MirGuard: Towards a Robust Provenance-based Intrusion Detection System Against Graph Manipulation Attacks

Anyuan Sang, Lu Zhou, Li Yang, Junbo Jia, Huipeng Yang, Pengbin Feng and Jianfeng Ma

***Abstract*—Learning-based Provenance-based Intrusion Detection Systems (PIDSes) have become essential tools for anomaly detection in host systems due to their ability to capture rich contextual and structural information, as well as their potential to detect unknown attacks. However, recent studies have shown that these systems are vulnerable to graph manipulation attacks, where attackers manipulate the graph structure to evade detection. While some previous approaches have discussed this type of attack, none have fully addressed it with a robust detection solution, limiting the practical applicability of PIDSes.**

To address this challenge, we propose MirGuard, a robust anomaly detection framework that combines logic-aware multi-view augmentation with contrastive representation learning. Rather than applying arbitrary structural perturbations, MirGuard introduces Logic-Aware Noise Injection (LNI) to generate semantically valid graph views, ensuring that all augmentations preserve the underlying causal semantics of the provenance data. These views are then used in a Logic-Preserving Contrastive Learning framework, which encourages the model to learn representations that are invariant to benign transformations but sensitive to adversarial inconsistencies. Comprehensive evaluations on multiple provenance datasets demonstrate that MirGuard significantly outperforms state-of-the-art detectors in robustness against various graph manipulation attacks without sacrificing detection performance and efficiency. Our work represents the first targeted study to enhance PIDS against such adversarial threats, providing a robust and effective solution to modern cybersecurity challenges.

***Index Terms*—Intrusion Detection System, Provenance Graph, Graph Manipulation Attack.**

I. Introduction

ADVANCED Persistent Threats (APTs) have become increasingly prevalent, posing significant risks to global cybersecurity [1]. These sophisticated and stealthy attacks target critical infrastructure, government systems, and private enterprises, often leading to severe data breaches, financial losses, and national security threats. The persistent nature of APTs allows attackers to maintain a foothold within compromised networks for extended periods, enabling them to exfiltrate sensitive information and disrupt operations, causing widespread harm to society and the economy.

Anyuan Sang, Junbo Jia, Huipeng Yang, Lu Zhou, and Li Yang are with the School of Computer Science and Technology, Xidian University, Xi'an, China.

Pengbin Feng and Jianfeng Ma are with the School of Cyber Engineering, Xidian University, Xi'an, China.

This work was supported in part by the National Key R&D Program of China (2023YFB3106900), and the National Natural Science Foundation of China (grants No.62302362, 62402364, 62472337), in part by the Fundamental Research Funds for the Central Universities, and in part by the Innovation Fund of Xidian University under Grant YJSJ25012.

*Li Yang is the corresponding author. E-mail: yangli@xidian.edu.cn.

Provenance graphs, which capture the causal relationships between system entities and events, have become a valuable foundation for behavior-based intrusion detection. These graphs provide rich contextual information that enables detailed analysis of system activity and potential attack chains [2], [3]. Detection methods based on provenance graphs can be broadly categorized into two approaches: knowledge-based [4]–[7] and learning-based techniques [8]–[14]. Knowledge-based methods rely on predefined rules or metrics to perform anomaly detection within the graph. However, their dependence on prior knowledge and inability to capture advanced, deep features have driven researchers toward learning-based approaches.

Learning-based detection methods leverage various levels of graph embedding techniques in upstream tasks (graph learning), such as node embedding [10], [12], edge embedding [9], [11], and subgraph embedding [8], [15], to derive expressive representations of the provenance graph. These techniques effectively capture both contextual and structural information. Subsequently, downstream detection algorithms, including outlier detection and vector similarity analysis, are applied to identify anomalies and detect potential attacks.

Although existing methods have demonstrated effective detection performance, recent graph manipulation attack strategies pose significant challenges to these detectors [16]–[18]. graph manipulation attacks involve attackers forging interaction information of malicious processes, such as adding sufficient edges connecting to benign nodes, to shift their representation in the embedding space and evade detection. This vulnerability arises from an inherent limitation of machine learning models, where small perturbations can lead to high-confidence misclassification [19], [20]. To the best of our knowledge, only a few studies [10], [12], [14], [21] have briefly explored the impact of such attacks on provenance graph-based detection methods, and even fewer have proposed targeted mitigation strategies. Generic robustness enhancement strategies, such as adversarial training [22], [23], face practical challenges in the PIDS domain due to the scarcity of malicious samples and may be ineffective against potential unseen attacks. Therefore, current defense mechanisms are inadequate for countering mimicry attacks, highlighting an urgent need for a novel robustness enhancement approach that improves the intrinsic robustness of detection models.

In this paper, we propose **MirGuard**, a novel anomaly detection method based on provenance graphs, designed to enhance robustness against graph manipulation attacks while maintaining high detection accuracy. Provenance graphs are vulnerable to such attacks, where adversaries mimic benign behaviors to conceal malicious activities and evade detection

[16]–[18]. In this work, we analyze the typical workflow of PIDS and identify two primary types of graph manipulation attacks (as detailed in the threat model section): graph poisoning attacks during the training phase and graph pollution attacks during the detection phase. To counter such attacks, MirGuard leverages a multi-view learning strategy that combines structured graph augmentation with contrastive learning. The key idea is to force the model to learn representations that are invariant to local perturbations and sensitive to global malicious patterns.

Specifically, MirGuard applies a logic-aware graph augmentation strategy, which ensures that all perturbations conform to structural semantics defined by the provenance context (e.g., disallowing file-to-network or network-to-network edges). This results in more realistic adversarial simulations compared to random augmentations. These augmentations disrupt attacker-crafted patterns and encourage the model to focus on more stable, graph-level semantics.

Based on these augmented views, MirGuard employs a contrastive learning framework that encourages semantic consistency across views while distinguishing unrelated behaviors. Unlike conventional approaches such as GraphCL [24] or MVGRL [25], our method emphasizes semantic consistency rooted in domain-specific logic, rather than superficial structural similarity alone. By learning representations invariant to benign-appearing manipulations but sensitive to semantic inconsistencies, MirGuard achieves strong robustness against both poisoning and evasion attacks. Our evaluations demonstrate that this design leads to improved generalization and more reliable anomaly detection in complex and adversarial environments.

After obtaining robust graph representations, MirGuard employs an unsupervised anomaly detection mechanism based on KMeans clustering. In the training phase, KMeans is used to partition the embedding space into k clusters. The centroids of these clusters, along with the average intra-cluster distance across training samples, are retained as references. During inference, each test sample is evaluated by computing its Euclidean distances to all cluster centroids. The minimum distance is taken as the initial anomaly score, which is then normalized by the global average distance. If the normalized score exceeds a predefined threshold, the sample is flagged as anomalous. This centroid-based detection strategy enables MirGuard to perform efficient and scalable anomaly detection, significantly reducing inference overhead while preserving high detection performance.

To comprehensively evaluate the efficiency of MirGuard, we utilized widely adopted provenance datasets, including DARPA TC THEIA, CADETS, TRACE [26], the Streamspot dataset [27], and the Unicorn Wget dataset [28]. We also employed several state-of-the-art graph learning-based anomaly detectors, such as Threatrace [10], MAGIC [12], and FLASH [14], as baselines. To thoroughly assess MirGuard's robustness against graph manipulation attacks, we implemented five types of such attacks during both the detection and training phases, based on prior evasion studies [16]–[18]. In our experiments, we first evaluated MirGuard's resistance to different attack types and compared its robustness with that of current state-of-the-art detection schemes. We then discussed whether MirGuard sacrifices detection performance to achieve robustness. Next, we demonstrated the rationale and necessity of MirGuard's module design through ablation experiments. Finally, we evaluated the overhead of MirGuard and discussed the impact of different parameter settings on its performance.

Our contributions are summarized as follows:

- To the best of our knowledge, we are the first to specifically enhance the robustness of PIDS models against graph manipulation attacks.
- We propose a novel graph learning-based PIDS, MirGuard, which is designed with a unique multi-view augmentation strategy and employs a contrastive learning mechanism to train the model. This enables MirGuard to achieve strong robustness against graph manipulation attacks while maintaining detection performance comparable to state-of-the-art detectors.
- We implemented five types of attacks across both the training and detection phases and conducted comprehensive evaluations of MirGuard's robustness and detection performance on multiple datasets. Experimental results demonstrate that, compared to baseline systems, MirGuard exhibits exceptional robustness against graph manipulation attacks without compromising detection performance or incurring additional detection overhead (achieving an average F1-score of over 96% with less than 10% AUC drop under graph manipulation attacks).

II. Background

A. Graph Manipulation Attacks

Graph manipulation attacks [16]–[18] pose a significant challenge to graph-based systems by strategically altering graph structures to evade detection or degrade model performance. These attacks often target critical graph elements, such as nodes, edges, or features, to disguise malicious behavior as benign or disrupt the learning process of graph-based models. For instance, attackers may inject fake nodes or edges to obscure critical relationships or modify existing features to mimic benign entities, making it harder to detect anomalies. In the context of provenance graphs, these attacks exploit the graph's structural and semantic complexity, where malicious subgraphs are embedded within larger benign structures, allowing adversaries to manipulate local patterns while preserving global consistency. This obfuscation enables attackers to bypass anomaly detection methods that heavily rely on local or static patterns. Addressing such attacks requires robust graph-based methods that can capture invariant global features and distinguish subtle manipulations, ensuring resilience against adversarial perturbations.

B. Provenance-based IDS

Since the provenance graph can express the relationship between system operating entities in time, existing research has used this feature to build an IDS based on the provenance graph. Including detection schemes based on knowledge labels [4]–[7], these schemes construct a series of matching rules

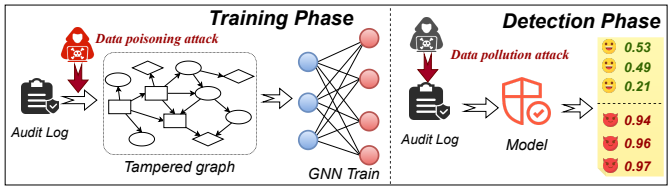


Fig. 1: The classic detection processes of PIDS identified two types of graph manipulation attacks: data poisoning attacks during the training phase and data pollution attacks during the detection phase.

based on expert knowledge to match in the origin graph to detect anomalies. Based on the statistics IDS scheme [29]–[31], they use the structural feature information of the graph, including: abnormality, discrepancy, time correlation and other features to analyze in the graph to detect anomalies. Recently, more learning-based IDS solutions have been proposed [8], [11], [12], [15], [21], [32]. These solutions use models such as graph representation learning and sequence learning to extract high-dimensional features from graphs to perform anomaly detection in downstream tasks.

III. Motivation Example

This scenario illustrates an APT attack conducted through a browser extension in the DARPA TC E3 dataset. Figure 2 provides a simplified visualization of this attack. The attack was initiated when the victim visited a malicious website that exploited a vulnerability in the *pass_mgr* extension of the Firefox browser. The attacker leveraged this vulnerability to download a program named *gtcache*. The *gtcache* program connected with the attacker and executed data theft operations. Additionally, it installed another program, *ztmp*, to gather system configuration details and perform port scans on the target network for internal reconnaissance. Notably, in this attack scenario, we introduce a manipulation strategy where the attacker alters the graph structure by inserting benign subgraphs into the attack subgraph to evade detection.

This attack poses a significant challenge to existing learning-based detection methods, especially those relying on graph embedding techniques such as GraphSAGE, GNNs, and Graph2Vec [8], [10], [11]. These methods are susceptible to graph manipulation attacks, which can modify the neighborhood structure of malicious nodes, causing them to resemble benign nodes more closely. As a result, the embeddings learned during training may increasingly resemble benign behavior, significantly impairing the model's ability to differentiate between malicious and benign activities. This phenomenon, known as evasion attacks, occurs when the attacker manipulates the graph such that malicious nodes are embedded in regions of the graph space typically occupied by benign nodes. Consequently, the detection model, trained on these altered embeddings, becomes more prone to evasion, resulting in a decline in its overall robustness. These challenges underscore the need for detection systems that can not only learn effective representations of graph data but also remain robust against adversarial manipulations designed to conceal malicious behaviors.

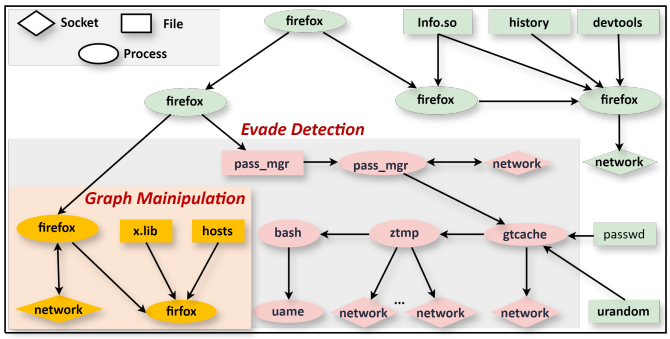


Fig. 2: In the provenance graph of the TC E3 browser extension attack, we considered a graph manipulation attack proposed by [16], where attackers could manipulate the graph structure by inserting benign subgraphs into the attack subgraph to evade detection. The green nodes represent benign nodes engaged in normal activities, red nodes represent attack nodes, and yellow nodes indicate attack nodes added by the attacker.

IV. Threat Model & Assumptions

Our experimental environment relies on a Trusted Computing Base (TCB) consisting of the operating system, auditing framework, and provenance analysis tools. We assume that all components within the TCB function correctly throughout the entire process from installation to execution. This assumption is standard in existing provenance-based detectors. Hardware trojans and side-channel attacks that cannot be captured through audit mechanisms are not considered in this paper. In addition, we assume that the integrity of the output audit data is guaranteed by existing secure provenance and integrity audit systems [33]–[37].

In our robustness evaluation experiments, we analyzed the typical processing pipeline of provenance-based systems, as shown in Figure 1, and identified two types of graph manipulation attacks: data poisoning attacks during the model training phase and data pollution attacks during the detection phase. Previous studies [16]–[18] assumed that adversaries can manipulate the structure of the provenance graph to launch attacks against the detector. This falls into the category of data pollution attacks. Based on this, we extend the attack model by introducing a stronger assumption in which adversaries can also inject crafted graph perturbations into the audit logs during the training phase. This results in data poisoning attacks that affect the model's training outcomes.

V. Design

As shown in Fig.3 MirGuard comprises three main components: (1) Graph Builder, (2) Graph Representation, and (3) Anomaly Detection.

In the graph builder module, MirGuard processes system audit logs to construct the provenance graph, where nodes represent system entities and edges denote interactions between them. Edge compression techniques are employed to merge redundant nodes and edges, optimizing the graph structure and reducing computational complexity. Additionally, batch-based

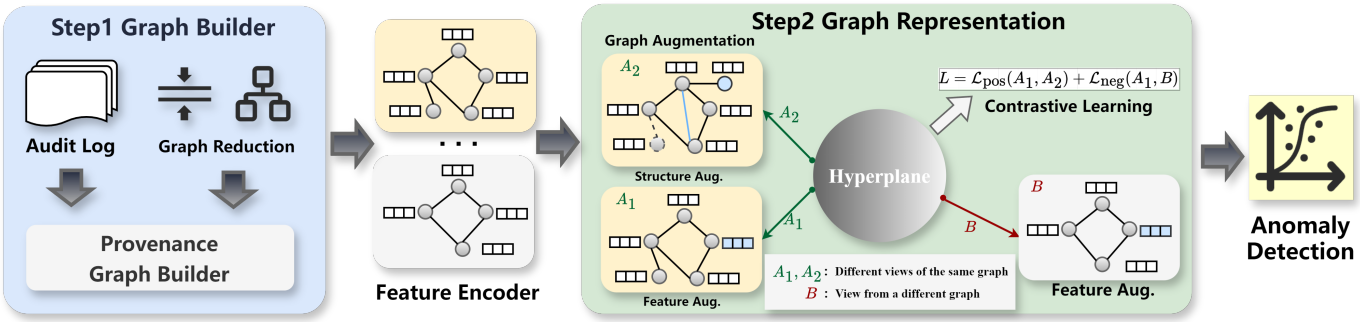


Fig. 3: Overview of MirGuard's architecture.

provenance graph construction is implemented to handle large-scale data by splitting the graph into smaller batches. Node and edge types are extracted for subsequent feature encoding in the representation module.

The core of MirGuard lies in the graph representation module, which includes feature encoding, graph augmentation, and contrastive learning. First, node and edge features are encoded using one-hot encoding to standardize the input. Then, GNN, such as a Graph Attention Network (GAT), is employed to extract higher-order structural and semantic features, capturing both local and global dependencies. Graph augmentation introduces controlled perturbations to simulate adversarial scenarios, enhancing the model's ability to learn invariant representations. Finally, a contrastive learning framework aligns embeddings of augmented views while maintaining separation between distinct graphs, ensuring robustness against adversarial manipulations.

The anomaly detection module employs a KMeans-based detection method to identify anomalous nodes in the graph. Although various classifiers were considered, KMeans-based detection demonstrated superior performance in our evaluations, as detailed in Section VI-D.

A. Graph Builder

Our system accepts streaming system audit logs and constructs the provenance graph, similar to previous research [38], [39]. It consists of three main components. First, MirGuard streams and extracts audit logs in batches from existing operating systems, such as Windows ETW logs or Linux audit logs. These logs contain information about interactions between system entities, including files, processes, and networks. Next, MirGuard extracts and processes this log information. Specifically, for each audit log within a batch, it extracts the fundamental components representing the nodes and edges of the provenance graph: the quadruple *(src, dst, timestamp, edge type)*, where *src* denotes the process node, *dst* represents the file or network node, *timestamp* indicates the time when the event occurred, and *edge type* specifies the type of edge. Finally, to accelerate model training and reduce computational complexity, we adopt multi-class graph denoising techniques from prior studies, removing only redundant nodes and those irrelevant to attack detection. MirGuard utilizes the CPR (Causal Persistent Reduction) method [40] for edge processing, retaining only one instance of edges that appear multiple times between two nodes within a short time window. Additionally, during graph construction, orphaned nodes and faulty nodes (potentially generated by logging errors) that are unrelated to the attack investigation are removed.

B. Graph Representation

Graph representation in MirGuard involves a systematic pipeline to transform the raw provenance graph into robust embeddings suitable for anomaly detection. This process begins with feature encoding, where node and edge attributes are represented using one-hot encoding and refined through GNNs to capture both local and global structural dependencies. Following this, graph augmentation introduces controlled perturbations to simulate adversarial scenarios, enhancing the model's ability to learn invariant representations. Finally, a contrastive learning framework aligns embeddings of augmented views while maintaining separation between distinct graphs, ensuring robustness against adversarial manipulations and capturing meaningful graph semantics. Together, these steps enable MirGuard to construct high-quality graph embeddings that are both expressive and resilient, forming the foundation for reliable detection in complex environments.

1) Feature Encoding: MirGuard begins by encoding the raw provenance graph's node and edge attributes using one-hot encoding. Each node and edge is represented by its type, and one-hot encoding is applied to generate a categorical feature vector. This process transforms discrete attributes, such as node types (e.g., processes, files) and edge types (e.g., read, write), into binary vectors that preserve their distinct semantics.

Once the one-hot encoding is complete, MirGuard employs a Graph Neural Network (GNN), such as a Graph Attention Network (GAT), to extract higher-order structural and semantic features. The GNN processes the graph by aggregating information from neighboring nodes and edges, capturing both local dependencies and global contextual patterns. For a node v, its feature representation $h_v^{(l+1)}$ at layer $l+1$ is computed as:

$$h_v^{(l+1)} = \sigma\left(W^{(l)} \cdot \text{AGG}\left(\{h_u^{(l)} \mid u \in \mathcal{N}(v)\}\right) + b^{(l)}\right),$$

where $h_v^{(l)}$ is the feature vector of node v at layer l, $\mathcal{N}(v)$ represents the neighbors of v, $\text{AGG}(\cdot)$ is an aggregation function (e.g., sum or mean), $W^{(l)}$ and $b^{(l)}$ are learnable parameters, and σ is an activation function (e.g., ReLU). This

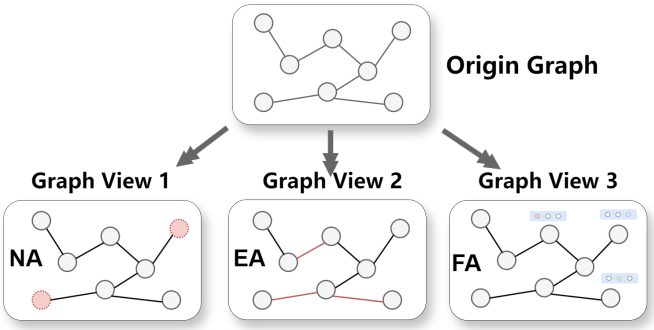


Fig. 4: Examples of logic-aware graph augmentation. Edge and node operations are constrained to preserve provenance semantics.

encoding process generates a dense feature vector for each node and edge, capturing both their individual properties and relational information.

The output of the GNN serves as the input for subsequent graph augmentation and contrastive learning steps.

2) Logic-Aware Graph Augmentation: MirGuard employs logic-aware graph augmentation strategies to simulate potential adversarial attacks while maintaining the semantic plausibility of the provenance graph. These augmentations include edge augmentation (EA), node augmentation (NA), and feature augmentation (FA), each applied to generate perturbed views of the original graph. The intensity of each augmentation operation is controlled by the hyperparameter γ, which specifies the proportion of nodes or edges to be modified.

To ensure the realism and logical validity of augmented graphs, we implement a strategy called **Logic-Aware Noise Injection (LNI)** as shown in Table I. This strategy enforces rationality constraints during augmentation to prevent the generation of semantically invalid structures. For instance, in edge augmentation, we prohibit the addition of edges that directly connect two network nodes or connect a file node to a network node—such configurations violate causal semantics in provenance graphs.

TABLE I: Logic-Aware Edge Augmentation Rules in Provenance Graphs

Source	Destination	Allowed	Edge
Process	File	✓	read/write
Process	Network	✓	connect/send/recv
File	Process	✓	exec/load
Process	Process	✓	fork/clone
Network	Process	×	violates causality
File	Network	×	no direct communication
Network	File	×	no direct communication
Network	Network	×	meaningless edge

a) Edge Augmentation (EA): Edge augmentation modifies the graph structure by adding or removing edges under logic constraints. The edge set E' is modified as follows:

$$E' = E \cup \{(u, v)\},$$

This operation adds an edge between nodes u and v only if it does not already exist *and* the connection satisfies domain-specific logic rules.

$$E' = E \setminus \{(u, v)\},$$

Alternatively, an existing edge can be removed. The perturbation intensity is controlled by γ; for example, 20% of edges are randomly selected for addition or removal, subject to logical validity.

b) Node Augmentation (NA): Node augmentation involves adding or removing nodes along with their associated edges, while ensuring logical consistency in their insertion or removal context:

$$V' = V \cup \{v'\},$$

This means a new node v' is added to the graph and linked to others only through permissible edge types.

$$V' = V \setminus \{v\}, \quad E' = E \setminus \{(u, v)\},$$

An existing node v may be removed along with all its connected edges. Node augmentation is also governed by γ; for example, 20% of the nodes are selected for addition or deletion.

c) Feature Augmentation (FA): Feature augmentation modifies node attributes while preserving semantic alignment. The feature vector of node v is replaced with that of another node w of the same type:

$$X'_v = X_w, \quad w \sim \{u \in V \mid \text{type}(u) = \text{type}(v)\},$$

This simulates adversarial feature manipulation without disrupting node-type semantics.

Collectively, these logic-aware augmentations enhance the model's ability to learn invariant patterns and detect adversarial perturbations that preserve surface semantics but violate causal consistency. They prepare the model for challenging attack scenarios while retaining the integrity of the graph's structural and semantic foundations.

3) Logic-Preserving Contrastive Learning: To further improve robustness, MirGuard introduces a contrastive learning framework tailored to provenance graphs, with an emphasis on preserving the underlying causal semantics. Unlike general-purpose contrastive learning frameworks such as GraphCL [24] or MVGRL [25], which rely on random augmentations and structure-based similarity, our approach incorporates domain-aware augmentations and logic consistency.

Given two augmented views G_i and G_j of the same original graph G, the encoder generates graph-level embeddings z_i and z_j, which are passed through a two-layer projection head:

$$p_v = \text{ReLU}(W_p^{(1)} z_v + b_p^{(1)}), \quad \hat{p}_v = W_p^{(2)} p_v + b_p^{(2)}.$$

We then compute the contrastive loss:

$$\mathcal{L} = -\log \frac{\exp(\text{sim}(\hat{p}_i, \hat{p}_j)/\tau)}{\sum_{k=1}^{N} \exp(\text{sim}(\hat{p}_i, \hat{p}_k)/\tau)},$$

where $\text{sim}(\cdot,\cdot)$ denotes cosine similarity, and τ is a temperature parameter. Positive pairs $(\hat{p}_i, \hat{p}_j)$ originate from different views of the same graph that preserve logical structure, while negatives $\hat{p}_k$ come from unrelated graphs.

This design ensures that the learned embeddings reflect consistent high-level behaviors rather than superficial structural features. It enables the model to resist manipulation that mimics graph topology while violating semantic logic, which is especially critical for provenance-based anomaly detection.

C. Anomaly Detection

To determine the most effective self-supervised anomaly detection mechanism, we evaluated several candidate classifiers, including Local Outlier Factor (LOF) [41], One-Class SVM [42], KMeans [43], and Isolation Forest [44]. As detailed in Section VI-D, KMeans demonstrated superior performance in terms of detection accuracy. Consequently, MirGuard employs a KMeans-based anomaly detector, which includes training and detection phases.

In our method, we apply K-means to partition the embedding space into k clusters and retain all cluster centroids for subsequent anomaly detection.

During the detection phase, each new embedding vector is evaluated by computing its distance to the nearest cluster centroid. The anomaly score S_i for a data point x_i is defined as the Euclidean distance to the closest centroid among the k clusters:

$$S_i = \min_{j=1,\ldots,k} \|x_i - c_j\|,$$

where c_j denotes the j-th cluster centroid obtained from training.

To ensure comparability across datasets and feature scales, we normalize the raw anomaly score using the mean nearest-centroid distance computed on the training set, denoted as D_{mean}. This is defined as:

$$D_{\text{mean}} = \frac{1}{N}\sum_{i=1}^{N} \min_{j=1,\ldots,k} \|x_i - c_j\|,$$

where N is the number of training samples. The normalized anomaly score $\tilde{S}_i$ is given by:

$$\tilde{S}_i = \frac{S_i}{D_{\text{mean}}}$$

A data point is considered anomalous if its normalized score exceeds a predefined threshold θ:

$$\text{Anomaly}(x_i) = \begin{cases} 1 & \text{if } \tilde{S}_i > \theta, \\ 0 & \text{otherwise.} \end{cases}$$

Given the large-scale nature of provenance data, this centroid-based evaluation strategy significantly reduces the inference overhead compared to pairwise distance-based approaches such as KNN [12], while maintaining effective anomaly detection performance, especially for large-scale provenance data.

VI. Evluation

In this section, we evaluate the performance of MirGuard by addressing the following research questions (RQs):

- **RQ1:** How is MirGuard's detection efficiency compared to baseline methods?
- **RQ2:** Does MirGuard successfully improve robustness against graph manipulation attacks compared to its baselines?
- **RQ3:** To what extent do the structured augmentations and multi-view contrastive learning contribute to MirGuard's ability to counteract graph manipulation attacks and detect malicious behaviors?
- **RQ4:** Does MirGuard introduce significant computational overhead compared to existing PIDSes?

A. Experiments Setup

In data processing, we adopted the log transformers in MAGIC [12] for processing streaming audit logs, including StreamSpot [45], Camflow [34] and DARPA TC Dataset [26]. Networkx is used to construct the provenance graph. The graph indicates that the learning module is implemented by Pytorch [46] and DGL [47].

Parameter settings. For the setup of MirGuard, the learning rate *lr* is set to 0.001. We use a 2-layer GAT encoder, and in data augmentation, the augmentation ratio is set to 0.5. The training batch size is 50 with *d* set to 64 on the DARPA TC dataset.

Datasets. We evaluated the performance of MirGuard under three open-source datasets: DARPA Engagement TC E3 [26], Streamspot and, Unicorn Wget. All three datasets are inconsistent in the scenarios they target and the granularity of their detections, and thus we believe they are able to provide insights into the performance of the system. The detail of the dataset description are as follows:

TABLE II: Dataset Statistics for Streamspot and Unicorn Wget

Dataset	Graph Pieces	Entities	Interactions	Size (GB)
Streamspot	100	8,292	113,229	2.8
		8,636	112,958	
		8,989	294,903	
		8,830	310,814	
		6,826	37,382	
		8,890	28,423	
Unicorn Wget	125	265,424	975,226	76
	25	257,156	949,887	

TABLE III: Dataset Statistics for DARPA E3

Dataset	Benign Nodes	Abnormal Nodes	Edges	Size (GB)
E3 Trace	3,220,596	68,082	4,080,457	67
E3 Cadets	1,614,189	12,846	3,303,264	
E3 Theia	3,505,326	25,362	10,929,710	

- **DARPA TC dataset.** The DARPA TC dataset is a benchmark dataset provided by DARPA for evaluating cybersecurity and intrusion detection systems. It was collected from networks during adversarial engagements. The red team conducted APT attacks using various vulnerabilities to exfiltrate information. Our evaluation

TABLE IV: Details of Graph Manipulation Attacks

Phase	Attack Type	Target	Rate (y)
Detection	GSPA	Node	y
	GFPA	Edge	y
	CGPA	Node & Edge	$0.5y + 0.5y$
Training	SPA	Node & Edge	$0.5y + 0.5y$
	FPA	Node	y

includes the TRACE, CADETS, and THEIA subdatasets, which contain millions of entities and interaction records. We used the ground truth information provided by the Threatrace [10] to perform entity-level detection and conduct attack investigations.

- **Unicorn Wget dataset.** The Wget dataset was designed by the authors of Unicorn [28] to simulate attack scenarios. It uses the Camflow [34] system to collect 150 batches of audit logs, with 125 batches containing no attack processes and 25 batches containing supply chain attacks. These attacks are carefully crafted to mimic benign entity interactions, making this dataset challenging to identify due to its large data volume and stealthy attacks. We will perform graph-level detection on this dataset as in previous approaches.
- **StreamSpot dataset.** The StreamSpot dataset is a publicly available dataset provided by the authors of StreamSpot [45], containing 600 information flow graphs. These graphs come from five benign scenarios and one attack scenario. Each scenario runs 100 times, generating 100 graphs using the Linux SystemTap Logging System. The five benign scenarios simulate normal user behavior, while the attack scenario simulates a drive-by download attack. We performed graph-level anomaly detection on the StreamSpot dataset, similar to previous studies [10], [28], as it only provides graph-level ground truth.

Baselines. To comprehensively evaluate the detection performance of MirGuard, we compare it with state-of-the-art (SOTA) and open-source graph-based methods in the PIDS domain, including Threatrace [10], MAGIC [12], and FLASH [14]. It is worth noting that several other approaches were not included in our comparison for the following reasons:

First, since MirGuard is a graph-based anomaly detection method, we excluded signature-based methods [5], [7], [48], priority-based approaches [29]–[31], and graph sketch-based techniques [28]. Additionally, some recent works [13], [49] adopt finer-grained root node labeling strategies, which differ significantly from our threat model and would hinder a fair comparison. As such, these methods were also excluded.

Second, as noted by the authors of [13], many learning-based detectors in the PIDS domain, such as ProvDetector [9], ShadeWatcher [11], RCAID [50], and ProGrapher [8], are not fully open-source. Reproducing these methods solely based on their published descriptions may introduce experimental bias; therefore, we chose not to include them in our evaluation.

Graph manipulation attack. We provide a detailed description of the experimental setup used to evaluate the robustness of MirGuard against graph manipulation attacks, which aim to evade detection by modifying either the graph structure or node features. We broadly categorize these attacks into two types: data pollution attacks that occur during the detection phase, and data poisoning attacks that take place during the model training phase. Following prior work [12]–[14], [21], we adopt five different attack scenarios for comprehensive evaluation.

(1) Data Pollution Attacks. Data pollution attacks aim to manipulate graph structures during the detection phase to hide malicious behaviors. For this category, we implemented three types of graph manipulation attacks:

- **Graph Feature Pollution Attack (GFPA).** Alters the features of malicious nodes to mimic those of benign nodes, thereby hiding malicious behavior and evading detection.
- **Graph Structure Pollution Attack (GSPA).** Selectively adds new edges between malicious nodes and benign nodes, thereby altering the graph structure to make malicious nodes appear similar to benign nodes.
- **Combined Graph Pollution Attack (CGPA).** Combines both malicious feature manipulation and malicious structure manipulation methods, simultaneously altering the features and structure of malicious nodes to maximize the concealment of malicious behavior and evade detection.

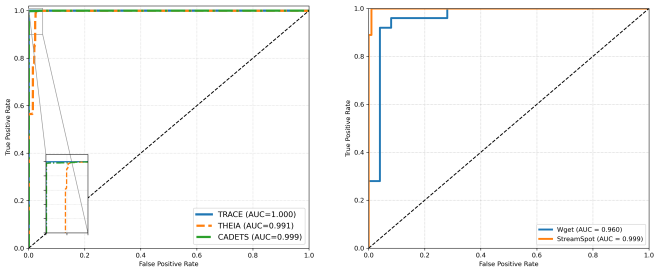
These attacks were simulated by perturbing malicious nodes and their surrounding structures within the victim graph, mimicking realistic attacker behavior aimed at tampering with the graph.

(2) Data Poisoning Attacks. Data poisoning attacks target the training phase, where the attacker perturbs the graph data used for training to compromise the model's robustness. Considering the practical difficulty for attackers to access the model directly, we focused on two types of poisoning attacks:

- **Structure Poisoning Attack (SPA)**: Perturbs a certain proportion of nodes and edges in the training graph by adding or modifying connections, thereby disrupting the original structural features.
- **Feature Poisoning Attack (FPA)**: Alters a certain proportion of node features in the training graph by swapping initial features between nodes, disrupting the feature distribution, and misleading the model during training.

In summary, we provide detailed information about the attacks in Table IV, including the attack targets (nodes or edges) and the perturbation rate (y). Specifically, the perturbation rate (y) represents the proportion of nodes or edges manipulated within the entire graph structure. these attacks are constructed at the node level, while for graph-level detection since the Streamspot and Unicorn datasets only provide anomalous graphs rather than nodes, we extend the attacks to the graph level. Specifically, we randomly select nodes or edges in the graph to be detected for attacks based on the perturbation rate (y).

Metrics. In evaluating the performance of MirGuard, we use a variety of common metrics to comprehensively assess the model's behavior under different tasks and experimental setups. The basic evaluation metrics include Recall (Rec), Precision (Pre), AUC (Area Under the Curve), F1 Score (F1), and Accuracy (Acc). Additionally, we introduced an Absolute



(a) ROC curves (Node). (b) ROC curves (Graph).

Fig. 5: ROC curves on each dataset.

Change Rate (ACR) as an extra metric to evaluate robustness, which is utilized in Figure 7. These metrics provide a holistic understanding of the model's performance, covering its detection capability, classification effectiveness, and balance between different classes.

TABLE V: Comparison of Anomaly Detection Methods

Dataset	Method	Metrics			
		Precision	F1-score	Recall	FPR
Theia	Threatrace	0.87	0.93	**0.99**	0.10%
	MAGIC	0.98	**0.99**	**0.99**	0.14%
	FLASH	0.93	0.96	**0.99**	0.05%
	MirGuard	**0.99**	**0.99**	**0.99**	**0.03%**
Cadets	Threatrace	0.90	0.95	**0.99**	0.20%
	MAGIC	0.94	0.97	**0.99**	0.09%
	FLASH	0.95	0.97	**0.99**	0.16%
	MirGuard	**0.98**	**0.99**	**0.99**	**<0.01%**
Trace	Threatrace	0.71	0.82	**0.99**	1.10%
	MAGIC	**0.99**	**0.99**	**0.99**	0.09%
	FLASH	0.95	0.97	**0.99**	0.16%
	MirGuard	**0.99**	**0.99**	**0.99**	**<0.01%**
Streamspot	Threatrace	0.98	0.99	0.99	0.4%
	MAGIC	0.99	0.99	**1.00**	0.6%
	FLASH	**1.00**	0.96	0.98	0.3%
	MirGuard	0.99	**0.99**	**1.00**	0.6%
Wget	Threatrace	0.93	0.95	0.98	7.4%
	MAGIC	0.96	0.95	**0.96**	2.0%
	FLASH	0.96	**0.96**	**0.96**	2.0%
	MirGuard	**0.98**	**0.96**	**0.96**	**0.6%**

B. MirGuard's Effectiveness (RQ1)

In this section, to evaluate the detection performance of MirGuard and its baseline models, we use precision, F1-score, and recall as evaluation metrics. The experiments are conducted on the Unicorn and Streamspot datasets for graph-level detection and the DARPA dataset for node-level detection. MirGuard adopts a self-supervised training approach, where the model is trained on benign data and evaluated on malicious data for detection.

Detection result. Table V provides the detection results of MirGuard, while Figure 5 shows the ROC curves for each dataset. In the graph-level anomaly detection datasets, Streamspot and Unicorn wget, MirGuard achieved near-perfect detection performance on the simpler Streamspot dataset, with a precision of 99% and a recall of 100%. This high performance is attributed to the dataset's collection of single-user activities per log batch, which are structurally and semantically distinct from each other.

On the more complex Unicorn Wget dataset, MirGuard still achieved high accuracy (96%) and recall (96%). Moving to node-level detection, MirGuard also demonstrated high performance on the DARPA TC datasets, achieving 99% accuracy and 99% recall. Due to the significant disparity between benign and malicious entities, MirGuard was able to accurately identify anomalies. This success is attributed to the use of KMeans for outlier detection, which effectively leverages the distinct feature distributions of benign and malicious entities.

Comparison study. To compare the performance of MirGuard with existing state-of-the-art methods, as described in our experimental setup VI-A, we selected several detectors for both graph-level and node-level anomaly detection, including Threatrace [10], FLASH [14], and MAGIC [12].

As shown in Table V, MirGuard demonstrates outstanding performance across five representative provenance datasets. For node-level detection, it achieves near-perfect results on Theia, Cadets, and Trace. On Theia, MirGuard reaches 0.99 in both precision and F1-score, while achieving the lowest FPR of 0.03%. On the Cadets and Trace datasets, it further reduces the FPR to below 0.01%, outperforming both supervised detectors like Threatrace and FLASH, as well as advanced unsupervised methods like MAGIC. For graph-level detection, MirGuard also shows strong performance on both the Streamspot and Wget datasets. On Streamspot, it achieves a perfect recall of 1.00 and a balanced F1-score of 0.99, matching the results of MAGIC and FLASH. On the more complex Wget dataset, MirGuard leads all baselines with an FPR of only 0.6%, significantly outperforming MAGIC and FLASH (2.0%) and especially Threatrace (7.4%). These results highlight the effectiveness of MirGuard's contrastive learning framework, which leverages multi-view graph augmentation to generate robust and generalizable representations. Unlike MAGIC's masked graph autoencoder approach that focuses on local reconstruction, MirGuard captures both local and global semantics by contrasting positive and negative views. This enables more reliable identification of stealthy or structurally evasive attack behaviors embedded in the provenance graph.

C. Adversarial Robustness Analysis (RQ2)

In this section, we conducted a comprehensive and in-depth evaluation of the robustness of MirGuard against graph tampering attacks. Table VI presents a comparative analysis of various attack types under different perturbation ratios. Threatrace and FLASH exhibit significant performance degradation under structural perturbations, especially at a 50% attack ratio, where their F1 scores drop to 0.489 and 0.657, respectively. This indicates their strong reliance on local neighborhood structures, making them vulnerable to shifts in the embedding space caused by adversarial modifications. MAGIC, which adopts a node-masking strategy, shows improved local robustness and performs reasonably well under low-intensity attacks (0.8 F1 under GSPA=20%). However, its performance deteriorates notably under structurally intensive

TABLE VI: Multi-various Graph Manipulation Attacks under Different Attack Rates.

Attack	Rate(%)	Threatrace [10]			MAGIC [12]			FLASH [14]			MirGuard		
		Precision	F1-score	AUC	Precision	F1-score	AUC	Precision	F1-score	AUC	Precision	F1-score	AUC
None	\	0.904	0.949	0.954	0.944	0.970	0.997	0.947	0.972	0.978	**0.981**	**0.989**	**0.999**
GSPA	10	0.731	0.813	0.910	0.734	0.848	0.921	0.817	0.837	0.921	**0.978**	**0.942**	**0.996**
	20	0.617	0.749	0.854	0.644	0.800	0.862	0.723	0.759	0.889	**0.957**	**0.932**	**0.984**
	50	0.307	0.489	0.756	0.334	0.533	0.745	0.593	0.657	0.828	**0.861**	**0.887**	**0.972**
GFPA	10	0.784	0.878	0.913	0.904	0.959	0.991	0.887	0.922	0.952	**0.979**	**0.988**	**0.999**
	20	0.744	0.843	0.907	0.873	0.950	0.979	0.840	0.896	0.941	**0.976**	**0.976**	**0.998**
	50	0.644	0.797	0.871	0.794	0.920	0.957	0.793	0.871	0.938	**0.967**	**0.963**	**0.988**
CGPA	10	0.767	0.845	0.901	0.784	0.870	0.974	0.807	0.877	0.934	**0.971**	**0.931**	**0.997**
	20	0.693	0.749	0.882	0.713	0.830	0.913	0.787	0.841	0.913	**0.953**	**0.937**	**0.989**
	50	0.484	0.489	0.824	0.527	0.655	0.819	0.667	0.777	0.865	**0.873**	**0.872**	**0.975**
SPA	10	0.783	0.803	0.882	0.769	0.807	0.958	0.876	0.808	0.895	**0.970**	**0.983**	**0.995**
	20	0.674	0.739	0.842	0.628	0.740	0.873	0.677	0.668	0.830	**0.949**	**0.937**	**0.983**
	50	0.494	0.589	0.761	0.571	0.631	0.815	0.572	0.522	0.753	**0.871**	**0.899**	**0.962**
FPA	10	0.834	0.821	0.904	0.904	0.960	0.983	0.877	0.952	0.953	**0.980**	**0.989**	**0.999**
	20	0.785	0.777	0.895	0.884	0.940	0.853	0.853	0.946	0.948	**0.978**	**0.970**	**0.998**
	50	0.744	0.759	0.874	0.807	0.890	0.916	0.790	0.931	0.922	**0.933**	**0.951**	**0.981**

global perturbations (i.e., higher attack ratios), revealing its lack of explicit global structural consistency enforcement.

In contrast, MirGuard, enhanced by multi-view perturbation strategies across nodes, edges, and features, consistently outperforms all baselines across all attack types and ratios. For instance, under CGPA-50%, MirGuard achieves an F1 score of 0.872 and an AUC of 0.975, significantly surpassing other methods. Its strength lies in guiding the model to learn globally robust representations, thereby mitigating the impact of structural manipulations.

Furthermore, we evaluated the robustness of all models under increasing attack ratios (10%, 20%, 50%). The results show that although all methods experience some performance drop under stronger perturbations, MirGuard exhibits the least degradation, consistently maintaining AUC 0.96 and F1 0.87 in nearly all cases. It is also important to note that large-scale perturbations are often difficult to execute stealthily in real-world scenarios and tend to leave more forensic traces. Prior studies [17], [18] have also indicated that high-ratio structural manipulations are challenging to realize in practice. Therefore, MirGuard demonstrates superior robustness against existing graph manipulation attacks.

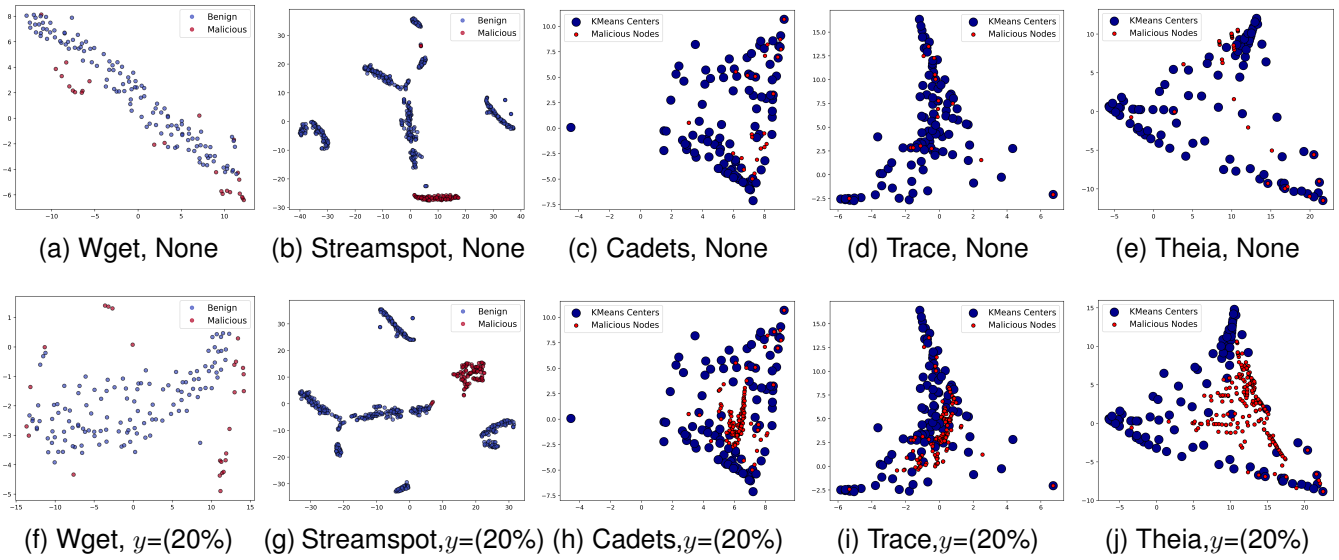
Visualization of Representations. The main contribution of MirGuard lies in its ability to learn high-quality representations that provide a comprehensive understanding of behavioral information, forming a clear decision boundary that effectively distinguishes between benign and malicious behavior nodes. To further analyze the internal representations learned by MirGuard, we visualized its latent representations under a graph manipulation attack (CGPA with an attack ratio of 0.2). We employed the t-SNE technique to project the graph representations of each input sample onto a 2D space. Figure 6 presents the learned representations, where subfigures (a), (b), (c), (d), and (e) depict the feature space distribution in the absence of attacks, while subfigures (f), (g), (h), (i), and (j) illustrate the feature space distribution after the attack.

In the figure, blue points represent benign samples, while red points represent malicious samples. Notably, in the DARPA dataset, due to the large number of nodes, we utilized K-means cluster centroids to approximate the distribution of benign nodes, while the red points indicate malicious nodes. Any node significantly distant from all cluster centroids is considered malicious. It can be observed that MirGuard's learned latent representations are well-structured, allowing for a clear distinction between benign and malicious samples. Importantly, this compact representation and well-defined decision boundary not only enhance the model's performance in classification tasks but also significantly increase the difficulty of adversarial attacks, thereby improving its robustness. After the graph manipulation attack, although some dispersion in the sample distribution is observed, MirGuard still effectively distinguishes between benign and malicious nodes. This can be attributed to MirGuard's use of contrastive learning, which brings samples from different augmented views closer together, enabling the model to focus on global features rather than local features. As a result, the impact of graph manipulation attacks on MirGuard remains minimal.

D. Ablation Study (RQ3)

This section aims to investigate whether different modules in MirGuard, such as multi-view graph augmentation and contrastive learning, can improve its robustness and detection performance. Specifically, we evaluate the robustness of the current design by replacing certain components in MirGuard. Since graph augmentation and contrastive learning are designed to work collaboratively, we demonstrate the necessity of this design from two perspectives. First, we highlight the robustness of MirGuard's graph representation learning by substituting different methods in the representation learning module. Second, we verify the effectiveness of the graph augmentation strategy by introducing various augmentation techniques. Additionally, we further explore the impact of augmentation rate selection and the choice of detection methods.

Effective of GCL Model. To evaluate the robustness of the graph representation learning module in MirGuard, we conducted comparative experiments by replacing it with alternative methods, including GraphSAGE [51], DGI [52], and MGAE [53]. The detection module still adopted the KNN-



(a) Wget, None (b) Streamspot, None (c) Cadets, None (d) Trace, None (e) Theia, None

(f) Wget, y=(20%) (g) Streamspot,y=(20%) (h) Cadets,y=(20%) (i) Trace,y=(20%) (j) Theia,y=(20%)

Fig. 6: The latent representation learned by MirGuard has good discriminability and is able to resist graph manipulation attacks against the model.

TABLE VII: Ablation study on graph representation model

Models	None				CGPA(y=20%)
	Precision	Recall	F1	AUC	F1
MirGuard(DGI)	0.9	0.8	0.91	0.9	0.678
MirGuard(GraphSAGE)	0.92	0.96	0.92	0.94	0.642
MirGuard(MGAE)	0.95	0.92	0.94	0.95	0.839
MirGuard	**0.98**	**0.99**	**0.99**	**0.99**	**0.96**

based strategy. In the experiments, we replaced the GCL module with these embedding methods to learn graph representations and performed anomaly detection. The training and testing datasets remained the same, and all experiments were conducted on the Cadets dataset with the attack ratio set to 0.2 to assess both robustness and detection performance. The results are shown in Table VII. The GCL module achieves the best detection performance. Moreover, the contrastive learning component plays a critical role in enhancing the model's robustness, exhibiting only minimal performance degradation. In comparison, the alternative approaches, namely DGI, GraphSAGE, and MGAE, all result in noticeable performance drops. Among them, MGAE shows the second-best robustness. A possible explanation is that MGAE utilizes a masked reconstruction mechanism that reconstructs the graph structure based on unmasked nodes. This allows some attacked nodes to be masked during training, thereby improving its resistance to certain types of graph manipulation attacks.

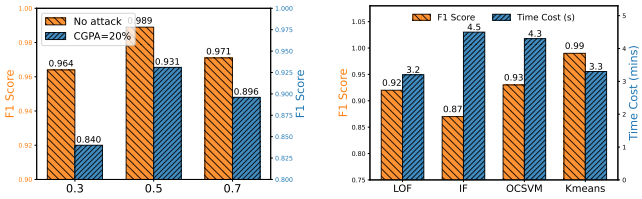
Effective of Graph Augmentation. MirGuard introduces three types of graph augmentation methods. These techniques, through contrastive learning, encourage the model to focus on global behaviors while ignoring local perturbations. To investigate the impact of these graph augmentation strategies on the robustness of MirGuard, we evaluated their effectiveness under a graph contrastive learning framework against different attacks, including GSPA, GFPA, and CGPA. Specifically, we trained the model using various augmentation strategies on the Cadets dataset and assessed its robustness. The experimental results are shown in Table VIII.

TABLE VIII: F1 for Different Augmentation Methods under Various Attack Types($\gamma_{\text{EA}}, \gamma_{\text{NA}}, \gamma_{\text{FA}}$)

Augmentation Type	AUC (CGPA)	AUC (GSPA)	AUC (GFPA)	AUC (None)
NA	0.954	**0.971**	0.967	0.989
EA	0.972	0.961	0.964	0.985
FA	0.965	0.963	0.947	0.973
NA+FA	0.979	**0.971**	**0.988**	0.991
EA+FA	**0.984**	0.959	0.987	0.989
NA+EA+FA	**0.984**	**0.971**	0.984	**0.999**

The results show that in the absence of attacks, applying all three augmentation methods achieves the best performance. Under attack scenarios, different augmentation strategies contribute differently to robustness. For example, NA achieves the highest AUC of 0.971 under GSPA attacks, while the combination of NA and FA performs best under GFPA attacks, achieving an AUC of 0.988. The combination of NA, EA, and FA demonstrates consistent robustness across all attack types, with AUC values ranging from 0.971 to 0.984. Overall, Table VIII indicates that well-designed augmentation strategies can effectively enhance MirGuard's robustness against various attacks. It is worth noting that although the NA+EA+FA combination does not always yield the best result, possibly due to excessive perturbations interfering with feature extraction, selecting appropriate augmentation strategies can still significantly improve adversarial robustness.

In addition, we explored the impact of the augmentation ratio on model robustness. We conducted experiments under both attack and non-attack scenarios. As shown in Figure 7(a), as the augmentation ratio increases, the model's performance first improves and then declines, and a similar trend is observed in its robustness. This is because a low perturbation ratio fails to effectively enhance the model's learning capacity, while an



(a) Ablation on γ (b) Ablation on detector

Fig. 7: Ablation study on detector modules and augment rate γ for MirGuard.

excessively high ratio disrupts the original graph structure. Therefore, we recommend an optimal augmentation ratio of $\gamma = 0.5$.

Effective of Detector. We explored the classification performance of MirGuard by employing different classifiers. To this end, we configured several lightweight classifiers, including Local Outlier Factor (LOF) [41], One-Class Support Vector Machine (OCSVM) [42], and Isolation Forest (IF) [44]. We conducted experiments with multiple detectors on the Cadets dataset, and the results are shown in Figure 7.(b). The experimental results show that IF and OCSVM incurred relatively high time overhead and delivered moderate performance. In contrast, LOF had a similar time cost to KMeans but exhibited slightly lower accuracy. Overall, KMeans achieved the best classification performance while maintaining low time overhead.

E. Performance Overhead (RQ4)

Besides the robustness and effectiveness of MirGuard, efficiency is another critical factor influencing its practical application. In this section, we compare the training and inference costs of MirGuard with its baseline models to evaluate its efficiency. It is important to emphasize that during the training of these detectors, we used the default settings provided in their open-source implementations to achieve optimal performance, and the training process was conducted on the same server and configuration.

Table IX summarizes the training costs of MirGuard and its baseline models. We observed that when trained with the same batch size, MirGuard and MAGIC exhibit similar time and memory overheads. On the other hand, FLASH and Threatrace adopt strategies that allow them to converge with smaller batch sizes and fixed graph sizes within each batch, which reduces memory overhead but results in longer training times. Regarding inference overhead, MirGuard maintains the best inference time and relatively low memory consumption. Therefore, overall, compared to the training costs of baseline detectors, it can be concluded that MirGuard ensures robustness without sacrificing efficiency.

VII. Related Work

Provenance-based IDS. Recently, provenance-based IDS methods have been categorized into three main types: learning-based, statistical-based, and rule-based approaches. Statistical

TABLE IX: Performance Comparison of Different Methods in Terms of Training Time and Memory Usage

Phase	Metric	FLASH	MAGIC	Threatrace	MirGuard
Train	Total Time (s)	4,580	**151**	2,780	214
	Memory (MB)	760	1,564	1,031	**1,525**
Inference	Total Times (s)	4,304	1,037	1,380	**437**
	Memory (MB)	**1,097**	1,667	2,301	1,532

methods [29], [30], [34], [54] model the anomaly degree of nodes using features such as temporal correlation, degree distribution, and rarity. Rule-based methods [4]–[7], [55], [56] create rules based on external knowledge to progressively match patterns in the provenance graph for anomaly detection. Learning-based approaches [8], [10]–[13], [15], [21], [32], [57] include sequence learning to extract and model sequence features for anomaly detection, as well as deep graph learning techniques for graph-level and edge-level detection using features like graph snapshots [8], [13], [28] and node interactions [11]. Recent research has also explored node-level detection within provenance graphs, laying the foundation for fine-grained anomaly analysis. Threatrace [10] uses GraphSAGE for node embedding and anomaly detection, while MAGIC [12] employs MGAE for unsupervised graph representation learning and KNN-based anomaly detection. FLASH [14] combines a GNN with Word2Vec for feature extraction and designs a caching mechanism to support scalability real-time detection.

Graph Manipulation Attacks for Provenance-based Detector. Wagner et al. [58] first introduced mimicry attacks in 2002, enabling attackers to evade IDS detection. Their theoretical framework laid the foundation for circumventing PIDS. Li et al. [59] questioned the robustness of provenance-based detectors, highlighting risks like dependency explosion and proposing mimicry-based circumvention methods. Goyal et al. [16] demonstrated the first practical evasion attacks against P-IDS in 2023. Kunal et al. [17] advanced this with the PROVNINJA framework, reducing new system events and expanding tolerable distribution differences. Sang et al. [18] proposed an obfuscation attack strategy, introducing meta-behavior mapping for realistic evasion, noting that large-scale graph tampering is impractical for attackers.

Robustness of Graph Neural Networks. The robustness of GNN has gained increasing attention in recent years due to its wide applications in critical domains such as social networks, recommendation systems, and cybersecurity. However, studies have shown that GNNs are vulnerable to adversarial attacks, which manipulate graph structures, node features, or both to degrade model performance. These attacks are generally categorized into evasion attacks and poisoning attacks. Evasion attacks target the inference phase by perturbing graph data to mislead model predictions [60], while poisoning attacks modify training data to undermine model robustness before deployment [61], [62].

To address these threats, researchers have proposed various defense mechanisms. Adversarial training is one of the most widely studied methods, which enhances model robustness by injecting adversarial perturbations during training [60]. Additionally, graph data preprocessing techniques, such as

graph sanitization [63], aim to filter out adversarial perturbations, while robust GNN architectures leverage mechanisms like attention mechanisms or spectral filtering to strengthen resistance against attacks [64], [65]. Recently, contrastive learning has emerged as a promising direction to improve GNN robustness by combining data augmentations and contrasting positive and negative samples, effectively learning more robust graph embeddings [24].

VIII. Discussion

Graph manipulation attacks. Recently, graph manipulation attacks have posed significant challenges to the performance of provenance-based detectors [16]–[18]. Attackers maliciously alter graph structures, causing the graph encoding of malicious nodes to resemble that of normal nodes, leading to false negatives. MirGuard demonstrates notable advantages in addressing these challenges, primarily due to our innovative approach of introducing various types of perturbations during the training phase and generating embeddings using contrastive learning. This method takes into account the potential manipulations of the graph structure by attackers during training, rendering attempts to alter the structure and features of malicious nodes ineffective. Therefore, MirGuard suggests a promising approach for countering such graph manipulation attacks in the future.

IX. Conclusion

In this study, we introduced MirGuard, a novel graph learning-based anomaly detection system designed to enhance the robustness of provenance-based intrusion detection. By integrating multi-view augmentations with contrastive learning, MirGuard effectively mitigates mimicry attacks that manipulate graph structures. Comprehensive evaluations on multiple datasets demonstrate that MirGuard outperforms state-of-the-art detectors in both robustness and detection accuracy (achieving an average F1-score of over 96% with less than 10% AUC drop under graph manipulation attacks), without compromising efficiency (with overhead comparable to existing detectors). Our work provides a robust solution to modern cybersecurity challenges, paving the way for more robust provenance-based intrusion detection systems.

References

[1] "Apt notes." https://github.com/kbandla/APTnotes, Last accessed on 2024-6-25.

[2] M. A. Inam, Y. Chen, A. Goyal, J. Liu, J. Mink, N. Michael, S. Gaur, A. Bates, and W. U. Hassan, "Sok: History is a vast early warning system: Auditing the provenance of system intrusions," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 307–325, IEEE Computer Society, 2022.

[3] F. Dong, S. Li, P. Jiang, D. Li, H. Wang, L. Huang, X. Xiao, J. Chen, X. Luo, Y. Guo, *et al.*, "Are we there yet? an industrial viewpoint on provenance-based endpoint detection and response tools," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2396–2410, 2023.

[4] M. N. Hossain, S. M. Milajerdi, J. Wang, B. Eshete, R. Gjomemo, R. Sekar, S. D. Stoller, and V. Venkatakrishnan, "Sleuth: Real-time attack scenario reconstruction from cots audit data.," in *USENIX Security Symposium*, pp. 487–504, 2017.

[5] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: real-time apt detection through correlation of suspicious information flows," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1137–1152, IEEE, 2019.

[6] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1172–1189, IEEE, 2020.

[7] T. Zhu, J. Yu, C. Xiong, W. Cheng, Q. Yuan, J. Ying, T. Chen, J. Zhang, M. Lv, Y. Chen, *et al.*, "Aptshield: A stable, efficient and real-time apt detection system for linux hosts," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[8] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang, "Prographer: An anomaly detection system based on provenance graph embedding," in *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 4355–4372, 2023.

[9] Q. Wang, W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, J. Rhee, Z. Chen, W. Cheng, C. A. Gunter, *et al.*, "You are what you do: Hunting stealthy malware via data provenance analysis.," in *NDSS*, 2020.

[10] S. Wang, Z. Wang, T. Zhou, H. Sun, X. Yin, D. Han, H. Zhang, X. Shi, and J. Yang, "Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3972–3987, 2022.

[11] J. Zeng, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua, "Shadewatcher: Recommendation-guided cyber threat analysis using system audit records," in *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 489–506, IEEE, 2022.

[12] Z. Jia, Y. Xiong, Y. Nan, Y. Zhang, J. Zhao, and M. Wen, "Magic: Detecting advanced persistent threats via masked graph representation learning," in *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 5197–5214, 2024.

[13] Z. Cheng, Q. Lv, J. Liang, Y. Wang, D. Sun, T. Pasquier, and X. Han, "Kairos: Practical intrusion detection and investigation using whole-system provenance," in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 3533–3551, IEEE, 2024.

[14] M. U. Rehman, H. Ahmadi, and W. U. Hassan, "Flash: A comprehensive approach to intrusion detection via provenance graph representation learning," in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 139–139, IEEE Computer Society, 2024.

[15] Y. Shen and G. Stringhini, "Attack2vec: Leveraging temporal word embeddings to understand the evolution of cyberattacks," *28 st USENIX Security Symposium (USENIX Security 2019)*, 2019.

[16] A. Goyal, X. Han, G. Wang, and A. Bates, "Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems," in *30th Network and Distributed System Security Symposium*, 2023.

[17] K. Mukherjee, J. Wiedemeier, T. Wang, J. Wei, F. Chen, M. Kim, M. Kantarcioglu, and K. Jee, "Evading provenance-based ml detectors with adversarial system actions," in *32nd USENIX Security Symposium (USENIX Security 23)*, (Anaheim, CA), pp. 1199–1216, USENIX Association, Aug. 2023.

[18] A. Sang, Y. Wang, L. Yang, J. Jia, and L. Zhou, "Obfuscating provenance-based forensic investigations with mapping system meta-behavior," in *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*, RAID '24, 2024.

[19] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 2206–2216, PMLR, 2020.

[20] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 274–283, PMLR, 2018.

[21] X. Han, X. Yu, T. Pasquier, D. Li, J. Rhee, J. Mickens, M. Seltzer, and H. Chen, "Sigl: Securing software installations through deep graph learning," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2345–2362, 2021.

[22] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia, X. Xing, C. L. Giles, and X. Liu, "Adversary resistant deep neural networks with an application to malware detection," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1145–1153, ACM, 2017.

[23] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," in *2018 IEEE Symposium on Security and Privacy Workshops (SPW)*, pp. 76–82, IEEE, 2018.

[24] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in neural information processing systems*, vol. 33, pp. 5812–5823, 2020.

[25] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *International conference on machine learning*, pp. 4116–4126, PMLR, 2020.

[26] "Transparent computing engagement 3 datarelease.." https://github.com/darpa-i2o/TransparentComputing/blob/master/README-E3.md, Last accessed on 2024-5-21.

[27] "Streamspot dataset." https://github.com/sbustreamspot/sbustreamspot-data, Last accessed on 2024-5-29.

[28] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," *NDSS*, 2020.

[29] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, "Nodoze: Combatting threat alert fatigue with automated provenance triage," in *network and distributed systems security symposium*, 2019.

[30] P. Fang, P. Gao, C. Liu, E. Ayday, K. Jee, T. Wang, Y. F. Ye, Z. Liu, and X. Xiao, "{Back-Propagating} system dependency impact for attack investigation," in *31st USENIX Security Symposium (USENIX Security 22)*, pp. 2461–2478, 2022.

[31] Y. Liu, M. Zhang, D. Li, K. Jee, Z. Li, Z. Wu, J. Rhee, and P. Mittal, "Towards a timely causality analysis for enterprise security.," in *NDSS*, 2018.

[32] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, "Atlas: A sequence-based learning approach for attack investigation.," in *USENIX Security Symposium*, pp. 3005–3022, 2021.

[33] A. Bates, D. J. Tian, K. R. Butler, and T. Moyer, "Trustworthy {Whole-System} provenance for the linux kernel," in *24th USENIX Security Symposium (USENIX Security 15)*, pp. 319–334, 2015.

[34] T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon, "Practical whole-system provenance capture," in *Proceedings of the 2017 Symposium on Cloud Computing*, pp. 405–418, 2017.

[35] R. Paccagnella, P. Datta, W. U. Hassan, A. Bates, C. Fletcher, A. Miller, and D. Tian, "Custos: Practical tamper-evident auditing of operating systems using trusted execution," in *Network and distributed system security symposium*, 2020.

[36] J. Zeng, C. Zhang, and Z. Liang, "Palantír: Optimizing attack provenance with hardware-enhanced system observability," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3135–3149, 2022.

[37] C. Zhang, J. Zeng, Y. Zhang, A. Ahmad, F. Zhang, H. Jin, and Z. Liang, "The hitchhiker's guide to high-assurance system observability protection with efficient permission switches," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 3898–3912, 2024.

[38] S. T. King and P. M. Chen, "Backtracking intrusions," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pp. 223–236, 2003.

[39] S. T. King, Z. M. Mao, D. G. Lucchetti, and P. M. Chen, "Enriching intrusion alerts through multi-host causality.," in *Ndss*, Citeseer, 2005.

[40] Z. Xu, Z. Wu, Z. Li, K. Jee, J. Rhee, X. Xiao, F. Xu, H. Wang, and G. Jiang, "High fidelity data reduction for big data security dependency analyses," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 504–516, 2016.

[41] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

[42] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," *Advances in neural information processing systems*, vol. 12, 1999.

[43] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, pp. 986–996, Springer, 2003.

[44] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*, pp. 413–422, IEEE, 2008.

[45] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1035–1044, 2016.

[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[47] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, *et al.*, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[48] L. Wang, X. Shen, W. Li, Z. Li, R. Sekar, H. Liu, and Y. Chen, "Incorporating gradients to rules: Towards lightweight, adaptive provenance-based intrusion detection," *arXiv preprint arXiv:2404.14720*, 2024.

[49] B. Jiang, T. Bilot, N. El Madhoun, K. Al Agha, A. Zouaoui, S. Iqbal, X. Han, and T. Pasquier, "ORTHRUS: Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems," in *Security Symposium (USENIX Sec'25)*, USENIX, 2025.

[50] A. Goyal, G. Wang, and A. Bates, "R-caid: Embedding root cause analysis within provenance-based intrusion detection," in *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*, pp. 257–257, IEEE, 2024.

[51] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[52] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax.," *ICLR (Poster)*, vol. 2, no. 3, p. 4, 2019.

[53] Q. Tan, N. Liu, X. Huang, R. Chen, S.-H. Choi, and X. Hu, "Mgae: Masked autoencoders for self-supervised learning on graphs," *arXiv preprint arXiv:2201.02534*, 2022.

[54] Y. Liu, X. Shu, Y. Sun, J. Jang, and P. Mittal, "Rapid: Real-time alert investigation with context-aware prioritization for efficient threat discovery," in *Proceedings of the 38th Annual Computer Security Applications Conference*, pp. 827–840, 2022.

[55] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. Venkatakrishnan, "Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pp. 1795–1812, 2019.

[56] M. N. Hossain, S. Sheikhi, and R. Sekar, "Combating dependence explosion in forensic analysis using alternative tag propagation semantics," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1139–1155, IEEE, 2020.

[57] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pp. 1777–1794, 2019.

[58] D. Wagner and P. Soto, "Mimicry attacks on host-based intrusion detection systems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 255–264, 2002.

[59] Z. Li, R. Yang, Q. A. Chen, and Y. Chen, "Mimic the whole attack chain: A first look at evasion against provenance graph based detection," 2020.

[60] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *International conference on machine learning*, pp. 1115–1124, PMLR, 2018.

[61] L. Sun, Y. Dou, C. Yang, K. Zhang, J. Wang, S. Y. Philip, L. He, and B. Li, "Adversarial attack and defense on graph data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 7693–7711, 2022.

[62] K. Xu, X. Ma, L. Liu, D. Deb, J. Liu, J. Tang, J. Fan, J. Bailey, and D. Song, "Topology attack and defense for graph neural networks: An optimization perspective," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3961–3967, 2019.

[63] X. Wu, X. Shi, H. Cheng, J. Zhou, and Y. Li, "Adversarial examples on graph data: Deep insights into attack and defense," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3776–3782, 2019.

[64] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 66–74, 2020.

[65] J. Guo, K. Huang, X. Yi, Z. Su, and R. Zhang, "Rethinking spectral graph neural networks with spatially adaptive filtering," *arXiv preprint arXiv:2401.09071*, 2024.