; Wang, L.; Nie, X.; Shao, F.; Wang, H.; Li, D.; Luo, X.; and Xiao, X. 2023. {DISTDET}: A {Cost-Effective} Distributed Cyber Threat Detection System. In *32nd USENIX Security Symposium (USENIX Security 23)*, 6575–6592.

Ede, T. v.; Aghakhani, H.; Spahn, N.; Bortolameotti, R.; Cova, M.; Continella, A.; Steen, M. v.; Peter, A.; Kruegel, C.; and Vigna, G. 2022. DEEPCASE: Semi-Supervised Contextual Analysis of Security Events. In *2022 IEEE Symposium on Security and Privacy (SP)*, 522–539. ISSN: 2375-1207.

Guo, J.; Du, L.; Liu, H.; Zhou, M.; He, X.; and Han, S. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data ? An Empirical Evaluation and Benchmarking. arXiv:2305.15066.

Hassan, W. U.; Bates, A.; and Marino, D. 2020. Tactical Provenance Analysis for Endpoint Detection and Response Systems. In *2020 IEEE Symposium on Security and Privacy (SP)*, 1172–1189. San Francisco, CA, USA: IEEE. ISBN 978-1-72813-497-0.

Hassan, W. U.; Guo, S.; Li, D.; Chen, Z.; Jee, K.; Li, Z.; and Bates, A. 2019. NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage. In *Proceedings 2019 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society. ISBN 978-1-891562-55-6.

Hossain, M. N.; Milajerdi, S. M.; Wang, J.; Eshete, B.; Gjomemo, R.; Sekar, R.; Stoller, S.; and Venkatakrishnan, V. 2017. {SLEUTH}: Real-time attack scenario reconstruction from {COTS} audit data. In *26th USENIX Security Symposium (USENIX Security 17)*, 487–504. Vancouver, BC: USENIX Association. ISBN 978-1-931971-40-9.

Hossain, M. N.; Sheikhi, S.; and Sekar, R. 2020. Combating Dependence Explosion in Forensic Analysis Using Alternative Tag Propagation Semantics. In *2020 IEEE Symposium on Security and Privacy (SP)*, 1139–1155. San Francisco, CA, USA: IEEE. ISBN 978-1-72813-497-0.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685.

Jin, Z.; Chen, Y.; Leeb, F.; Gresele, L.; Kamal, O.; Lyu, Z.; Blin, K.; Gonzalez Adauto, F.; Kleiman-Weiner, M.; Sachan, M.; et al. 2023. Cladder: Assessing causal reasoning in language models. *Advances in Neural Information Processing Systems*, 36: 31038–31065.

Kaiser, F. K.; Dardik, U.; Elitzur, A.; Zilberman, P.; Daniel, N.; Wiens, M.; Schultmann, F.; Elovici, Y.; and Puzis, R. 2023. Attack Hypotheses Generation Based on Threat Intelligence Knowledge Graph. *IEEE Transactions on Dependable and Secure Computing*, 20(6): 4793–4809. Conference Name: IEEE Transactions on Dependable and Secure Computing.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907.

Kurniawan, K.; Ekelhart, A.; Kiesling, E.; Quirchmayr, G.; and Tjoa, A. M. 2022. KRYSTAL: Knowledge graph-based framework for tactical attack discovery in audit data. *Computers & Security*, 121: 102828.

Landauer, M.; Skopik, F.; Frank, M.; Hotwagner, W.; Wurzenberger, M.; and Rauber, A. 2022a. Maintainable Log Datasets for Evaluation of Intrusion Detection Systems. *IEEE Transactions on Dependable and Secure Computing*, 20(4): 3466–3482.

Landauer, M.; Skopik, F.; Wurzenberger, M.; Hotwagner, W.; and Rauber, A. 2021. Have it Your Way: Generating Customized Log Datasets With a Model-Driven Simulation Testbed. *IEEE Transactions on Reliability*, 70(1): 402–415.

Landauer, M.; Skopik, F.; Wurzenberger, M.; and Rauber, A. 2022b. Dealing with Security Alert Flooding: Using Machine Learning for Domain-independent Alert Aggregation. *ACM Transactions on Privacy and Security*, 25(3): 1–36.

Liu, J.; Zhang, R.; Liu, W.; Zhang, Y.; Gu, D.; Tong, M.; Wang, X.; Xue, J.; and Wang, H. 2022. Context2Vector: Accelerating security event triage via context representation learning. *Information and Software Technology*, 146: 106856.

Mao, B.; Liu, J.; Lai, Y.; and Sun, M. 2021. MIF: A multi-step attack scenario reconstruction and attack chains extraction method based on multi-information fusion. *Computer Networks*, 198: 108340. Publisher: Elsevier.

Milajerdi, S. M.; Gjomemo, R.; Eshete, B.; Sekar, R.; and Venkatakrishnan, V. 2019. HOLMES: Real-Time APT Detection through Correlation of Suspicious Information

Flows. In *2019 IEEE Symposium on Security and Privacy (SP)*, 1137–1152. San Francisco, CA, USA: IEEE. ISBN 978-1-5386-6660-9.

OpenAI. 2025. Introducing GPT-4.1 in the API. Accessed: 2025-07-24.

Patil, R.; Muneeswaran, S.; Sachidananda, V.; Hongyi, P.; and Gurusamy, M. 2025. PRIORITI: scoring and categorization-based threat prioritization. *The Journal of Supercomputing*, 81(1): 335.

Qamar, S.; Anwar, Z.; Rahman, M. A.; Al-Shaer, E.; and Chu, B.-T. 2017. Data-driven analytics for cyber-threat intelligence and information sharing. *Computers & Security*, 67: 35–58. Publisher: Elsevier Advanced Technology.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020.

Tang, J.; Yang, Y.; Wei, W.; Shi, L.; Su, L.; Cheng, S.; Yin, D.; and Huang, C. 2024. GraphGPT: Graph Instruction Tuning for Large Language Models. arXiv:2310.13023.

Tariq, S.; Baruwal Chhetri, M.; Nepal, S.; and Paris, C. 2025. Alert Fatigue in Security Operations Centres: Research Challenges and Opportunities. *ACM Comput. Surv.*, 57(9): 224:1–224:38.

Ullah, S.; Han, M.; Pujar, S.; Pearce, H.; Coskun, A.; and Stringhini, G. 2024. LLMs Cannot Reliably Identify and Reason About Security Vulnerabilities (Yet?): A Comprehensive Evaluation, Framework, and Benchmarks. In *2024 IEEE Symposium on Security and Privacy (SP)*, 862–880. San Francisco, CA, USA: IEEE. ISBN 9798350331301.

Zhang, W.; Zhang, Q.; Yu, E.; Ren, Y.; Meng, Y.; Qiu, M.; and Wang, J. 2024. LogRAG: Semi-Supervised Log-based Anomaly Detection with Retrieval-Augmented Generation. In *2024 IEEE International Conference on Web Services (ICWS)*, 1100–1102. Shenzhen, China: IEEE. ISBN 9798350368550.