

PROVX: Generating Counterfactual-Driven Attack Explanations for Provenance-Based Detection

Weiheng Wu^{*†}, Wei Qiao^{*†}, Teng Li^{*†}, Yebo Feng[‡], Zhuo Ma^{*}, Jianfeng Ma^{*}, and Yang Liu[‡]

^{*}School of Cyber Engineering, Xidian University, Xi'an, China

[†]State Key Laboratory of Integrated Services Networks (ISN), Xi'an, China

[‡]Nanyang Technological University, Singapore

{wdz478149507, qiaoweiXidian, litengxidian}@gmail.com, {mazhuo, jfma}@mail.xidian.edu.cn, {yebo.feng, yangliu}@ntu.edu.sg

Abstract—Provenance graph-based intrusion detection systems are deployed on hosts to defend against increasingly severe Advanced Persistent Threat. Using Graph Neural Networks to detect these threats has become a research focus and has demonstrated exceptional performance. However, the widespread adoption of GNN-based security models is limited by their inherent black-box nature, as they fail to provide security analysts with any verifiable explanations for model predictions or any evidence regarding the model's judgment in relation to real-world attacks.

To address this challenge, we propose PROVX, an effective explanation framework for explaining GNN-based security models on provenance graphs. PROVX introduces counterfactual explanation logic, seeking the minimal structural subset within a graph predicted as malicious that, when perturbed, can subvert the model's original prediction. We innovatively transform the discrete search problem of finding this critical subgraph into a continuous optimization task guided by a dual objective of prediction flipping and distance minimization. Furthermore, a Staged Solidification strategy is incorporated to enhance the precision and stability of the explanations.

We conducted extensive evaluations of PROVX on authoritative datasets. The experimental results demonstrate that PROVX can locate critical graph structures that are highly relevant to real-world attacks and achieves an average explanation necessity of 51.59%, with these metrics outperforming current SOTA explainers. Furthermore, we explore and provide a preliminary validation of a closed-loop Detection-Explanation-Feedback enhancement framework, demonstrating through experiments that the explanation results from PROVX can guide model optimization, effectively enhancing its robustness against adversarial attacks.

I. INTRODUCTION

System-level detection of Advanced Persistent Threat (APT) is a significant challenge in modern cybersecurity. To counter these threats, which are characterized by long dormancy periods and complex attack chains, the research community has recently focused on the host interior, leveraging operating system audit logs to capture and represent the complete causal fabric of system execution (Fig. 1)¹. By constructing discrete log entries into a provenance graph—where system entities (e.g., processes, files) are nodes and their causal relationships (e.g., fork, read, write) are edges—a structured view that clearly depicts the complete causal chain of system activities

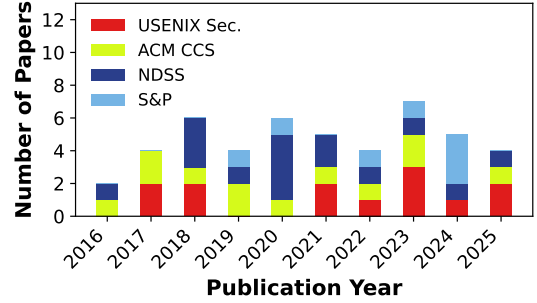


Fig. 1. Statistics on the number of security conferences on provenance graph detection from 2016 to 2025 [1]–[11], [14]–[16], [18]–[26], [28]–[52].

is provided for Provenance-based Intrusion Detection Systems (PIDS) [1]–[27].

Analyzing provenance graphs to detect malicious activity is a core task of PIDS. Early research efforts included rule-based methods that use predefined attack pattern templates for subgraph matching [2], [10], [23], and statistics-based anomaly detection based on macroscopic graph features (e.g., graph degree, centrality) [3], [4], [53]. However, these methods are highly dependent on prior knowledge or hard to distinguish between normal recordings and previously unobserved but semantically related activity, and often perform poorly in real-world environments with vast and noisy log data. Consequently, the research community's focus has gradually shifted to learning-based methods, which can automatically learn and represent the deep patterns in graph-structured data. The core principle of these methods is to train a model that maps system entities and their relationships into low-dimensional vector representations (i.e., embeddings), thereby capturing complex patterns to distinguish between benign behaviors and malicious attacks [7], [12], [16], [17], [20], [21], [24], [26], [50], [54], [55].

The graph structure of provenance graphs makes them naturally suitable for analysis using graph representation learning. In recent years, using Graph Neural Networks (GNNs) to detect APTs on provenance graphs has become a research hotspot. As shown in TABLE I, a large number of related research results have emerged in top cybersecurity conferences. By designing various graph embedding and aggregation

¹As of the submission of this article, some articles from 2025 may not be included in the statistics because they have not been fully published.

strategies, these studies have shown excellent performance in anomaly detection in provenance graphs.

However, despite the detection efficacy achieved by GNNs, their opaque, black-box nature severely limits security analysis, creating a fundamental crisis of **trustworthiness** and **usability** [52], [56], [57]. The model’s output fails to provide human operators with any verifiable explanations for its predictions or any evidence related to a real attack, leading to a significant trust deficit. For a security analyst, an alert without an explanation is nearly unactionable. When a security model merely outputs a malicious label, it leaves a significant gap between the analyst and an executable response. The analyst cannot know: Was this alert triggered by a genuine, novel 0-day attack, or is it merely a false positive generated by the model for an anomalously behaving benign administrator script? This opacity not only consumes immense analytical effort and delays critical incident response times but also fundamentally erodes human trust in automated systems, preventing them from being entrusted with critical responsibilities in real-world security operations. This has become a core bottleneck hindering advanced AI technologies like GNNs from realizing their full potential in the security domain. Moreover, adaptive and feedback loops post-detection are severely lacking in existing work, which casts doubt on the usability of PIDS in real-world environments.

While existing research [58]–[63] has begun to discuss GNN decision-making and provide human-understandable explanations, work on model interpretability remains scarce in the security domain, primarily focusing on binary analysis [64] and code vulnerability detection [65]–[67]. For APT detection, such work is even rarer and is concentrated on fact-based explanation methods, which source evidence from empirical input data to justify a specific outcome. Current approaches either use surrogate models to indirectly explain the security model [68] or exploit the change in permutation importance to identify a subset of anomalies that may have led to the detection [69]. However, analysts remain uncertain about the actual impact of the entities and interactions that constitute the explanation subgraph, casting doubt on the results.

In this paper, we design and implement PROVX, a counterfactual explainer for GNN-based detection models on provenance graphs. PROVX is designed to answer the question: ”For an attack graph classified as malicious, what is the most critical combination of nodes or edges that, had they not occurred, would have flipped the model’s prediction?” This approach intuitively highlights for the analyst the root cause of the alert. Furthermore, we explore the potential of a closed-loop enhancement framework that progresses from Detection to Explanation, and finally to Feedback, where explanations guide model optimization.

In terms of design, the core idea of PROVX is to ingeniously transform the discrete problem of graph structure edit search into a continuous optimization task. Specifically, modifications to the graph structure are achieved by learning a numerically differentiable edge perturbation mask that shares the same dimensions as the graph’s adjacency matrix. This learning

TABLE I
ANALYSIS OF SOME OF THE RESEARCH ON PROVENANCE DIAGRAM THAT HAVE ATTRACTED ATTENTION IN RECENT YEARS.

System	Time	Encoder	Decoder	Adaption
SLEUTH [23] ¹	2017	-	-	✗
NoDOZE [4] ²	2019	-	-	✗
SIGL [19]	2021	Graph LSTM	MLP	✗
ATLAS [7]	2021	LSTM+CNN	-	✗
THREATHTRACE [17]	2022	GraphSAGE	-	✓
SHADEWATCHER [21]	2022	TransR+GNN	Inner Product	✓
MAGIC [18]	2024	GAT	GAT+MLP	✓
KAIROS [20]	2024	TGN	MLP	✗
R-CAID [24]	2024	GAT	-	✗
FLASH [26]	2024	GraphSAGE	XGBoost	✗
SLOT [50]	2025	GCN+RL	MLP	✗

¹ SLEUTH builds rules based on typical or specific APT patterns and matches these rules to detect.

² NoDOZE measures anomalies based on the frequency of related events.

process is guided by a dual-objective optimization framework: one is the Prediction Flip Loss, which drives the model to find structural changes sufficient to reverse the original malicious prediction; the other is the Mask Distance Loss, which ensures that modifications to the graph are as minimal as possible, thereby locating the most critical, core set of key edges. Finally, PROVX introduces an innovative Staged Solidification strategy. In the later stages of training, this strategy locks in the importance of edges that have become clear and penalizes optimization attempts that deviate from this trend, thereby significantly enhancing the stability and clarity of the explanation results. Ultimately, The framework is able to achieve prediction flipping and output high-confidence counterfactual explanations in a minimal perturbation scheme based on the three most widely used PIDS basis models.

We evaluate the performance of PROVX on the widely used and authoritative APT datasets, DARPA E3 [70] and DARPA OpTC [71]. These datasets contain real-world attacks, allowing for a realistic reflection of our explainer’s capabilities. We design experiments to measure the relevance of the explanation results to the actual ground truth, and we use the Probability of Necessity (PN) to evaluate the necessity of the explanations output by PROVX. We conduct extensive performance analyses, hyperparameter analyses, and overhead analyses, and compare PROVX with state-of-the-art explainers. The experimental results demonstrate that PROVX outperforms existing methods on most metrics. Finally, we design an adversarial attack scenario, explain the resulting evasive attack graphs, and explore the potential of a feedback mechanism between the explainer and the detector. In summary, this paper makes the following contributions:

- We propose PROVX, an explainer for the domain of APT detection on provenance graphs, designed to provide explanations for the judgments of black-box security models that are understandable to security analysts.
- Distinct from existing works, PROVX is based on a counterfactual explanation mechanism, transforming the explanation problem into a search for the minimal struc-

tural subset that can subvert the security model’s prediction. It also incorporates a novel Staged Solidification mechanism to ensure the stability and clarity of the explanations.

- We explore and validate a closed-loop enhancement framework from detection to explanation and back to feedback. Experiments demonstrate that the explainer can enhance the robustness of the model against adversarial attacks under the guidance of security analysts.
- We conduct extensive evaluations on widely recognized APT datasets, validating both the relevance of PROVX’s explanations to real-world attacks and the necessity of these explanations. We compare PROVX with SOTA explainers, and the necessity of explanation on three GNN models is 7.75%, 15.91% and 34.05% higher than the SOTA average level.

II. BACKGROUND AND MOTIVATION

A. Background and Related work

1) *System Provenance*: System provenance techniques are widely adopted in academia to conduct attack investigations on large-scale audit logs from hosts. Specifically, by constructing audit logs into provenance graphs (the construction method of which can be referred to in XI), we can obtain a more intuitive and information-rich representation of system execution processes. Leveraging provenance graphs, security analysts can perform causal analysis on system activities, thereby enabling intrusion detection and security event attribution. Based on the utilization of data provenance, current research primarily focuses on learning-based PIDS.

In recent years, the effectiveness of PIDS employing Deep Learning techniques, especially in defending against APT, has been demonstrated. Notably, even systems that demonstrate excellent performance in detecting APT—such as UNICORN [16], which utilizes graph sketching techniques and evolutionary behavior modeling to detect APT; KAIROS [20], which employs a GNN-based encoder-decoder architecture to quantify the abnormality of system events and reconstruct attack footprints; MAGIC [18], which leverages mask graph representation learning to model benign system entity behaviors and achieve high-precision multi-granularity detection; and FLASH [26], Word2Vec-based semantic encoding of the basic semantics of the audit log, and learning using GraphSAGE—often suffer from their black-box nature, making it difficult for analysts to gain insight into their decision-making rationale. In the latest work [52], [57], multiple works including MAGIC, KAIROS and FLASH were systematically analyzed, and this phenomenon was keenly confirmed. Since human experts cannot directly comprehend the learning logic of these black-box models, they often rely on post-detection manual review to locate the cause of attacks, which incurs significant time and economic costs. In view of this, there is an urgent need for an PIDS explainer oriented towards APT scenarios to provide security analysts with more instructive security insights.

2) *Explanations generated for security models*: In the field of eXplainable Artificial Intelligence (XAI), researchers have recently focused on generating understandable GNN explanations for human experts. For instance, fact-based explainers such as PGExplainer [59], SubgraphX [60], and GNNExplainer [58] generate explanations by probing the internal mechanisms of GNNs to identify critical edges, nodes, or important substructures. However, the depth of these explanations is limited to “which input features the model’s current decision relies on”.

Furthermore, it is noteworthy that despite the increasing application of GNNs in the security domain, research on the interpretability of these security models has not received adequate attention. Particularly in the field of APT detection, related research is especially scarce, and existing studies have their respective limitations. For example, [69] systematically perturbs the analyzed provenance graph and continuously observes how the anomaly score output by the target APT detection model changes to derive explanations. However, limited by the fidelity of the explanations and the constraints of fact-based explainers, the authors eventually had to concede that not all perturbation strategies are universally applicable. [68] pioneered the exploration of using interpretable surrogate models (e.g., decision trees) to provide instance-level factual explanations. However, this approach relies on customized, specific graph structures and uses simple machine learning models as substitutes for GNNs. These explanation methods are still not intuitive or sufficient, cannot directly reveal the direct reasons for security model judgment, and are not easy to understand. In this paper, we focus on the graph-level explanation task and use various industry-standard graph neural network models to model provenance graphs and provide trustworthy explanations that are easy to analyze.

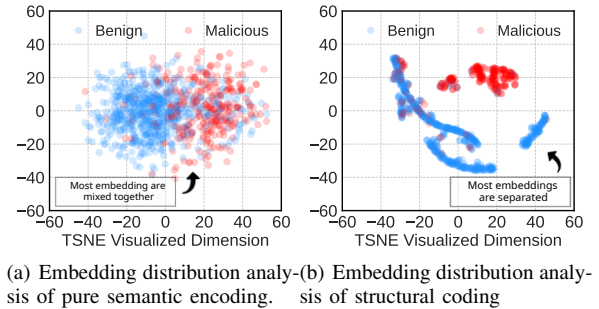


Fig. 2. Analysis of embedding distribution of benign and malignant subgraph patterns.

B. Explanation Necessity and Motivating Example

1) *Key Observation*: At the beginning of the research, we performed a simple division and processing of PG to obtain several smaller traceability subgraphs, each representing a specific attack pattern or benign behavior, hoping to find some clues from them. An interesting and key observation is that the structure of attack-prone subgraphs differs significantly from that of purely benign subgraphs. In contrast, their semantic properties do not show such obvious differences.

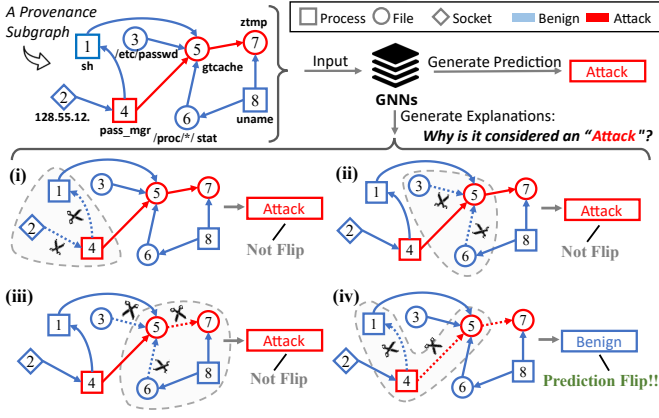


Fig. 3. An attack example from DARPA E3 Trace and how to make counterfactual explanations, (i)-(iv) represent different situations in different counterfactual reasoning, which will lead to two results, the model predicts *Not Flip* or *Prediction Flip*.

Specifically, Fig. 2 shows the TSNE comparison of benign and malicious subgraphs extracted from multiple real APT datasets in terms of semantic similarity and structural similarity. To calculate semantic similarity, we encode the textual attributes of each node in the subgraph (e.g., process name, username, command line parameters, file path, IP address, etc.), aggregate the resulting vectors into a graph-level representation, and then compare the cosine similarity between graphs. For structural similarity, we randomly generate node features and use GNNs to learn embeddings. The results clearly show that structural features are more helpful in distinguishing APT attack behaviors than semantic attributes of system entities. This finding is also confirmed by mature network security frameworks such as Kill Chain and ATT&CK [72]. In summary, topological structure is an important basis for guiding our efforts to mine and understand malicious attack patterns.

2) *Counterfactual explanations for provenance graphs*: We use the APT attack shown in the Fig. 3 as a driving example for illustration. This instance originates from an attack segment within the DARPA TC Trace dataset. In this scenario, the attacker infiltrates the target host via a browser extension named `pass_mgr` and subsequently executes the `gtcache` program. `gtcache` then establishes communication with the attacker’s Command and Control (C&C) server. Its primary objective is to scan the host for sensitive information and operate on the `/ztmp` directory, preparing for subsequent attack stages. For clarity in the subsequent discussion, each system entity in the figure has been numbered.

In the above attack, using existing fact-based explainers, taking SubgraphX as a nexample, typically identifies a subgraph that contributes significantly to the malicious prediction, for example: “The structure formed by nodes ③, ⑤, ⑥, ⑦ and ⑧ is crucial to the predicted outcome”. However, this explanation can still be confusing for security analysts. It merely identifies a suspicious area but fails to reveal the root cause or critical turning point that led the model to

conclude an attack. A key question remains: How do system interactions within this crucial subgraph influence the security model’s judgment? In fact, rather than determining whether a set of edges is crucial, finding some core operations is the real explanation requirement.

Therefore, we adopt a counterfactual explanation approach to explore various hypothetical structural changes (as shown in (i), (ii), (iii), and (iv) in Fig. 3). Our goal is to investigate: after what structural modifications to the original provenance graph will the detection system make a prediction opposite to the original one. For instance:

- (i) After deleting edges ②-④ and ④-① (representing partial interaction between `pass_mgr` and `sh`), the subgraph’s predicted outcome is still “attack”. This suggests that the operation of `pass_mgr` on `sh` represented by ④-① may have malicious tendencies, but its individual effect is not the fundamental reason for the system classifying the entire subgraph as malicious.
- (ii) and (iii) Modifications similar to these might also yield comparable results.
- However, in (iv), if both edge ④-① (interaction between `pass_mgr` and `sh`) and edge ④-⑤ (execution of `gtcache` by `pass_mgr`) are deleted, the detection system flips the predicted outcome to “benign”. This counterfactual result reveals the critical significance of the `pass_mgr` executing `gtcache` operation in the security determination, and also indicates that the interaction between `pass_mgr` and `sh` is likewise an important part of the malicious topological structure learned by the detection system.

This type of explanation, independent of the ground truth, can assist security analysts in overcoming the limitations of black-box models, thereby obtaining a more thorough and easily understandable analysis of the detection results.

III. PROBLEM FORMALIZATION

A. Definition

We define key designs and notations for the task. Assume we have N provenance subgraphs $\{G_1, G_2, G_3, \dots, G_N\}$, where each subgraph G_k corresponds to a ground truth label $Y_k \in \{0, 1\}$. Here, 0 indicates that all system entities within the subgraph are benign (i.e., the subgraph is benign), while 1 indicates the presence of malicious system entities (i.e., the subgraph contains at least one stage of an APT attack).

For model selection, we use GNN to learn from source subgraphs. The learned embeddings are then fed into a simple downstream classifier to perform the attack prediction task on these source subgraphs. During the testing phase, we use the trained security model $f(\cdot)$ to predict the probability for each subgraph G_k and obtain its predicted outcome \hat{Y}_k .

The counterfactual explainer aims to analyze and output the minimal structural information that causes the security model’s prediction to flip. For a subgraph G_k , we introduce a small perturbation to generate its counterfactual instance \tilde{G}_k . Note that the predicted class for this instance by the

security model is opposite to the original prediction, i.e., $f(\tilde{G}_k) \neq f(G_k)$. Our objective is to minimize the difference between the perturbed subgraph and the original subgraph, such that the prediction flips. The mathematical definition is as follows:

$$\begin{aligned} & \min_{\tilde{G}_k} d(\tilde{G}_k, G_k), \\ & \text{s.t., } \hat{Y}_k \neq \hat{Y}_k. \end{aligned} \quad (1)$$

B. Threat Model and Explanation Principle

In our threat model setting, we assume that attackers will leave traceable behavioral patterns and attack traces in the audit logs of the target system. These traces enable the security model we construct to effectively distinguish between malicious attack activities and normal benign system behaviors. Similar to other related research [2], [9], [15], [16], we presume that the data collected by audit logging tools is complete and acquired on top of a Trusted Computing Base (TCB). This assumption ensures the reliability and authenticity of the provenance graphs we construct [73], [74].

Regarding the evaluation of the explanation results, we will refer to the method in [17], [18], [26], [50] and use public documents containing real attack annotations as the basis for evaluating the accuracy of the explanation. Considering the increasing prevalence of adversarial attacks in the current field of network security, we also carefully evaluate the performance changes of the explanation method proposed in this paper when facing the attack mode that has been obfuscated or adversarially modified. The relevant discussion and experimental results will be presented in detail in the experimental part VII of this paper.

IV. PROVENANCE DETECTION

Our explainer is built upon an efficient APT detection model, and here we define our graph-level detection task. Given host audit log data, which is constructed into a provenance graph containing system entities and their interaction information, the goal of provenance detection is to utilize a GNN to analyze this graph and determine if it represents APT activity. This process aims to identify key patterns or substructures within the provenance graph that are related to APT behavior.

Existing detectors use GNNs for autoencoding [18], semi-supervised detection [50], and others. However, ProvX only handles downstream detection tasks, so we simply map APT detection to a GNN-based subgraph classification task. Specifically, we focus on particular structures within provenance subgraphs (e.g., suspicious processes, abnormally accessed files, or interactions between critical system events) and attempt to classify these structures and their local environments to determine if they are involved in malicious activities.

Next, we detail the steps of provenance detection:

(1) Provenance Graph Construction. Host audit logs are first processed and converted into a provenance graph $G_{host} = (V, E)$, where nodes V represent system entities or events that occur, and edges E represent causal dependencies between

these entities (e.g., a process creating a file, a process sending data over the network, etc.). This provenance graph G_{host} serves as the basis for subsequent subgraph extraction.

(2) Subgraph Extraction. We use an improved Louvain [75] algorithm to extract appropriately sized malicious and benign provenance subgraphs. Notably, benign subgraphs consist entirely of benign system entities, while malicious subgraphs depict attack processes with a mix of benign and malicious entities. In our modified Louvain algorithm, some important “hub” nodes will be allowed to appear redundantly across various provenance subgraphs to ensure event completeness.

We extract N provenance subgraphs $\{G_1, G_2, G_3, \dots, G_N\}$, where each subgraph G_k corresponds to a ground truth label $Y_k \in \{0, 1\}$. Here, 0 indicates that all system entities within the subgraph are benign (i.e., the subgraph is benign), while 1 indicates the presence of malicious system entities (i.e., the subgraph contains at least one stage of an APT attack).

(3) Provenance Detection. The goal of provenance detection is to predict a label for each extracted subgraph G_k , indicating whether this local system behavior is APT-related. By inputting the provenance subgraphs into GNN models, we learn a mapping function $f(\cdot)$ to get the predicted probability $P(c|G_k)$. The final result label \hat{Y}_K for G_k is determined by choosing the highest probability:

$$\hat{Y}_k = \arg \max_{c \in \{0,1\}} P(c|G_k). \quad (2)$$

Graph Neural Networks: For model selection, we will adopt standard GNN model frameworks widely used in the industry (including GCN, GAT and GraphSAGE)². It should be emphasized that the security model used for attack detection is only an auxiliary tool to help us screen and locate the source subgraphs containing suspicious activities (i.e. potential attacks) and determine the predicted flips. Readers can replace any other GNN-based detection model; the in-depth explanation of the decision-making mechanism of the trained security model is the core and innovation of our research.

V. DESIGN OF PROVX

In this section, we propose PROVX, a counterfactual explanation framework for provenance graphs, which comprises the following three components:

① Provenance Subgraph Perturbation(V-A). To perform structural changes on the provenance subgraph, we represent perturbations to the provenance graph by learning adjustable edge masks, thereby transforming the discrete graph edit search problem for counterfactual changes into a continuous mask learning task.

② Counterfactual Reasoning(V-B). We seek a minimal graph perturbation scheme that can explicitly flip the security model’s original judgment of an APT threat. PROVX constructs a dual optimization objective framework comprising a

²It is worth noting that most of the custom models in these works [17], [18], [21], [24], [26], [50] are also based on these basic GNN architectures.

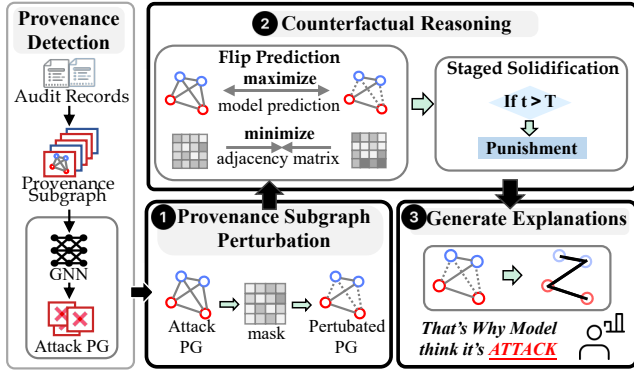


Fig. 4. **PROVX working mode.** Provenance detection is the upstream task of PROVX. PROVX explains the detected malicious provenance subgraphs.

threat misclassification guidance loss and an attack path minimal perturbation loss, and introduces a staged solidification strategy to focus on core attack elements.

③ **Generate Explanations(V-C).** After the above optimization learning process, we sort the final edge masks according to their importance and transform them into independent counterfactual subgraphs to obtain a complete explanation.

A. Provenance Graph Perturbation

Since APT attacks often involve meticulously planned event sequences and concealed dependencies, PROVX explores variations in both the provenance graph structure (i.e., dependencies and correlations between system events) and entity/behavioral features, its objective is to alter the edge structure of the original provenance subgraph to simulate various obfuscation and evasion strategies that attackers might employ. We focus on perturbing system entity interactions. For a provenance subgraph G_k , its adjacency matrix is $A_k \in \{0, 1\}^{n \times n}$, where 0 represents the absence of an entity pair and 1 represents its presence.

PROVX employs an edge masking method to derive a perturbed graph \tilde{G}_k by masking edges of the original provenance subgraph G_k . Specifically, we achieve graph structure perturbation by learning a differentiable, continuous-valued edge mask parameter matrix $M_k \in \mathbb{R}^{n \times n}$ with the same dimensions as A_k (where n is the number of nodes in the provenance graph). The perturbed adjacency matrix \tilde{A}_k is formulated as follows:

$$\tilde{A}_k = A_k \odot M_k, \quad (3)$$

where $M_k \in \{0, 1\}^{n \times n}$ is a binary edge mask matrix, and \odot denotes element-wise multiplication. If an element $M_{k,ij}$, it indicates that the edge (i, j) is masked in A_k . Considering that directly learning the binary edge mask matrix M_k is non-differentiable, we relax M_k to continuous real values, i.e., $\hat{M}_k \in \mathbb{R}^{n \times n}$, then,

$$\tilde{A}_k = A_k \odot \sigma(\hat{M}_k), \quad (4)$$

where $\sigma(\cdot)$ denotes the sigmoid function, which maps the edge mask to the range $[0, 1]$, enabling a smooth transition between

the presence and absence of edges. Consequently, starting from a randomly initialized edge mask, \hat{M}_k can be optimized via gradient descent.

B. Counterfactual Reasoning

We conduct counterfactual reasoning and learn masks for the provenance subgraphs predicted by the security model as malicious [66]. The core idea of our counterfactual reasoning is to determine the minimum perturbation of the provenance subgraph, which is achieved by establishing an optimization task and optimizing the composite loss function. It is worth mentioning that our counterfactual optimization task only focuses on the learning of edge masks and does not involve the learning of the GNN security model. The parameters in the GNN security model are fixed.

The specific workflow is as follows: We perturb the target provenance subgraph as described in section V-A to generate a perturbed subgraph \tilde{G}_k , and input it together with the original subgraph G_k into the well-trained GNN-based security model we built in the previous article to generate the corresponding prediction results, which are:

$$\hat{Y}_k = \text{GNN}(A_k), \quad \tilde{Y}_k = \text{GNN}(\tilde{A}_k). \quad (5)$$

We learn the edge mask \hat{M}_k based on the optimization objective of the counterfactual reasoning problem to determine the minimal counterfactual perturbation. The mask learning process is achieved by minimizing a composite loss function $\mathcal{L}_{\text{PROVX}}$ over T training epochs. This loss function simulates the trade-off between attempting to change the model's prediction (akin to an attacker evading detection) and minimizing modifications to the original activity logs (maintaining provenance fidelity and the minimal perturbation principle). It consists of a primary counterfactual loss \mathcal{L}_{CFX} and a staged solidification loss \mathcal{L}_S .

1) *Primary Counterfactual Loss:* The guiding principle of \mathcal{L}_{CFX} is to find modifications to the provenance graph that can effectively change the APT detection result while keeping the perturbation as small as possible. Thus, \mathcal{L}_{CFX} is made up of two loss terms:

Prediction Flip Loss. The prediction flip loss $\mathcal{L}_{\text{Pred}}$ drives the model to find a modification to the provenance graph such that the modified graph is no longer classified under the original threat category (i.e., is no longer considered an APT attack behavior). We achieve this by minimizing the probability corresponding to the original threat prediction \hat{Y}_k , which encourages the model to significantly reduce its confidence in the original threat assessment when faced with tampered attack evidence chains:

$$\mathcal{L}_{\text{Pred}} = P(\hat{Y}_k, \tilde{A}_k). \quad (6)$$

Mask Distance Loss. The mask distance loss $\mathcal{L}_{\text{dist}}$ aims to ensure that the identified provenance graph modification, which can change the model's judgment, is as small as possible. This means identifying only those most critical, core activity segments or system interactions that an attacker would most want to hide or exploit. If a masking scheme requires

modifying a large number of event correlations to make the GNN model change its prediction, then this loss term will be larger:

$$\mathcal{L}_{\text{dist}} = \text{BinaryCrossEntropy}(\sigma(\hat{M}_k), A_k). \quad (7)$$

Get Primary Counterfactual Loss. We integrate the above two loss terms into the primary counterfactual loss function to optimize them collaboratively:

$$\mathcal{L}_{\text{CFX}} = \alpha \cdot \mathcal{L}_{\text{pred}} + (1 - \alpha) \cdot \mathcal{L}_{\text{dist}}, \quad (8)$$

where α is responsible for regulating the trade-off between the prediction loss term and the distance loss term. A higher α prioritizes changes to counterfactual predictions at the expense of larger perturbations, while a lower α focuses more on minimizing graph modifications.

2) *Staged Solidification*: To provide security analysts with clearer and more reliable insights into critical attack paths and potential vulnerabilities, PROVX introduces a staged solidification mechanism. This mechanism is controlled by a solidification factor γ_S , a solidification stage starting ratio $R_S \in [0, 1]$, and confidence thresholds τ_{low} and τ_{high} . Its core idea is to further reinforce the judgment on event correlations that have already been initially identified as extremely important or unimportant for prediction flipping in the later stages of explanation model training, making their contributions more prominent.

The solidification process is activated after an initial exploratory learning phase, specifically starting from the T_{start} -th epoch, and its strength is controlled by the solidification factor γ_S .

Mask Snapshot and Confident Edge Identification. At the T_{start} epoch, the currently activated edge mask (specifically for $\sigma(\hat{M}_k)$ in the paper, denoted as $M_{\text{snap}}^k = \sigma(\hat{M}_k^{(T_{\text{start}})})$ is recorded. Then, based on predefined low confidence threshold τ_{low} (confident threshold low) and high confidence threshold τ_{high} (confident threshold high), two sets of edge indices are identified:

- For low confidence edge index set $I_{\text{low}} = \{i | m_{\text{snap},i}^k < \tau_{\text{low}}\}$ (edges whose mask values are already close to 0 in the snapshot),
- For high confidence edge index set $I_{\text{high}} = \{j | m_{\text{snap},j}^k > \tau_{\text{high}}\}$ (edges whose mask values are already close to 1 in the snapshot).

Solidification Penalty Application. For epochs $t > T_{\text{start}}$, the solidification penalty \mathcal{L}_S is calculated as follows:

$$\mathcal{L}_S = \sum_{i \in I_{\text{low}}} (\sigma(\hat{M}_{k,i}^{(t)}) - m_{\text{snap},i}^k)^2 + \sum_{j \in I_{\text{high}}} (\sigma(\hat{M}_{k,j}^{(t)}) - m_{\text{snap},j}^k)^2. \quad (9)$$

This penalty term targets edges whose mask values already showed a clear trend (approaching 0 or 1) at T_{start} moment. If these edges' mask values deviate from their snapshot state during subsequent training, they are penalized. This encourages the mask values to consolidate towards 0 or 1.

3) *Final Loss*: The total loss function $\mathcal{L}_{\text{PROVX}}$ is the sum of the counterfactual loss and the weighted solidification loss:

$$\mathcal{L}_{\text{PROVX}} = \mathcal{L}_{\text{CFX}} + \gamma_S \cdot \mathcal{L}_S, \quad (10)$$

where the γ_S parameter is used to fix the weight of the solidification loss.

C. Generating Counterfactual Explanations

In this phase, we generate intuitive security explanations to answer for security personnel the predictions of the APT detection model. After implementing counterfactual reasoning, we will obtain a final set of event correlation masks \tilde{M}_k^* . By subtracting the mask values from 1 value, we can obtain the contribution of each event correlation to flipping the GNN's original APT judgment. High-contribution event correlations indicate that their corresponding edges should be removed; these represent the critical behavioral links in the provenance graph that, if successfully hidden, tampered with, or missed by an attacker, are most likely to cause the GNN detection model to miss or misclassify an APT. Low-contribution event correlations indicate their corresponding edges should be retained. We select the top- K highest-contribution edges, where K is a hyperparameter used to control the conciseness of the counterfactual explanation. To highlight the impact of structural properties on model predictions, we randomize the nodes associated with these edges to avoid diluting the influence of edges due to overly dense node interactions. Our objective is to explain the change in the prediction by modifying as few edges as possible. Finally, we extract the subgraph formed by these K removed edges. This subgraph, \tilde{G}_k^* , is the core of the counterfactual explanation, stating: **"If these K edges were removed from the original graph, then this provenance graph would not be predicted as an APT attack."** Such explanations will reveal the predictive mechanisms of the security model and assist security analysts in understanding the warnings output by PIDS.

VI. EXPLAINATION TO DETECTION FEEDBACK

Although PIDS are widely deployed, their robustness against adaptive adversaries remains to be proven. When a malicious activity successfully deceives a PIDS through camouflage or obfuscation [47], causing it to be misclassified as benign, we refer to it as a successful adversarial evasion. This type of adversarial attack modify attack patterns to make them behaviorally indistinguishable from benign processes.

The core capability of PROVX is to answer the question, "Why is a certain behavior classified as malicious?". However, its counterfactual framework can also be extended to answer a more challenging question: "Why is a malicious behavior able to successfully evade detection?". Therefore, we explore a closed-loop enhancement framework guided by security analysts and based on explanation feedback. By utilizing PROVX as a forensic analyzer for adversarial behaviors, we aim to explain the reasons for adversarial evasion and feed the analysis results back to the detector to enhance its adversarial

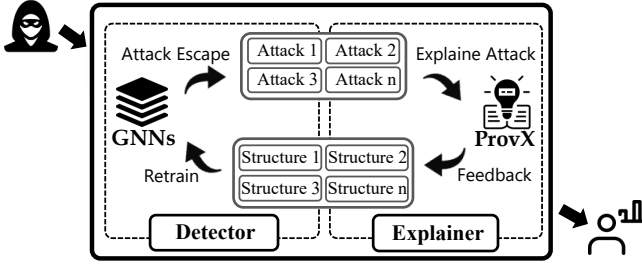


Fig. 5. Feedback Loop between Explainers and Detectors.

robustness. Unlike existing PIDSs that integrate adaption module [17], [18], [21], our framework actively learns adversarial patterns that induce false negatives, rather than focusing on passively correcting false positives.

The framework’s design is formalized in Fig.5. When an adversarial sample G_{adv} evades the current security model $f(\cdot)$, PROVX is tasked with a new objective: to find the minimal structural perturbation, G_{exp} , that would flip the model’s prediction from “benign” back to “malicious”. This generated explanation, G_{exp} , represents the benign structural elements used by the adversary to evade detection. This explanation is then presented to a security analyst for verification, then, security analyst will remove G_{exp} from G_{adv} and retain the remaining subgraph structure G_{left} for verification. This part of the structure contains the malicious structure determined by the model and mix with some benign noise. Upon confirmation, the G_{adv} and G_{left} —along with their now-correct malicious label—is added to a new set of training samples together. This allows us to retrain two different adversarial behaviors with more abundant data.

One noteworthy point is that our framework primarily mitigates the risk of increasing false positives through analyst verification, a step that influences the model to filter out ambiguous or low-confidence explanations. Furthermore, fine-tuning on the full graph structure, rather than completely re-training on a dismantled subgraph, also helps reduce false positives while simultaneously feeding back adversarial strategies to reduce false negatives. The effectiveness of this feedback mechanism is quantitatively evaluated in Section VII-F.

VII. EVALUATION

In this section, we evaluate the effectiveness of PROVX in explaining attacks. Our goal is to answer the following research questions:

- **RQ1.** What is the relevance of the explanations generated by ProvX to real-world attacks? **(VII-B)**
- **RQ2.** To what extent can PROVX explain the security model’s predictions? **(VII-C)**
- **RQ3.** What is the impact of hyperparameters on the explanation performance of PROVX? **(VII-D)**
- **RQ4.** What is the overhead of PROVX? **(VII-E)**
- **RQ5.** How does PROVX as an explainer help detect adversarial threats? **(VII-F)**

A. Experimental Setup

1) **Datasets:** Consistent with existing APT detection works, we conduct our experimental evaluation on two widely used APT datasets: the DARPA TC (Transparent Computing program) and the DARPA OpTC (Operationally Transparent Cyber)³. The DARPA TC dataset [70] was collected during adversarial engagements in a realistic environment. In these scenarios, a red team employed various exploits and attack techniques to compromise target hosts and obtain sensitive information, while a blue team worked to detect attacks and collect data on suspicious behavior through methods like host-level auditing and causal analysis. DARPA TC targets specific hosts for attack, and for our research, just like [18], we select the **Cadets**, **Theia** and **Trace** scenarios. The DARPA OpTC dataset [71] provides a broad view of benign and malicious audit records from approximately 1,000 hosts. A key reason for choosing these two datasets over others is their enhanced realism and credibility, owing to their high-quality ground truth. To improve training quality, we perform random negative sampling on the scarce malicious behavior subgraphs to obtain a balanced dataset. We split the data into training, validation, and test sets at a ratio of 7:1:2. Please note that the explainer provides explanations only for the detection model’s predictions on the test set.

TABLE II
DARPA OpTC DETECTION PERFORMANCE ON THREE GNNs.

GNN type	Acc	Pr	Rec	F ₁	AUC	FPR ¹
GCN	0.7379	0.5757	0.6340	0.6034	0.8006	0.2144
GAT	0.9126	0.8464	0.8824	0.8640	0.9670	0.0735
GraphSAGE	0.9404	0.9276	0.8791	0.9027	0.9784	0.0315

¹ FPR (False Positive Rate), a detection system with a lower false positive rate can better avoid false positive attacks during detection.

TABLE III
DARPA TC E3 DETECTION PERFORMANCE ON THREE GNNs.

GNN	Datasets	Acc	Pr	Rec	F ₁	AUC	FPR
GCN	Cadets	0.9972	0.9999	0.9286	0.9630	0.9938	0.0001
	Theia	0.9612	0.8491	0.7031	0.7692	0.8947	0.0127
	Trace	0.9968	0.9778	0.9635	0.9706	0.9915	0.0013
GAT	Cadets	0.9944	1.0000	0.8571	0.9231	0.9647	0.0001
	Theia	0.9554	0.7705	0.7344	0.7520	0.9209	0.0222
	Trace	0.9968	0.9850	0.9562	0.9704	0.9918	0.0009
Graph SAGE	Cadets	0.9986	0.9999	0.9643	0.9818	0.9931	0.0001
	Theia	0.9885	0.9828	0.8906	0.9344	0.9617	0.0016
	Trace	0.9955	0.9701	0.9489	0.9594	0.9907	0.0017

2) **Implementation Details:** Here, we detail the implementation of our two main components:

GNN-based APT Attack Detection Model. We implemented subgraph-level attack detection models based on GCN, GAT, and GraphSAGE. In our design, each model is configured

³Existing PIDS based on provenance graphs all use one [21] or more [17], [20] of these datasets to verify APT detection performance. Our work refers to [18], [26], [50] and selects the above datasets.

with a 2-layer GNN architecture. Each GNN layer is followed by a ReLU activation function and a Dropout layer. Then an average pooling layer is applied to obtain the graph-level representation and input it into the classifier. We use a simple 2-layer MLP (Multi-layer Perceptron) as the classifier, and the final output of the model is a prediction score that distinguishes benign and malicious subgraphs. We use the Adam optimizer for optimization, with the number of iterations set to 50 and the learning rate to 0.001. The test performance of the security model on all datasets and GNNs is listed in Table II and Table III, respectively. The results show that the three datasets of DARPA TC E3 perform well in all indicators, showing considerable attack sensitivity and extremely low false alarm rate. For DARPA OpTC, GAT performs second best and GCN performs the weakest, indicating that the complexity of OpTC does make it difficult to classify directly as in the E3 dataset. The above detection model provides a solid foundation for the interpreter to locate malicious subgraphs and make accurate prediction flip judgments.

Counterfactual Explainer. For each detected provenance subgraph, PROVX conducts the explanation training process independently. We set the learning rate to 0.01 and the number of epochs to 200. The solidification ratio is set to 60% of the total epochs, solidification factor is set to 0.5, and the lower and upper confidence bounds are set to 0.05 and 0.95, respectively. Additionally, we conducted hyperparameter experiments on the counterfactual parameter α within \mathcal{L}_{CFX} and the solidification penalty strength γ_S within \mathcal{L}_S to analyze their impact on PROVX's explanation results. Furthermore, during the explanation generation phase, we also discuss the selection of the number of edges, K , for the explanation. Details can be found in Section VII-D.

3) *Evaluation Metrics:* In the following evaluation of explanation performance, we define two core tasks to assess the quality of an explainer from two aspects, we establish corresponding metrics for each:

Task1. Explanation of hit degree. We first explore whether the counterfactual explanations generated by the explainer are related to real attacks. We use the collection of malicious labels from existing works [17], [18], [26] as the ground truth for evaluation. After the explainer is trained, we generate explanations for the target subgraphs. We use Accuracy, Precision, Recall and F1-score as evaluation metrics for explanation hit accuracy. By comparing the explanation results with the ground truth, we quantitatively assess how accurately the generated explanations pinpoint key information related to the attack. Specifically, if an edge within the Top- K explanation provided by the explainer corresponds to a real attack, we consider it a 'Hit'. We define Accuracy as the proportion of malicious graphs in the test set for which the explainer achieves at least one such hit. We then calculate Precision and Recall for each subgraph by comparing the explanation results with the ground truth, and finally average these scores over all subgraphs. Precision verifies what proportion of the key events found by PROVX are real key attack steps. Recall verifies what proportion of all real key attack steps are successfully found

by PROVX. The final F1 score is obtained by dividing the F1 score of each subgraph by the number of malignant subgraph detected.

Task2. Explanation of necessity. We investigate at the model level whether the counterfactual explanations generated by the explainer truly influence the GNN security model's prediction for an APT attack. This is because the structures that influence model predictions are not necessarily all attacking structures; some benign structures also play an important role in the explanation and analysis. They are the key to reversing benign and malignant predictions. Unlike in [68], we do not use *Fidelity+*/*Fidelity-* to measure the explainer's performance, because we do not need surrogate models for indirect explanations; instead, we directly explain the graph structure. Inspired by research in other fields [66], [76], [77], we introduce concepts from causal inference and use the **Probability of Necessity (PN)** [78] to evaluate the necessity of an explanation. In simple terms, it asks: If the explanation E did not occur, would the prediction P still occur? In our scenario, this translates to: For a specific graph structure \tilde{G}_k^* within an attack graph, if it were removed, would the GNN model still classify the graph as an APT? PN is defined as follows:

$$PN = \frac{1}{N} \sum_{k=1}^N p_k, \quad p_k = \begin{cases} 1, & \text{if } \hat{Y}'_k \neq \hat{Y}_k, \\ 0, & \text{else.} \end{cases} \quad (11)$$

Where \hat{Y}_k is the original prediction, \hat{Y}'_k represents the prediction result of the security model after removing the explanation subgraph \tilde{G}_k^* from the original traceability graph G_k .

4) *Explainers for Comparison:* To the best of our knowledge, there are very few works on provenance-based explanations. Since these works are not open source, we are unable to reproduce the comparative experiments [68], [69]. Therefore, we use other SOTA fact-based explainers with similar ideas to these works for comparison (these SOTA explainers are also used for comparative experiments in [68], [69]). These include: (1) perturbation-based methods such as GNNExplainer [58], SubgraphX [60], and PGExplainer [59]; (2) decomposition-based methods like GNN-LRP [61] and DeepLIFT [62]; and (3) gradient-based methods like GradCam [63]. The detailed comparison can be found in Section VII-B and Section VII-C.

B. Explanation Relevance to Ground-Truth Attacks (RQ1)

A critical path found by an explainer might mathematically flip a model's prediction, but it may not correspond to the actual root cause of an attack in a real-world cybersecurity context. Therefore, a major focus of our work is to investigate whether the important interaction structures identified by the explainer can match the real attack paths manually annotated by domain experts. We adopt the metrics mentioned in VII-A3 to conduct experiments and comparisons between ProvX and the baselines. TABLE IV presents the evaluation results of ProvX across all datasets and GNN models, while TABLE V shows the comparison between ProvX and the baseline explainers.

TABLE IV
EXPLAINING ATTACK CORRELATION PERFORMANCE OF DARPA OpTC
AND TC-E3 ON GCN, GAT, AND GraphSAGE, WITH $K=10$.

GNN Type	Metrics ¹	DARPA OpTC	DARPA TC E3		
			Cadets	Theia	Trace
GCN	Acc	0.8402	0.9615	1.0000	1.0000
	Pr	0.1009	0.9290	0.8803	1.0000
	Rec	0.7573	0.1198	0.1077	0.0990
	F1	0.1683	0.1904	0.1821	0.1802
GAT	Acc	0.9148	0.9630	0.9778	1.0000
	Pr	0.1293	0.9021	0.9124	1.0000
	Rec	0.8169	0.1616	0.1051	0.1089
	F1	0.2118	0.1959	0.1884	0.1287
GraphSAGE	Acc	0.9033	0.9988	1.0000	1.0000
	Pr	0.1306	1.0000	0.9491	1.0000
	Rec	0.7035	0.1089	0.1387	0.1287
	F1	0.2006	0.1964	0.2380	0.2281

¹ PROVX uses different security model, so the number of malicious subgraphs captured is different.

TABLE V
COMPARISON OF PROVX AND BASELINES' PERFORMANCE IN
EXPLAINING ATTACK CORRELATION ON THREE GNNs, IN DARPA OpTC.

GNNs	Explainers ¹	Acc	Pr	Rec	F1
GCN	GradCam	0.2165▼74.2%	0.0944▼6.4%	0.2473▼67.3%	0.1418▼25.5%
	DeepLIFT	0.2639▼68.6%	0.0355▼64.8%	0.1832▼75.8%	0.0730▼61.7%
	GNN-LRP	0.2526▼69.9%	0.0891▼11.7%	0.1774▼76.6%	0.1186▼37.7%
	PGExplainer	0.7938▼5.5%	0.0978▼3.1%	0.7147▼5.6%	0.1426▼25.1%
	SubgraphX	0.7783▼7.4%	0.1019▲1.0%	0.6413▼15.3%	0.1011▼46.9%
	GNNExplainer	0.84020.0%	0.1025▲1.6%	0.7492▼1.1%	0.1698▼10.8%
	PROVX	0.8402▲60.3% ²	0.1009▲16.2%	0.7573▲67.5%	0.1904▲53.0%
GAT	GradCam	0.6570▼28.2%	0.1173▼9.0%	0.6867▼15.9%	0.1894▼10.6%
	DeepLIFT	0.7956▼13.0%	0.1044▼19.3%	0.7039▼13.5%	0.1948▼8.0%
	GNN-LRP	0.8540▼6.6%	0.1151▼10.7%	0.7666▼6.2%	0.1974▼6.8%
	PGExplainer	0.8905▼1.6%	0.1188▼8.1%	0.8007▼1.9%	0.2021▼5.1%
	SubgraphX	0.9015▼1.4%	0.1162▼10.6%	0.7938▼3.6%	0.1950▼8.0%
	GNNExplainer	0.8759▼4.2%	0.1130▼12.1%	0.7848▼2.2%	0.1977▼6.7%
	PROVX	0.9148▲10.3%	0.1293▲13.3%	0.8169▲8.0%	0.2118▲8.0%
GraphSAGE	GradCam	0.7434▼17.7%	0.0784▼39.8%	0.6202▼11.9%	0.1311▼34.6%
	DeepLIFT	0.7199▼20.2%	0.0727▼43.9%	0.5925▼15.8%	0.1230▼38.7%
	GNN-LRP	0.6841▼24.3%	0.0877▼32.8%	0.5800▼17.6%	0.1206▼40.0%
	PGExplainer	0.8922▼1.2%	0.1233▼5.6%	0.6748▼4.1%	0.1779▼11.3%
	SubgraphX	0.90330.0%	0.1298▼0.6%	0.7124▲1.3%	0.2147▼7.0%
	GNNExplainer	0.9108▲0.8%	0.1247▼4.5%	0.7002▼0.5%	0.1994▼0.6%
	PROVX	0.9033▲11.7%	0.1306▲27.1%	0.7035▲8.8%	0.2006▲24.5%

¹ Bold denotes the best results, and underline denotes the second-best results.
▼% represents the percentage of reduction, ▲% represents the percentage of improvement.

² We adopt the hyperparameter configuration of previous work [65] to set up the baselines explainers.

³ Improvement compared to the average of all detection systems except PROVX.

The results show that ProvX achieves high Accuracy across all datasets, with an average Accuracy of 0.8861, demonstrating that at least a portion of the edges considered important by ProvX overlaps with real attacks. Influenced by the inherent distribution of the datasets, the results show a data distribution where Recall is particularly outstanding on DARPA OpTC, while Precision is more prominent on DARPA TC. This is a very interesting phenomenon. We found that the prevalence of star structures (such as those related to Nginx) in the DARPA TC dataset significantly affects the learning trend of

the detection model. The K key edges found by PROVX cannot fully cover the numerous interactions of the hub nodes. The attack distribution in OpTC is more scattered and complex. The interpreter can obtain multiple important links of the scattered attack links, but these important edges may be crucial to model prediction, rather than the real steps annotated by experts. The above performance is also partly due to the limitation of the subgraph partitioning method. Interested readers can view our subgraph partitioning design in the Appendix.

In the comparative experiments, PROVX outperforms the baseline explainers in the vast majority of cases, demonstrating the high efficiency of the counterfactual explanation approach in the APT explanation domain. Among the baselines, the perturbation-based methods (GNNExplainer, PGExplainer, and SubgraphX) perform slightly better than the decomposition-based methods (GNN-LRP and DeepLIFT-Graph), while the gradient-based method, Gradcam, performs the worst. The experimental results indicate that perturbing the subgraph structure can effectively highlight important structures in APT datasets. In contrast, due to the generally high confidence of the model's predictions, decomposition-based methods struggle to create a significant gap in edge importance scores within the provenance subgraphs. Meanwhile, the gradient-based approach of using gradients to analyze edge importance tends to explain hidden correlations rather than actual attacks amidst the dense interactions of provenance graphs.

TABLE VI
EXPLANATION NECESSITY PERFORMANCE OF DARPA OpTC AND
TC-E3 ON GNNs, WITH $K=10$.

GNN Type	DARPA OpTC	DARPA TC E3		
		Cadets	Theia	Trace
GCN	0.1856	0.3943	0.3197	0.3515
GAT	0.5926	0.4534	0.4356	0.4679
GraphSAGE	0.7695	0.4877	0.4556	0.5751

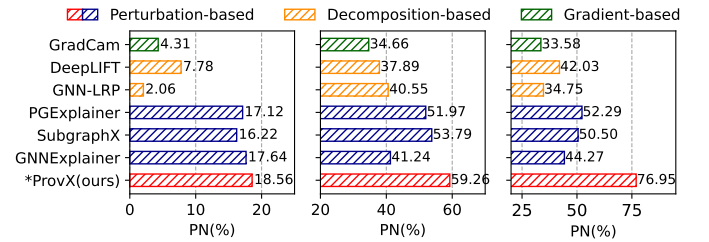


Fig. 6. PROVX's PN(%) performance on DARPA OpTC, from left to right are the results of GCN, GAT, and GraphSAGE. PROVX is also a perturbation-based scheme, but is the only counterfactual-based explainer among these works.

C. Analysis of Explanation Necessity (RQ2)

Unlike RQ1, which investigates whether the generated explanations hit real-world attacks, RQ2 analyzes the necessity of the explanations. It examines whether the explanations

generated by PROVX genuinely influence the security model's predictions, thereby assisting security analysts in understanding the model's decision-making mechanism. A high PN score indicates that the explanation found by PROVX is indeed critical to the model's decision, and removing it can effectively change the model's prediction. This suggests that the explanation has high fidelity and causal validity. Conversely, a low PN score implies that the so-called critical path found by the explainer is not the true reason for the model's decision, as the model still considers the graph malicious after its removal. TABLE VI presents the evaluation results of PROVX on all datasets, while Fig. 6 shows a comparison between PROVX and the baselines using the OpTC dataset as an example. We can see that, with $K=10$ as the baseline parameter, PROVX achieves a PN result greater than 40% on all datasets, with an average PN result of 51.59%. In other words, on the test set, the set of important edges explained by PROVX does indeed subvert the security model's prediction upon removal, leading to a completely opposite judgment.

We observe the comparison with the baselines. The purpose of this comparison is to study the relative merits of the counterfactual explanation mechanism represented by PROVX and the factual explanation mechanisms represented by the other baselines. The results show that PROVX's explanations for the three GNN-based models outperform all the state-of-the-art fact-based explainers. This is because PROVX's explanation approach seeks the minimal change to a provenance subgraph that alters the prediction. This method, distinct from fact-based approaches, can guarantee the identification of the truly necessary edges that influence the security model's prediction.

Among the various fact-based baselines, their different explanation logics lead to different results in the PN evaluation. Specifically, the optimization objective of perturbation-based methods can most intuitively lock important decisions, while the local nature of gradient-based methods makes their connection to the necessity of a global decision the weakest. Therefore, we can observe a systematic performance gap among these three types of methods: the perturbation-based explainers achieve relatively high PN values, followed by the decomposition-based methods, while the gradient-based method has the lowest PN value. However, in this evaluation, the settings of several hyperparameters can also significantly affect the PN assessment, which we will discuss in subsequent sections.

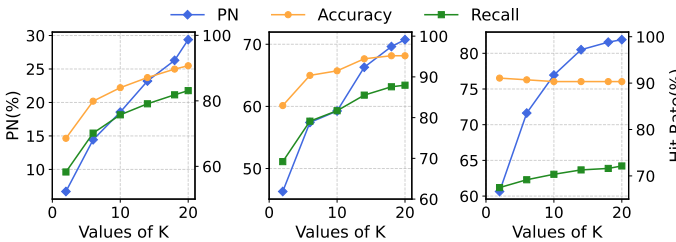


Fig. 7. PROVX's PN(%) and Hit rate(%) changes when the K changes, from left to right are the results of GCN, GAT and GraphSAGE.

D. Hyperparameter Investigation (RQ3)

The setting of hyperparameters is crucial for the performance of machine learning-based models. We investigate the impact of several different hyperparameters on the explanation performance of ProvX:

Explanation Size K : K directly determines the number of edges from the explainer that are included in the evaluation, making it one of the most direct hyperparameters affecting the assessed performance. We evaluate K on the OpTC, varying its value from 1 to 20. The results are shown in Fig. 7. Increasing K directly improves the PN value. On GraphSAGE, when K is 20, the PN value can even reach 81.91%. This demonstrates that if the 20 evaluated edges are removed from each target graph (malicious subgraph), the prediction for 81.91% of these subgraphs will flip directly from 'Attack' to 'Benign'. However, K should not be excessively large, as this would diminish the significance of the evaluation; it is not surprising that removing a large portion of a graph's edges would cause its prediction to flip. Furthermore, across the three security models, increasing K also increases the Accuracy and Recall of the ground truth evaluation. It is worth noting that Precision, in most cases, shows an opposite trend to Recall. This is easy to understand: the number of true positive edges is capped by the ground truth and does not change as K increases, but the denominator in the Precision calculation (the total number of selected edges, K) becomes larger.

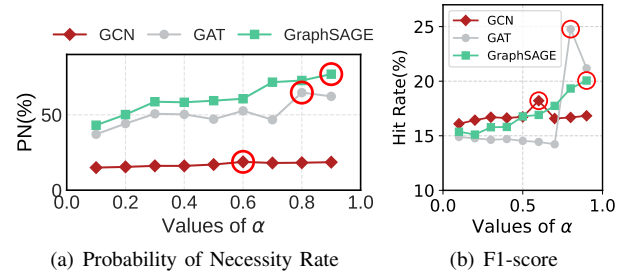


Fig. 8. PN and F_1 -score changes of PROVX when the α changes.

Counterfactual Trade-off Parameter α : The parameter α is responsible for balancing the Prediction Flip Loss $\mathcal{L}_{\text{pred}}$ and the Mask Distance Loss $\mathcal{L}_{\text{dist}}$. The former aims to change the model's prediction result, while the latter aims to minimize the modifications to the original graph. The experimental results are shown in Fig. 8. Fig. 8(a) displays the change in PN with respect to α , and Fig. 8(b) shows the change in F1-score with respect to α . For ProvX's explanations on the three security models based on GCN, GAT, and GraphSAGE, the optimal values for α are 0.6, 0.8, and 0.9, respectively. This is validated by the performance in terms of both PN and F1-score. It can be observed that when α exceeds the optimal point, both experimental metrics begin to decline. This demonstrates that the explainer, by trying too hard to generate perturbations to obtain the opposite prediction, disrupts the balance between flipping the model's prediction and minimizing the perturbation, thereby failing to achieve the flip at a minimal cost.

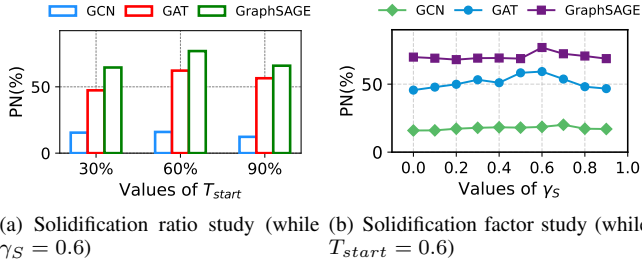


Fig. 9. PN changes of PROVX when the T_{start} and γ_S changes.

TABLE VII

THE OVERHEAD OF PROVX PERFORMING A COMPLETE COUNTERFACTUAL EXPLANATION AND THE DETECTION OVERHEAD ARE ALSO RECORDED.

Phase	Time consumption (s)			Peak Memory consumption (MB)		
	GCN	GAT	Graph SAGE	GCN	GAT	Graph SAGE
Detection Training	113	174	140	1377	1405	1379
Explanation	Training	733	1196	994	1491	1519
	Evaluating	3	4	4	1541	1561

Solidification Start Ratio T_{start} & Penalty Strength γ_S : Solidifying explanations is one of the core features of PROVX. The solidification ratio determines when the training transitions into the strict solidification phase, activating this mechanism only after a specific proportion of the training is complete. Once in the solidification phase, the solidification strength determines the magnitude of the penalty applied to explanations that attempt to deviate from established, clear judgments (i.e., where mask values are already close to 0 or 1). Fig. 9 shows the analysis of T_{start} and γ_S . Fig. 9(a) shows the trend of PN as T_{start} varies, with γ_S fixed at 0.6. It can be seen that when T_{start} is controlled at 0.6, entering the solidification phase at this point allows the explainer to retain sufficient training epochs to hypothesize counterfactual possibilities. A value of T_{start} that is too low leads to the premature arrival of penalties, while a value that is too high causes the counterfactual hypotheses to become too divergent, leading the model to favor some overly exaggerated perturbation patterns. Fig. 9(b) shows the trend of PN as γ_S varies, with T_{start} fixed at 0.6. It is observable that when γ_S is 0 (i.e., no penalty), the explainer effectively skips the solidification phase, which leads to a significant drop in performance. The best performance is achieved when γ_S is 0.7, 0.6, and 0.6 for GCN, GAT, and GraphSAGE, respectively.

E. System Overhead (RQ4)

PROVX is a security model explainer, and its complete workflow includes training the detection model, training the explainer, and evaluating the explanation results. We test the time and memory overhead for three types of GNN cores. Notably, considering that most real-world threat analysis scenarios lack GPU support, we conduct our evaluation in a CPU environment.

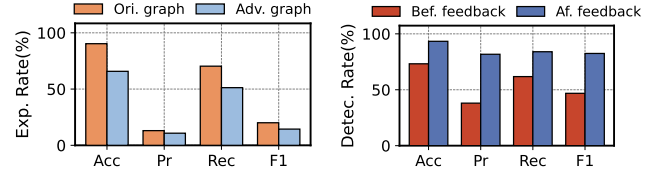


Fig. 10. PROVX finds the key structures that evade model detection and feeds back to the security model to enhance adversarial robustness.

The experimental results are shown in TABLE VII. As can be seen, training the detector does not incur excessive overhead. For the explainer training, even the longest case (using GAT) requires less than 20 minutes, and the average peak memory usage is only 1.43 GB, which can be easily handled by a standard host machine. Furthermore, the final evaluation of the explanation can be completed in just 3-4 seconds. Overall, PROVX demonstrates excellent performance in terms of time and memory overhead. Its low hardware requirements significantly reduce the barrier and cost for enterprises to interpret APT, allowing for easy integration into existing standard server architectures. Security teams can leverage it to rapidly iterate on explanation models based on newly emerging attack patterns.

F. Explanation Feedback for Adversarial Attacks (RQ5)

In VI, we investigate security analyst-guided explanation feedback for adversarial attacks, utilizing the counterfactual explainer PROVX as a forensic analyzer for adversarial behaviors. In this chapter, we design adversarial experiments to verify the actual performance of the adversarial attack Feedback Loop based on PROVX.

Specifically, our experimental design is as follows:

- Based on existing malicious provenance subgraphs, we add benign structures according to the method in [47] to create adversarial samples capable of evading PIDS detection. We then input these samples into PROVX to analyze its explanation.
- The counterfactual objective is set to instruct PROVX to find modifications that flip the prediction of the adversarial example. This portion of the explanation is the attacker-manipulated obfuscated structure.
- We retain the adversarial subgraph structures outside the explanation structure as new training examples. These examples and original adversarial attack are manually fed back into PIDS for retraining to enhance its ability to learn against adversarial behavior.

We will record the explanation performance of PROVX before and after the addition of adversarial behaviors, as well as the detection performance of the PIDS on adversarial behaviors before and after the explanation feedback, as shown in Fig. 10(a) and Fig. 10(b).

As can be seen, adding adversarial structures to the same attack graph leads to a decrease in explanation performance.

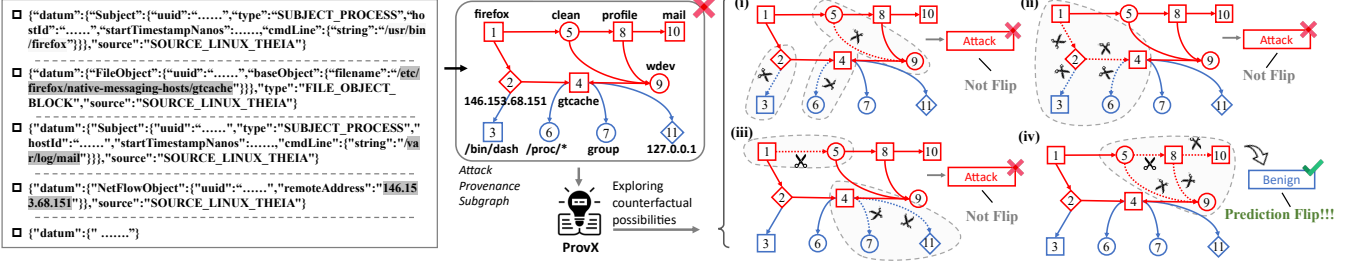


Fig. 11. An attack case from the DARPA E3 Theia dataset. We construct a provenance graph from the audit log, which includes processes, files, and sockets. We input the graph that the model identifies as an attack into PROVX for interpretation. The figure shows four possible ways to perturb PROVX.

This is because the new structures mislead the detector’s prediction mechanism. However, thanks to PROVX’s staged solidification mechanism, we can progressively lock onto the key interfering entities, enhancing the stability of the explanation results against minor, non-core perturbations in the input graph. When security analysts feed the explained counterfactual prediction structures back into the security model for retraining, we can observe significant improvements in detection performance, specifically, Accuracy, Precision, Recall, and F1 improved by 20.1%, 43.7%, 22.2%, and 35.5%, respectively.

VIII. CASE STUDY

We use an attack from the DARPA E3 Theia audit log dataset as an illustrative example for our case study. As shown in the Fig. 11, the attacker targets the Theia host, first establishing a connection to the victim host through a backdoor vulnerability in `firefox` and writing a malicious payload named `clean` to the disk. Next, the attacker uses elevated privileges to execute the payload, communicates with the attacker’s C&C server, and downloads another payload named `profile`. Finally, another payload, `wdev`, is retrieved, and a port scan of the victim’s network is executed via `gtcache` and `mail`.

In this case study, we observe that PROVX’s mechanism works by trying different combinations of perturbations to lock onto the subset of edges that truly causes the model’s prediction to flip. For example, in scenario (i), PROVX discovers that after deleting three sets of interactions—the malicious IP 146.153.68.151 with `/bin/dash`, `gtcache` with `/proc/*`, and `clean` with `wdev`—the model’s prediction remains “Attack”. This indicates that these specific interactions are not the fundamental cause of the prediction. Analyzing other scenarios similarly, we find that in scenario (iv), deleting the interactions among `clean`, `profile`, `mail`, and `wdev` directly causes the model’s prediction to flip. This directly guides security analysts to focus on the roles of the malicious payloads `clean` and `profile` in the overall attack flow and to analyze the real reason `mail` was used to exfiltrate user privacy. It is noteworthy that ProvX can also yield multiple sets of counterfactual explanations, which may contain other sets of edges and topological structures that influence the model’s decision.

IX. DISCUSSION & FUTURE WORK

Explanation Granularity. PROVX focuses on edge-granular interpretability for graph-level detection models. While we believe that the graph-level is a suitable granularity for PIDS, it can go beyond the limitations of a single data point or connection, reveal complex attack patterns that are composed of a series of seemingly normal but combined malicious activities, such as lateral movement, data theft, or DDoS attacks, and support security analysts to reasonably filter batch alerts, recent research has shown a trend towards finer granularity, with a preference for node-level [18], [20], [26], [50] and edge-level [21] analysis. Conducting interpretability analysis for node-level and edge-level detectors is a very interesting research direction. ProvX does not fall within this scope, and we look forward to future work addressing this issue.

Detection & Explanation. As discussed in the main text, *the complexity of APTs leads to the incomprehensibility of provenance-based detection, and the opacity of GNN models leads to the unreliability of security detection models*. The output of existing PIDS is thin. We firmly believe that PIDS should provide security analysts with a clear understanding of where malicious activity is located, rather than overwhelming analysis centers with a deluge of warnings (which is why alert fatigue is rampant). In our vision, the explainer is an effective support for the detector and should be integrated into existing PIDS frameworks to provide interpretability assistance. In the main text, we briefly proposed a closed-loop “Detection-Explanation-Feedback” framework for analyzing and retraining on false negatives and false positives. We will explore and refine this framework in our future work.

X. CONCLUSION

We propose PROVX, a provenance graph explainer for APT detection that uses a counterfactual explanation framework. It is designed to output human-understandable explanatory analyses for the predictions of GNN-based security models, addressing the fundamental crisis of trustworthiness and usability caused by the opaque, black-box nature of GNNs. After provenance detection, PROVX ingeniously transforms the discrete problem of graph structure edit search into a continuous optimization task. By maximizing the Prediction Flip Loss, minimizing the Mask Distance Loss, and incorporating Staged Solidification, PROVX finds the minimal structural

subset capable of flipping the model's prediction. Through evaluations on real-world APT datasets, the results show that PROVX's explanations have a high relevance to actual attacks, achieving excellent Accuracy and Recall. Furthermore, PROVX's explanations demonstrate high necessity, proving the importance of the identified edge structures to the security model's predictions. Finally, we discuss the real-world challenges of adversarial detection and explore and validate a closed-loop Detection-Explanation-Feedback enhancement framework, which serves as a powerful supplement to existing PIDS.

REFERENCES

- [1] S. Ma, X. Zhang, and D. Xu, "Protracer: Towards practical provenance tracing by alternating between logging and tainting," in *23rd Annual Network And Distributed System Security Symposium (NDSS 2016)*. Internet Soc, 2016.
- [2] 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)*. IEEE, 2019, pp. 1137–1152.
- [3] 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.
- [4] 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.
- [5] J. Zeng, Z. L. Chua, Y. Chen, K. Ji, Z. Liang, and J. Mao, "Watson: Abstracting behaviors from audit logs via aggregation of contextual semantics," in *NDSS*, 2021.
- [6] P. Gao, X. Xiao, D. Li, Z. Li, K. Jee, Z. Wu, C. H. Kim, S. R. Kulkarni, and P. Mittal, "{SAQL}: A stream-based query system for {Real-Time} abnormal system behavior detection," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 639–656.
- [7] 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 *30th USENIX security symposium (USENIX security 21)*, 2021, pp. 3005–3022.
- [8] 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)*, 2022, pp. 2461–2478.
- [9] 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*, 2019, pp. 1795–1812.
- [10] 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)*. IEEE, 2020, pp. 1172–1189.
- [11] E. Altinisik, F. Deniz, and H. T. Sencar, "Provg-searcher: A graph representation learning approach for efficient provenance graph search," in *Proceedings of the 2023 ACM SIGSAC conference on computer and communications security*, 2023, pp. 2247–2261.
- [12] K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, and D. Xu, "Hercule: Attack story reconstruction via community discovery on correlated log graph," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016, pp. 583–595.
- [13] F. Capobianco, R. George, K. Huang, T. Jaeger, S. Krishnamurthy, Z. Qian, M. Payer, and P. Yu, "Employing attack graphs for intrusion detection," in *Proceedings of the new security paradigms workshop*, 2019, pp. 16–30.
- [14] S. Li, F. Dong, X. Xiao, H. Wang, F. Shao, J. Chen, Y. Guo, X. Chen, and D. Li, "Nodlink: An online system for fine-grained apt attack detection and investigation," in *In Network and Distributed System Security Symposium (NDSS'24)*, 2024.
- [15] 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.
- [16] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," in *In Network and Distributed System Security Symposium (NDSS'20)*, 2020.
- [17] S. Wang, Z. Wang, T. Zhou, H. Sun, X. Yin, D. Han, H. Zhang, X. Shi, and J. Yang, "Thretrace: 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.
- [18] 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)*, 2024, pp. 5197–5214.
- [19] 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)*, 2021, pp. 2345–2362.
- [20] 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)*. IEEE, 2024, pp. 3533–3551.
- [21] J. Zengy, 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)*. IEEE, 2022, pp. 489–506.
- [22] 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)*, 2023, pp. 4355–4372.
- [23] M. N. Hossain, S. M. Milajerdi, J. Wang, B. Eshete, R. Gjomemo, R. Sekar, S. Stoller, and V. Venkatakrishnan, "Sleuth: Real-time attack scenario reconstruction from cots audit data," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 487–504.
- [24] A. Goyal, G. Wang, and A. Bates, "R-caid: Embedding root cause analysis within provenance-based intrusion detection," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 3515–3532.
- [25] 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.
- [26] 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)*. IEEE Computer Society, 2024, pp. 139–139.
- [27] W. Wu, W. Qiao, W. Yan, B. Jiang, Y. Liu, B. Liu, Z. Lu, and J. Liu, "Brewing vodka: Distilling pure knowledge for lightweight threat detection in audit logs," in *Proceedings of the ACM on Web Conference 2025*, 2025, pp. 2172–2182.
- [28] Y. Ji, S. Lee, E. Downing, W. Wang, M. Fazzini, T. Kim, A. Orso, and W. Lee, "Rain: Refinable attack investigation with on-demand inter-process information flow tracking," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 377–390.
- [29] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285–1298.
- [30] Y. Ji, S. Lee, M. Fazzini, J. Allen, E. Downing, T. Kim, A. Orso, and W. Lee, "Enabling refinable {Cross-Host} attack investigation with efficient data flow tagging and tracking," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1705–1722.
- [31] S. Ma, J. Zhai, F. Wang, K. H. Lee, X. Zhang, and D. Xu, "{MPI}: Multiple perspective attack investigation with semantic aware execution partitioning," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1111–1128.
- [32] T. Pasquier, X. Han, T. Moyer, A. Bates, O. Hermant, D. Eysers, J. Bacon, and M. Seltzer, "Runtime analysis of whole-system provenance," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 1601–1616.
- [33] W. U. Hassan, L. Aguse, N. Aguse, A. Bates, and T. Moyer, "Towards scalable cluster auditing through grammatical inference over provenance graphs," in *Network and Distributed Systems Security Symposium*, 2018.
- [34] Y. Kwon, F. Wang, W. Wang, K. H. Lee, W.-C. Lee, S. Ma, X. Zhang, D. Xu, S. Jha, G. Ciocarlie *et al.*, "Mci: Modeling-based causality inference in audit logging for attack investigation," in *Network and Distributed Systems Security (NDSS) Symposium*, 2018.

- [35] 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*, 2019, pp. 1777–1794.
- [36] R. Paccagnella, K. Liao, D. Tian, and A. Bates, "Logging to the danger zone: Race condition attacks and defenses on system audit frameworks," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1551–1574.
- [37] W. U. Hassan, M. A. Noureddine, P. Datta, and A. Bates, "Omegalog: High-fidelity attack investigation via transparent multi-layer log analysis," in *Network and distributed system security symposium*, 2020.
- [38] R. Yang, S. Ma, H. Xu, X. Zhang, and Y. Chen, "Uiscope: Accurate, instrumentation-free, and visible attack investigation for gui applications," in *NDSS*, vol. 24, 2020, p. 141.
- [39] D. Han, Z. Wang, W. Chen, Y. Zhong, S. Wang, H. Zhang, J. Yang, X. Shi, and X. Yin, "Deepaid: Interpreting and improving deep learning-based anomaly detection in security applications," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 3197–3217.
- [40] L. Yu, S. Ma, Z. Zhang, G. Tao, X. Zhang, D. Xu, V. E. Urias, H. W. Lin, G. F. Ciocarlie, V. Yegneswaran *et al.*, "Alchemist: Fusing application and audit logs for precise attack provenance without instrumentation," in *NDSS*, 2021.
- [41] J. Zeng, C. Zhang, and Z. Liang, "Palantir: Optimizing attack provenance with hardware-enhanced system observability," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3135–3149.
- [42] I. J. King and H. H. Huang, "Euler: Detecting network lateral movement via scalable temporal graph link prediction," in *NDSS*, 2022.
- [43] Z. Xu, P. Fang, C. Liu, X. Xiao, Y. Wen, and D. Meng, "Depcomm: Graph summarization on system audit logs for attack investigation," in *2022 IEEE symposium on security and privacy (SP)*. IEEE, 2022, pp. 540–557.
- [44] H. Ding, J. Zhai, D. Deng, and S. Ma, "The case for learned provenance graph storage systems," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 3277–3294.
- [45] 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)*, 2023, pp. 1199–1216.
- [46] 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*, 2023, pp. 2396–2410.
- [47] 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.
- [48] 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)*. IEEE, 2023, pp. 2620–2638.
- [49] R. Sekar, H. Kimm, and R. Aich, "caudit: A fast, scalable and deployable audit data collection system," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 3571–3589.
- [50] W. Qiao, Y. Feng, T. Li, Z. Ma, Y. Shen, J. Ma, and Y. Liu, "Slot: Provenance-driven apt detection through graph reinforcement learning," *arXiv preprint arXiv:2410.17910*, 2024.
- [51] 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.
- [52] T. Bilot, B. Jiang, Z. Li, N. El Madhoun, K. Al Agha, A. Zouaoui, T. Pasquier, and R. Model, "Sometimes simpler is better: A comprehensive analysis of state-of-the-art provenance-based intrusion detection systems," in *USENIX Security Symposium*, 2025.
- [53] W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, D. Wang, Z. Chen, Z. Li, J. Rhee, J. Gui *et al.*, "This is why we can't cache nice things: Lightning-fast threat hunting using suspicion-based hierarchical storage," in *Proceedings of the 36th Annual Computer Security Applications Conference*, 2020, pp. 165–178.
- [54] Y. Shen, E. Mariconti, P. A. Vervier, and G. Stringhini, "Tiresias: Predicting security events through deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 592–605.
- [55] Z. Li, X. Cheng, L. Sun, J. Zhang, and B. Chen, "A hierarchical approach for advanced persistent threat detection with attention-based graph neural networks," *Security and Communication Networks*, vol. 2021, pp. 1–14, 2021.
- [56] A. Nadeem, D. Vos, C. Cao, L. Pajola, S. Dieck, R. Baumgartner, and S. Verwer, "Sok: Explainable machine learning for computer security applications," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2023, pp. 221–240.
- [57] Y. Chen, Y. Bian, B. Han, and J. Cheng, "How interpretable are interpretable graph neural networks?" *arXiv preprint arXiv:2406.07955*, 2024.
- [58] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in neural information processing systems*, vol. 32, 2019.
- [59] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," *Advances in neural information processing systems*, vol. 33, pp. 19620–19631, 2020.
- [60] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *International conference on machine learning*. PMLR, 2021, pp. 12 241–12 252.
- [61] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schütt, K.-R. Müller, and G. Montavon, "Higher-order explanations of graph neural networks via relevant walks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 11, pp. 7581–7596, 2021.
- [62] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International conference on machine learning*. PMIR, 2017, pp. 3145–3153.
- [63] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.
- [64] A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck, "Don't paint it black: White-box explanations for deep learning in computer security," *CoRR*, 2019.
- [65] Y. Hu, S. Wang, W. Li, J. Peng, Y. Wu, D. Zou, and H. Jin, "Interpreters for gnn-based vulnerability detection: Are we there yet?" in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 1407–1419.
- [66] Z. Chu, Y. Wan, Q. Li, Y. Wu, H. Zhang, Y. Sui, G. Xu, and H. Jin, "Graph neural networks for vulnerability detection: A counterfactual explanation," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2024, pp. 389–401.
- [67] T. Ganz, M. Härterich, A. Warnecke, and K. Rieck, "Explaining graph neural networks for vulnerability discovery," in *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, 2021, pp. 145–156.
- [68] K. Mukherjee, J. Wiedemeier, T. Wang, M. Kim, F. Chen, M. Kantarcioglu, and K. Jee, "Interpreting gnn-based ids detections using provenance graph structural features," *arXiv e-prints*, pp. arXiv–2306, 2023.
- [69] F. Welter, F. Wilkens, and M. Fischer, "Tell me more: Black box explainability for apt detection on system provenance graphs," in *ICC 2023-IEEE International Conference on Communications*. IEEE, 2023, pp. 3817–3823.
- [70] "Darpa transparent computing program engagement 3 data release," <https://github.com/darpa-i2o/Transparent-Computing>, 2020.
- [71] "Darpa operationally transparent cyber (optc) data release," <https://github.com/FiveDirections/OpTC-data>, 2019.
- [72] "Adversarial tactics, techniques and common knowledge," https://attack.mitre.org/wiki/Main_Page.
- [73] T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eysers, M. Seltzer, and J. Bacon, "Practical whole-system provenance capture," in *Proceedings of the 2017 Symposium on Cloud Computing*, 2017, pp. 405–418.
- [74] 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)*, 2015, pp. 319–334.
- [75] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.
- [76] J. Tan, S. Geng, Z. Fu, Y. Ge, S. Xu, Y. Li, and Y. Zhang, "Learning and evaluating graph neural network explanations based on counterfactual

and factual reasoning,” in *Proceedings of the ACM web conference 2022*, 2022, pp. 1018–1027.

- [77] J. Tan, S. Xu, Y. Ge, Y. Li, X. Chen, and Y. Zhang, “Counterfactual explainable recommendation,” in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1784–1793.
- [78] J. Pearl, “Causal inference in statistics: An overview,” 2009.

XI. APPENDIX

A. Provenance Graph of the Audit Logs

The TABLE VIII shows how existing PIDS constructs audit log datasets into provenance graphs, but it is worth mentioning that we do not use edge connection relationship types and only retain four ways of building connections.

TABLE VIII
SYSTEM BEHAVIORS EXTRACTED FROM AUDIT LOGS.

System Behavior	Relation Description
Process \rightarrow R1 \rightarrow Process	R1: fork, execute, exit, clone, etc.
Process \rightarrow R2 \rightarrow File	R2: read, open, close, write, etc.
Process \rightarrow R3 \rightarrow Netflow	R3: connect, send, rcv, write, etc.
Process \rightarrow R4 \rightarrow Memory	R4: read, mprotect, mmap, etc.

B. Improved Louvain Detail

Here we give the specific implementation of the improved Louvain:

In order to effectively analyze large-scale traceability graphs, we designed and implemented a subgraph partitioning algorithm based on the Louvain community detection algorithm. The core goal of this algorithm is to decompose the huge monomer traceability graph into a series of subgraphs with controllable size and semantic cohesion to facilitate the processing of downstream tasks. The main workflow of this algorithm includes: graph preprocessing, community detection based on the Louvain algorithm, partitioning of ultra-large-scale communities, and context expansion and formatted output of subgraphs. Our subgraph partitioning method includes the following key steps:

1) *Graph Construction and Preprocessing*: The algorithm loads the graph's topology from an edge list and reads the set of known attack/malicious nodes from a node list. Before performing community detection, we preprocess the graph by removing all isolated nodes (nodes with a degree of 0). This step is necessary because isolated nodes contain no relational information and do not contribute to structure-based community detection or subsequent graph learning tasks.

2) *Community-based Initial Partitioning*: We employ the widely-used Louvain community detection algorithm to partition the preprocessed graph. The Louvain algorithm can efficiently discover modular structures in a network, identifying densely connected groups of nodes as communities. This aligns with the characteristic of provenance graphs where system behaviors are clustered logically and temporally; therefore, the resulting communities typically represent relatively complete behavioral units.

If the number of nodes in a community exceeds a preset threshold, S_{max} , we perform a forced partitioning on it. This is to ensure that all finally generated subgraphs are within a manageable size, preventing difficulties in downstream model analysis or memory overflow issues due to excessively large individual subgraphs. The partitioning operation sequentially

Algorithm 1 Provenance Subgraph Partitioning Algorithm

Require: Graph edge file E_{file} , Attack node file A_{file} , Max subgraph size S_{max}

Ensure: Formatted subgraph data file O_{file}

```

1: procedure PARTITIONPROVENANCE-
  GRAPH( $E_{file}, A_{file}, S_{max}$ )
2:    $\triangleright$  Step 1: Graph loading and preprocessing
3:    $G \leftarrow \text{ReadGraphFromEdges}(E_{file})$ 
4:    $A_{nodes} \leftarrow \text{ReadAttackNodes}(A_{file})$ 
5:    $G_p \leftarrow \text{RemoveIsolatedNodes}(G)$ 
6:    $\triangleright$  Step 2: Community detection and splitting
7:   if  $G_p$  has no edges then
8:      $C \leftarrow \text{PartitionNodesBySize}(G_p.\text{nodes}(), S_{max})$ 
9:   else
10:     $C_{raw} \leftarrow \text{LouvainCommunityDetection}(G_p)$ 
11:     $C \leftarrow \emptyset$ 
12:    for all community  $c$  in  $C_{raw}$  do
13:      if  $|c| > S_{max}$  then
14:         $C \leftarrow C \cup \text{SplitCommunity}(c, S_{max})$ 
15:      else
16:         $C \leftarrow C \cup \{c\}$ 
17:     $\triangleright$  Step 3: Finalize subgraphs with contextual
    expansion
18:    Open  $O_{file}$  for writing
19:    for all node partition  $P$  in  $C$  do
20:       $G_{sub} \leftarrow G_p.\text{subgraph}(P)$ 
21:      if  $G_{sub}.\text{number\_of\_edges}() = 0$  then
22:         $\triangleright$  Expand context for internally disconnected
        partitions
23:         $N_{out}, E_{out} \leftarrow \text{GetOneHopNeighborhood}(G_p, P)$ 
24:      else
25:         $N_{out} \leftarrow P$ 
26:         $E_{out} \leftarrow G_{sub}.\text{edges}()$ 
27:         $\triangleright$  Step 4: Labeling and writing to file
28:         $is\_attack \leftarrow (\exists n \in N_{out} \text{ such that } n \in A_{nodes})$ 
29:        WriteFormattedSubgraph( $O_{file}, N_{out}, E_{out}, is\_attack$ )
30:    Close  $O_{file}$ 

```

splits the node list of the large community into multiple sublists that satisfy the S_{max} limit.

3) *Subgraph Contextual Expansion and Label Generation*: When generating the final subgraph files, we designed a key contextual expansion mechanism. For a partitioned subgraph, we first check its internal connections.

- If edges exist within the subgraph, its set of nodes and edges remains unchanged.
- If there are no internal edges within the subgraph (i.e., all connections between its nodes were severed during partitioning), we perform a One-hop Neighbor Expansion to preserve its structural information from the original graph. Specifically, we find all the direct neighbors of these nodes in the original graph and include these neighboring nodes and the edges connecting them into

TABLE IX
DETAILS OF THE DATASETS.

Datasets	# of Benign Graphs	Avg # of Nodes / Edges	(Min-Max) # of Nodes / Edges	# of Malicious Graphs	Avg # of Nodes / Edges	(Min- Max) # of Nodes / Edges
DARPA OpTC ¹						
OpTC	8339	50.97 / 48.06	(2-233) / (1-240)	1534	66.82 / 92.26	(7-4012) / (1-4011)
DARPA TC E3						
Cadets	3450	100.25 / 100.72	(1-110) / (1-344)	144	100.37 / 104.74	(22-104) / (2-636)
Theia	3159	103.11 / 99.80	(2-2260) / (1-12046)	321	102.07 / 114.61	(94-106) / (4-186)
Trace	11633	99.98 / 86.20	(1-312) / (1-4212)	686	101.05 / 100.55	(66-175) / (3-861)

¹ The DARPA OpTC dataset contains different types of attacks over three days. The first day depicts the rehearsal scenario of PowerShell Empire. The second day records the data leakage incident. The third day records the upgrade of malware. In the explanation, we construct the three days of data together as the origin subgraph and do not separate them for separate analysis..

the subgraph. This step significantly enriches the subgraph's structural information and prevents the creation of information islands due to partitioning.

Finally, based on whether the final set of nodes in the subgraph contains any known attack nodes, we label the subgraph as "Malicious" or "Benign".

4) *Formatted Output*: The algorithm writes each processed subgraph to an output file in a standardized four-line format. This format includes the subgraph's unique identifier, its final list of nodes, its final list of edges (source and destination nodes), and its malicious/benign label, facilitating direct parsing and use by subsequent programs.

C. Details of the Datasets

TABLE IX shows the size of the dataset subgraphs we used, which is divided by the algorithm XI-B. We used two authoritative APT datasets, DARPA OpTC and DARPA TC E3, and selected the widely used Cadets, Theia, and Trace in the E3 dataset. As you can see, we cut the origin graph into several subgraphs, each with different numbers of nodes and edges, and divided them into benign subgraphs and malignant subgraphs according to the groundtruth. It is worth noting that we can divide the subgraphs into different sizes for different degrees of graph interpretation, but too large a granularity may increase the difficulty of interpretation and the understandability of human experts. Interested readers can try the interpretation results of different granularities.

D. Case Study in Baselines

Here we expand the case study on the baselines. After the explanations of these six fact-based explainers, different explanations are output respectively. Among them: the gradient-based method GradCam and the decomposition-based methods DeepLIFT and GNN-LRP can hardly obtain a more accurate attack explanation. The perturbation-based methods SubgraphX and GNNExplainer can provide a partial set of structures that contain some attack causes, but the former has a range deviation, while the latter contains some benign behaviors (which may also participate in model decision-making, but will mislead security analysts). PGExplainer locks

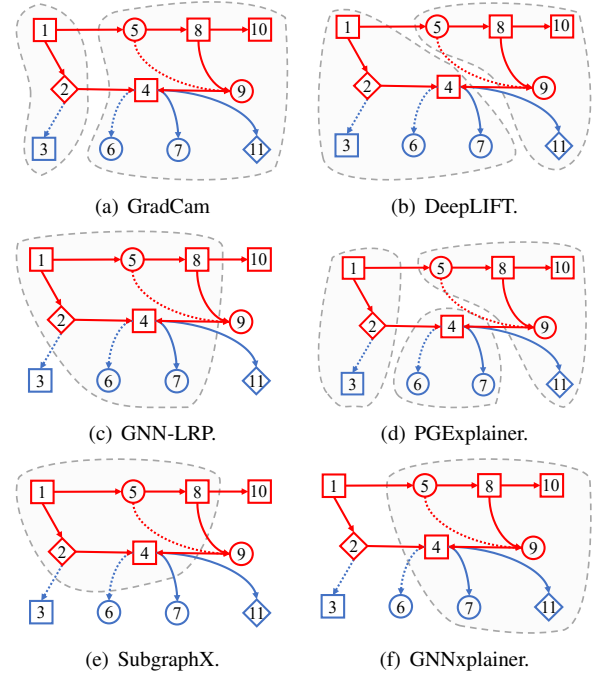


Fig. 12. The case study inherited from VIII includes six baselines: GradCam, DeepLIFT, GNN-LRP, PGExplainer, SubgraphX, and GNNExplainer.

the explanation into three more accurate sets of structures. By analyzing ⑤, ⑧, ⑩, ⑨, ⑪, we can roughly obtain understandable explanations, but it is still not intuitive enough.