USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Cutting the Fuse: Actionable APT Attack Blocking in Provenance-based IDS

Weiheng Wu, *School of Cyber Engineering, Xidian University;* Wei Qiao, *School of Cyber Engineering, Xidian University; State Key Laboratory of Integrated Services Networks (ISN);* Teng Li, *School of Cyber Engineering, Xidian University; State Key Laboratory of Integrated Services Networks (ISN); Songshan Laboratory;* Yebo Feng, *Nanyang Technological University;* Zhuo Ma and Jianfeng Ma, *School of Cyber Engineering, Xidian University;* Yang Liu, *Nanyang Technological University*

https://www.usenix.org/conference/usenixsecurity26/presentation/wu-weiheng

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

Cutting the Fuse: Actionable APT Attack Blocking in Provenance-based IDS

Weiheng Wu[1,*], *Wei Qiao*[1,3,*], *Teng Li*[1,3,4,†], *Yebo Feng*[2], *Zhuo Ma*[1], *Jianfeng Ma*[1], *Yang Liu*[2]
[1]*School of Cyber Engineering, Xidian University*
[2]*Nanyang Technological University*
[3]*State Key Laboratory of Integrated Services Networks (ISN)*
[4]*Songshan Laboratory*
wdz478149507@gmail.com, qiaoweiXidian@gmail.com, litengxidian@gmail.com
yebo.feng@ntu.edu.sg, mazhuo@mail.xidian.edu.cn, jfma@mail.xidian.edu.cn, yangliu@ntu.edu.sg

Abstract

Provenance-based Intrusion Detection Systems (PIDS) are a critical line of defense against Advanced Persistent Threats (APTs). However, their practical deployment is crippled by alert flooding; state-of-the-art PIDS focus on detection performance, outputting thousands of low-information, unactionable alerts. These outputs fail to isolate the true cause of the detector's alert and cannot provide the security operations center(SOC) with actionable blocking strategies.

To address this challenge, we propose PROVX, a novel APT defense framework designed for blocking attack steps within alerts. Unlike existing works, PROVX does not aim to merely improve attack detection performance; it is dedicated to transforming unactionable raw alerts into a core edge list for immediate review. Specifically, we introduce counterfactual explanation logic to reduce the alerts and find the minimal structural subset (i.e., the core attack edges) within an alert that determines its malicious prediction. This subset, when perturbed, can subvert the model's original prediction. The framework then applies a staged solidification strategy to enhance the precision and stability of the alert analysis. Ultimately, PROVX outputs an actionable core edge list for immediate response and blocking of critical attack steps. Experiments on DARPA datasets demonstrate that PROVX can locate key behaviors within alerts that are highly relevant to real-world attacks, and the identified core edges can flip many detector alerts under the simulated intervention used in our model-level evaluation. Furthermore, we explore and provide a preliminary validation of an alert-feedback enhancement framework, showing that PROVX's analysis results can guide model optimization in adversarial scenarios.

1 Introduction

Advanced Persistent Threats (APTs) pose a severe challenge to modern cybersecurity due to their long-term persistence, multi-stage nature, and high stealth. Traditional defense mechanisms (e.g., firewalls, antivirus) struggle to detect such complex attack chains. To counter this threat, the security community's focus has shifted to the host interior, leveraging operating system audit logs to capture and correlate system activities. In this context, data provenance technology has emerged as one of the most promising solutions. The provenance graph constructs a unified causal graph from discrete log entries (e.g., process creation, file access, network connections), providing security analysts with a structured view to depict the entire scope of an attack.

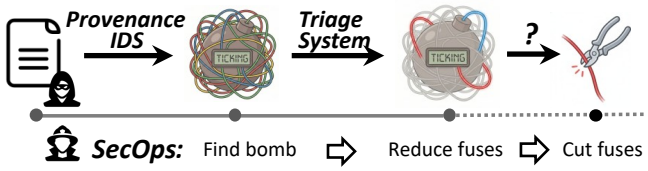


Figure 1: The status quo of APT defense in SecOps.

The rise of provenance technology has spurred the development of provenance-based intrusion detection systems (PIDS). Early PIDS relied primarily on rule-based methods [16,19,28] or statistics-based methods [18, 25]. While these methods could achieve low false-positive rates, they were highly dependent on prior knowledge, struggled to discover unknown attack patterns, and often performed poorly in real-world environments with voluminous and noisy log data. In recent years, the research focus has clearly shifted to learning-based methods [5, 8, 13, 14, 24, 32–34, 37, 43, 46], especially Graph Neural Networks (GNNs). The powerful graph representation learning capabilities of GNNs make them naturally suited for processing complex provenance graph structures.

While the research community has achieved excellent performance in attack detection tasks, even exceeding 99% accuracy, through the design of various graph embedding and aggregation strategies, industry security operations (SecOps) are simultaneously plagued by a problem: alert fatigue. To achieve high recall and capture all potential threats, these sys-

[*]Weiheng Wu and Wei Qiao contributed equally to this work.
[†]Corresponding authors: Teng Li.

tems are designed to be highly sensitive, inevitably leading to an overwhelming flood of alerts, the vast majority of which are unactionable noise. As shown in Figure 1, while a PIDS can successfully identify that a "bomb" is ticking within the host, it leaves analysts with a tangled mess of thousands of potential "fuses"—discrete system events—without any guidance on which ones are critical. Our experience indicates that a typical enterprise SOC is often inundated with thousands of daily alerts, far exceeding the operational capacity of a ten-person team, which can effectively process only 300.

This order-of-magnitude gap between detection and response forces analysts to ignore the vast majority of alerts, allowing real threats to be buried in the noise. While existing triage works (e.g., NoDoze [17]) attempt to reduce the number of *fuses* through statistical refinement, they often fail to pinpoint the exact choke point required to disarm the threat. Therefore, we argue that the next critical challenge for the PIDS domain is no longer just improving detection rates, but rather providing analysts with immediately actionable attack blocking solutions—identifying the minimal, precise "wire" that must be cut to neutralize the attack chain.

To address these challenges, we propose ProvX, an APT alert-refinement framework designed to synthesize blocking-oriented response guidance. We use mitigation in a model-relative sense: ProvX identifies edges whose simulated removal flips the detector's prediction. Unlike existing PIDS that focus solely on detection, ProvX aims to provide analysts with candidate choke points within alerted provenance subgraphs. Drawing inspiration from the potential of explainability, we introduce interventional reasoning and formalize the defense task as a minimal intervention problem. This involves pinpointing the critical *fuses* (i.e., the minimal set of system edges) whose simulated removal changes the detector's judgment, thereby revealing the structural evidence most necessary to the alert. Furthermore, we explore a closed-loop enhancement framework, leveraging these blocking-oriented results to guide the optimization of the underlying detector.

In terms of design, ProvX's core idea is to transform the discrete problem of searching for attack choke points into a continuous optimization task. Specifically, ProvX learns a numerically differentiable blocking mask to simulate the process of an analyst cutting specific attack fuses within the provenance graph. This process is guided by a dual-objective optimization: first, a prediction flip loss, which drives the model to find the structural intervention required to change the detector's malicious prediction; second, a mask distance loss, which ensures the operational intervention cost is minimized by localizing only the most critical edges. Finally, ProvX employs a staged solidification strategy to lock in confident judgments on important interactions, enhancing the stability and clarity of the identified choke points. Ultimately, this framework provides a high-confidence, actionable blocking list on top of foundational GNN detectors.

We evaluate ProvX's mitigation efficacy on the authoritative DARPA TC E3 and DARPA OpTC datasets. These real-world attack scenarios provide realistic attack scenarios for evaluating core-edge localization and model-level blocking efficacy. Our experiments show that ProvX achieves a recall rate of up to over 70% in pinpointing ground-truth attack steps. More importantly, we introduce the Mitigation Efficacy Rate (MER) to evaluate whether the recommended core-edge intervention flips the detector's alert under a controlled structural-attribute intervention, achieving an average MER of 45.74%. We quantitatively define the cost-benefit trade-off for security operations and demonstrate that ProvX significantly outperforms state-of-the-art PIDS and statistical triage tools. Finally, we validate ProvX as a forensic analyzer for adversarial evasion, exploring a feedback loop that uses identified evasion structures to harden the detector against future attacks. Our contributions are as follows:

- We propose ProvX, a novel APT defense framework. Its goal is not merely attack detection, but to refine massive alerts into an actionable core edge list for attack mitigation.
- ProvX applies interventional reasoning based on counterfactual logic to identify the most critical system interactions within a detected threat, and formalize attack mitigation as a minimal intervention optimization task.
- We design and validate a closed-loop feedback framework that integrates detection, mitigation, and forensic feedback, that can enhance the underlying detector's robustness against adaptive attacks.
- We evaluate ProvX on authoritative APT datasets across key dimensions, including localization accuracy, mitigation efficacy, and SecOps cost-benefit performance. Comprehensive comparisons against SOTA PIDS and statistical triage algorithms prove ProvX's superiority in generating high-value, low-cost blocking strategies.

2 Background and Motivation

2.1 Related work

2.1.1 System Provenance

System provenance technology is widely adopted in academia to conduct attack investigations on large-scale host audit logs(Table 1). Research on Provenance-based Intrusion Detection Systems (PIDS) has roughly progressed through three phases. Early PIDS relied primarily on rule-based methods [16, 19, 28] or statistics-based methods [18, 25]. While these approaches achieved low false-positive rates in specific scenarios by strictly adhering to expert knowledge, predefined attack templates, or event frequencies, their performance was highly dependent on expert guidance. This dependency caused them to sacrifice the detection capability for unknown

System	Time	Encoder	Decoder	Granularity
SLEUTH [19]	2017	-	-	Path
UNICORN [14]	2020	-	-	Graph
SIGL [15]	2021	Graph LSTM	MLP	Graph
ATLAS [5]	2021	LSTM+CNN	-	-
THREATRACE [43]	2022	GraphSAGE	-	Node
SHADEWATCHER [46]	2022	TransR+GNN	Inner Product	Edge
MAGIC [22]	2024	GAT	GAT+MLP	Node
KAIROS [8]	2024	TGN	MLP	Graph
R-CAID [13]	2024	GAT	-	Path
FLASH [34]	2024	GraphSAGE	XGBoost	Node
ORTHRUS [23]	2025	Graph-Transformer	MLP	Node
SLOT [33]	2025	GCN+RL	MLP	Node

Table 1: Analysis of representative provenance-based detection studies in recent years.

or potential threats, making them ill-suited for the complexity and variability of APT attacks.

Consequently, the research focus has shifted to learning-based methods [5, 8, 13, 14, 24, 32–34, 37, 43, 46], particularly Graph Neural Networks (GNNs). The powerful graph representation learning capabilities of GNNs are naturally suited for processing complex provenance graph structures, and their effectiveness in detecting APTs has been demonstrated. A significant body of work has focused on using GNNs and various custom GNN-based models to learn embeddings for downstream detection tasks. Although these methods can achieve good detection performance by understanding the structural and semantic information of audit logs, high false-positive rates and unactionable alerts remain critical bottlenecks. These issues prevent security analysts from effectively localizing the true attack and forming actionable investigation results. Therefore, recent work [8, 33, 34] has trended towards performing post-detection attack reconstruction to enhance comprehensibility for security analysts, even if this only reveals highly aggregated attack groups.

2.1.2 Alert Fatigue and Triage

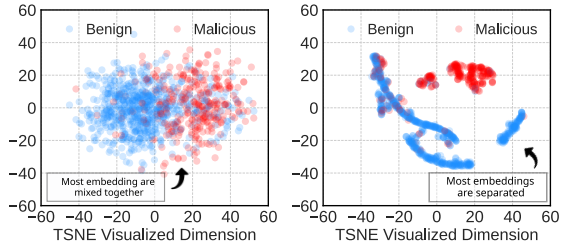
Despite high detection performance, modern PIDS face a critical operational hurdle: Alert Fatigue. A typical enterprise SOC may receive over 17,000 alerts per week, with over 51% of them considered false positives (and some studies [17] citing this figure as high as 60-80%) [41]. This volume creates an order-of-magnitude gap between detection and the limited response capacity of security analysts.

To mitigate this, triage tools like NoDoze [17] and DepImpact [10] were developed to rank alerts based on statistical heuristics, such as event frequency or path weights. However, these methods are often ill-equipped to handle Living-off-the-Land (LotL) attacks, where attackers use legitimate system commands that mimic benign statistical signatures. Consequently, these triage efforts merely reduce the number of leads without identifying the truly critical nodes in the attack chain.

2.1.3 Actionability in SecOps: The Missing Link

The ultimate goal of security operations is not just to detect a "ticking bomb" but to effectively disarm it. However, a significant actionability gap persists between state-of-the-art PIDS and practical defense. While current systems may accurately flag a subgraph as malicious, they often output thousands of unprioritized entities, leaving analysts without a clear blocking strategy.

As highlighted in the SoK by Inam et al. [21] , current anomaly-based detection offers poor actionable explanations. The study suggests that an effective solution must isolate anomalous activity to a minimal set of paths to be useful for investigators. We argue that the next frontier for PIDS lies in interventional actionability—transitioning from post-hoc explanation to providing a prescriptive defense scheme that pinpoints the candidate choke points that should be prioritized for blocking-oriented response.



(a) Embedding distribution analysis of pure semantic encoding.

(b) Embedding distribution analysis of structural coding

Figure 2: Embedding distributions of benign and malicious subgraph patterns.

2.2 Motivation of PROVX

2.2.1 Key Observation

We first examine which signal is most reliable for distinguishing malicious behavior from benign behavior. By decomposing a complete APT attack provenance graph, we can obtain several smaller subgraphs that represent specific attack patterns or benign behaviors. As illustrated in Figure 2, by comparing benign and malicious subgraphs in terms of both semantic and structural similarity, we arrive at a key observation: subgraphs related to different attacks exhibit clear structural similarities to one another, and the structural properties of these attack subgraphs also show significant differences from those of benign subgraphs. In contrast, semantic attributes do not offer this clear distinguishing capability.

In provenance graphs, causal dependency structure is a necessary condition for the existence of attack chains. Attackers can forge filenames (semantics), but it is difficult to forge causal paths (structure) without breaking the attack chain. Mature cybersecurity frameworks, such as Kill Chain and MITRE

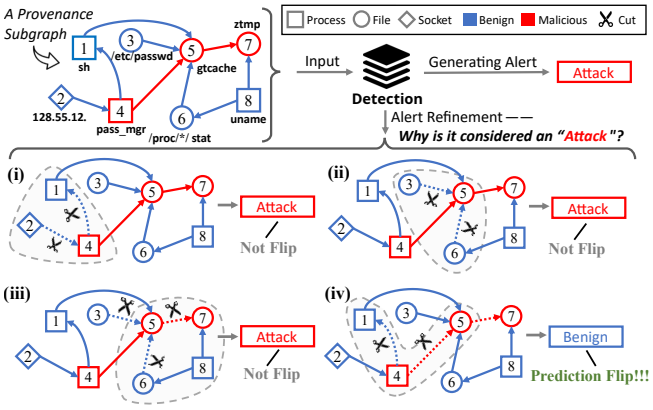


Figure 3: A simplified attack example from DARPA TC E3 Trace. Panels (i)–(iv) show different simulated interventions under counterfactual reasoning, leading to either ***Not Flip*** or ***Prediction Flip*** outcomes in the detector. A prediction flip indicates that the selected edges are important to the detector's malicious judgment and can guide analyst review of candidate blocking points.

ATT&CK [1], also confirm this finding, describing the interrelated and predictable structural dependencies between attack strategies and techniques. Therefore, our framework focuses on precisely locating these key structural bottlenecks in attribution graphs to generate a list of actionable APT mitigation and prevention measures.

2.2.2 Attack Blocking for Provenance Detection

We use the APT attack shown in the Figure 3 as a driving example for illustration. 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.

Taking the attack case above as an example, some existing studies are limited to graph analysis, outputting only graph-level alerts [8, 14]. This coarse granularity poses a significant challenge for security analysts: it only informs the analyst that "this graph is problematic" without revealing which specific internal structures triggered the alert. Simply put, the analyst is unable to discern the root cause of the alert. Furthermore, while subsequent research has begun to focus on node-level [33, 34, 43] or edge-level [46] detection tasks, the effort to capture all attacks has resulted in a high volume of false positives, directly leading to alert flooding. Our focus is therefore on how to automatically refine a suspicious alerted graph down to its core components—the parts that analysts can immediately review, remediate and block the attack.

We observe that, in provenance-based alerts, attack blocking and counterfactual reasoning share a useful structural analogy. Security operations seeks low-cost interventions that disrupt suspicious dependencies (e.g., terminating a process, deleting a file) , while counterfactual reasoning seeks minimal structural perturbations that flip the detector's prediction. ProvX uses this analogy to derive model-level blocking guidance, without equating prediction flipping with guaranteed real-world attack disruption. Accordingly, we employ counterfactual logic to refine alerts. Our objective is to investigate: what minimal structural modifications (simulating interventions) to the original alerted graph would cause the detection system to reverse its prediction to benign? Therefore, we adopt a counterfactual explanation approach to explore various hypothetical structural changes (as shown in (i), (ii), (iii), and (iv) in Fig. 3). For instance:

- (i) After deleting edges ②-④ and ④-① (representing partial interaction between `pass_mgr` and `sh`), the alert is still triggered. 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 root cause for the system triggering the alert.

- (ii) and (iii) explore other simulated interventions, blocking different sets of interactions such as ③-⑤ or ⑥-⑤. These attempts also yield comparable results, failing to flip the system's prediction. This demonstrates that, like the intervention in (i), these targeted changes are insufficient on their own to resolve the alert.

- 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, indicating that these edges are critical to the detector's malicious judgment.

This result reveals that the minimal flipping edge set ④-① and ④-⑤ is precisely the critical basis on which the detector triggers the alert. ProvX automatically refines a large, complex alert into a concise, high-relevance core edge list (Figure 3(iv)). At the operational level, these edges become natural candidates for analyst validation: preventing `pass_mgr` from accessing sh and from launching gtcache may stop the `gtcache` process from being created, thereby disrupting subsequent behaviors such as *communicating with the C&C server* and *scanning for sensitive information*. This list helps analysts reason about the alert's likely choke points, and achieve a more thorough and comprehensible analysis of the alert's root cause and led to block this attack.

3 Problem Formalization

3.1 Definition

We define key designs and notations for the task. In a SOC deployment, an alert may originate from a provenance-based detector or from other security signals, such as endpoint alarms, network indicators, or analyst investigation. These alerts can be mapped to relevant host audit events and expanded into suspicious provenance subgraphs for downstream analysis. In this paper, we instantiate this setting with graph-level PIDS alerts, but the formalization only requires an alert-associated provenance subgraph as input. Assume we have N such provenance subgraphs $\{G_1, G_2, G_3, \ldots, G_N\}$, 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, while 1 indicates that the subgraph contains at least one stage of an APT attack.

We use a GNN-based detector $f(\cdot)$ to classify these provenance subgraphs. During testing, the trained detector predicts each subgraph G_k and outputs its predicted label $\hat{Y}_k$. PROVX focuses on triggered alerts, i.e., subgraphs predicted as malicious by the detector. The goal of PROVX is to support attack blocking by converting such an alert into a minimal core edge set that analysts can prioritize for response.

Since a provenance graph is an abstraction of host-level behavior, PROVX first solves a model-level blocking problem: it asks which simulated edge-level interventions on the alert graph are sufficient to flip the detector's malicious prediction. For an alerted subgraph G_k, we generate a counterfactual subgraph $\tilde{G}_k$ by perturbing a subset of edges. If the detector's prediction changes, i.e., $f(\tilde{G}_k) \neq f(G_k)$, the perturbed edges are considered critical to the detector's original alert. Our objective is to find the smallest such structural perturbation, so that the resulting core edge set is concise enough for analyst review and possible host-level enforcement. The mathematical definition is as follows:

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

where $d(\tilde{G}_k, G_k)$ measures the structural difference between the original alert and its counterfactual version, and $\hat{\tilde{Y}}_k$ is the detector's prediction on $\tilde{G}_k$.

This objective defines blocking at the detector level. A feasible solution means that the selected edges are necessary for the current PIDS model to maintain its malicious judgment. It does not by itself prove that the corresponding host-level enforcement action is always safe, feasible, or sufficient to disrupt the real attack. In deployment, the resulting core edge list should therefore be treated as analyst-prioritized blocking guidance rather than an autonomous blocking policy.

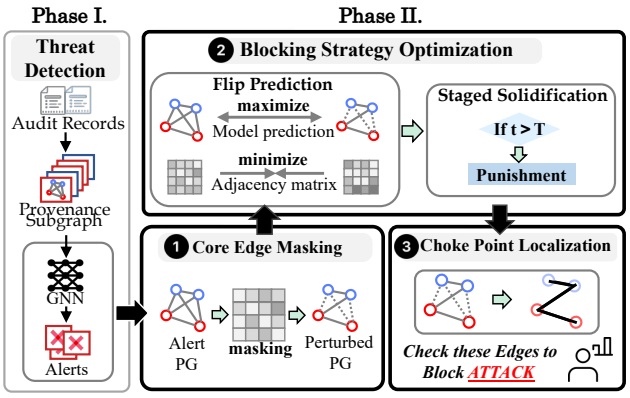


Figure 4: PROVX working mode with two phases.

3.2 Threat Model and Analysis 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 [14, 27, 28, 42], 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 [6, 30]. We acknowledge that this assumption may not hold in all deployments: attackers may tamper with logging pipelines or exploit blind spots in audit collection. Under such incomplete logging, PROVX can only reason over the observed provenance graph.

Considering the increasing prevalence of adversarial attacks in the current field of network security, we also carefully evaluate the performance changes of the core edge localization algorithm 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 Section 6 of this paper.

4 The PROVX Framework

As shown in Figure 4, PROVX is a two-phase alert refinement framework designed to resolve alert flooding and support attack blocking.

Phase I: Threat Detection (4.1). This phase obtains suspicious provenance subgraphs for downstream refinement. In this paper, we instantiate it with a GNN-based graph-level detector over provenance subgraphs. In deployment, this acquisition step can also be replaced by mature SOC alert-discovery pipelines, such as existing PIDS alerts, EDR/SIEM alarms expanded through provenance queries, network indicators mapped to host events, or analyst-guided investigation.

Phase II: Attack Mitigation (4.2). PROVX refines each alerted subgraph with counterfactual reasoning and outputs a core edge list as blocking-oriented response guidance. This phase consists of three steps: First, ❶ **Core Edge Masking** relaxes the discrete edge intervention search into a continuous core edge mask learning task. Then, ❷ **Blocking Strategy Optimization** uses a dual-objective optimization framework to flip the detector's prediction while minimizing the simulated intervention cost, supplemented by staged solidification to improve stability. Finally, ❸ **Choke Point Localization** converts the learned mask into a concise core edge list for analyst review and possible host-level enforcement.

4.1 Threat Detection

Phase I is an instantiation of how PROVX obtains alert-associated provenance subgraphs. In real SOC deployments, suspicious subgraphs may come from existing PIDS, endpoint alarms, network indicators, or analyst-guided provenance expansion. We construct this phase using a GNN-based graph-level detector and a Louvain-based partitioning procedure to approximate the alert-subgraph acquisition process.

Existing detectors use GNNs for autoencoding [22], semi-supervised detection [33], and other detection objectives. However, the main contribution of PROVX lies in downstream alert refinement rather than in proposing a new detector. Therefore, we map APT detection to a GNN-based subgraph classification task and use it to obtain alerted provenance subgraphs for Phase II. Specifically, we focus on local structures within provenance subgraphs, such as suspicious processes, abnormally accessed files, or interactions between critical system events, and classify whether these structures and their local context are associated with malicious activity. Next, we detail this experimental instantiation:

(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 a Louvain-based community partitioning procedure [7] to extract provenance subgraphs with manageable sizes. Benign subgraphs consist entirely of benign system entities, while malicious subgraphs contain at least one attack-related entity or attack stage. To preserve contextual completeness, we allow important hub nodes to appear in multiple extracted subgraphs when needed. This partitioning procedure is used to create evaluation subgraphs with controllable size and contextual completeness; it is not required when a deployment already provides alert-associated provenance subgraphs through mature SOC workflows. Alternative ways to construct alert-associated subgraphs are discussed in Section 7.

We extract N provenance subgraphs $\{G_1, G_2, \ldots, G_N\}$, 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.

(3) Threat Detection. The goal of provenance detection is to predict a label for each extracted subgraph G_k, and trigger an alert when it is associated with an APT. 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 chosing the highest probability:

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

For model selection, we adopt standard differentiable GNN frameworks, including GCN, GAT, and GraphSAGE[1]. While existing systems such as SLOT [33] may include non-differentiable components such as reinforcement learning, the current counterfactual optimization in PROVX requires a compatible differentiable detector to compute prediction flips. Thus, the alert-discovery source is replaceable, but the Phase II optimization currently assumes access to a differentiable GNN-based detector.

4.2 Attack Mitigation

4.2.1 Core Edge Masking

The first step of the PROVX attack mitigation algorithm is to simulate edge-level interventions within an alert-associated provenance subgraph G_k. Since APT alerts often depend on structural dependencies, our objective is to learn a core edge mask that can flip the detector's alert at minimal simulated intervention cost. We focus on perturbing system entity interactions. For an alert subgraph G_k, its structure is represented by the adjacency matrix $A_k \in \{0, 1\}^{n \times n}$.

Directly searching for a discrete, binary ($\{0, 1\}$) blocking mask is an NP-hard combinatorial optimization problem that cannot be solved directly. Therefore, PROVX relaxes this discrete search problem into a continuous, differentiable optimization task. Specifically, we introduce a learnable, continuous-valued parameter matrix $\hat{M}_k \in \mathbb{R}^{n \times n}$ with the same dimensions as A_k. We then use the sigmoid function $\sigma(\cdot)$ to map this matrix into the (0, 1) range, generating a soft mask. This soft mask represents the probability of retaining each edge. The perturbed adjacency matrix $\tilde{A}_k$ is then defined as:

$$\tilde{A}_k = A_k \odot \sigma(\hat{M}_k), \tag{3}$$

where $\odot$ denotes element-wise multiplication. The $\sigma(\cdot)$ function provides a smooth transition between "retaining" (a value near 1) and "blocking" (a value near 0). Consequently, starting from a randomly initialized mask, $\hat{M}_k$ can be optimized via

[1] It is worth noting that many custom models in prior PIDS works [13, 22, 33, 34, 43, 46] are also built on these basic GNN architectures.

gradient descent to learn the minimal set of edges required to flip the detector's alert.

4.2.2 Blocking Strategy Optimization

We conduct the core optimization task by learning masks for the provenance subgraphs (i.e., alerts) predicted by the security model as malicious. The core idea of algorithm is to determine the minimum perturbation of the provenance subgraph (i.e., the minimal blocking set), which is achieved by establishing an optimization task and optimizing the composite loss function. It is worth mentioning that our blocking strategy 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 alert subgraph as described in section 4.2.1 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 Phase I to generate the corresponding prediction results, which are:

$$\hat{Y}_k = \text{GNN}(A_k), \quad \tilde{Y}_k = \text{GNN}(\tilde{A}_k). \tag{4}$$

We learn the edge mask $\hat{M}_k$ based on the optimization objective of counterfactual reasoning to determine the minimal perturbation. The mask learning process is achieved by minimizing a composite loss function $\mathcal{L}_{\textbf{ProvX}}$ over T training epochs. This loss function simulates the trade-off between attempting to flip the detector's alert (i.e., change the model's prediction) and minimizing the response cost (i.e., minimizing modifications to the original activity logs). It consists of a primary counterfactual loss $\mathcal{L}_{\textbf{CFX}}$ and a staged solidification loss $\mathcal{L}_{\textbf{S}}$.

(1) Primary Counterfactual Loss $\mathcal{L}_{\textbf{CFX}}$. The guiding principle of $\mathcal{L}_{\textbf{CFX}}$ is to find modifications to the provenance graph that can change the detector's malicious prediction while keeping the perturbation (i.e., blocking cost) as small as possible. Thus, $\mathcal{L}_{\textbf{CFX}}$ is made up of two loss terms, Prediction Flip Loss and Mask Distance Loss. The Prediction Flip Loss $\mathcal{L}_{\textbf{Pred}}$ 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., the alert is cleared). 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:

$$\mathcal{L}_{\textbf{Pred}} = P(\hat{Y}_k, \tilde{A}_k). \tag{5}$$

The Mask Distance Loss $\mathcal{L}_{\textbf{dist}}$ aims to ensure that the identified provenance graph modification is as small as possible. This means identifying only those most critical, core activity segments or system interactions that form the root cause of the alert. 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:

$$\mathcal{L}_{\textbf{dist}} = \text{BinaryCrossEntropy}(\sigma(\hat{M}_k), A_k). \tag{6}$$

Then, we integrate the above two loss terms into the primary counterfactual loss function to optimize them collaboratively:

$$\mathcal{L}_{\textbf{CFX}} = \alpha \cdot \mathcal{L}_{\textbf{pred}} + (1-\alpha) \cdot \mathcal{L}_{\textbf{dist}}, \tag{7}$$

where α is responsible for regulating the trade-off between the predict flip loss term and the mask distance loss term. A higher α prioritizes flipping the detector's alert at the expense of larger perturbations, while a lower α focuses more on minimizing graph modifications.

(2) Staged Solidification Loss $\mathcal{L}_{\textbf{S}}$. To provide security analysts with clearer and more reliable localization results for critical attack paths, 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. At the T_{start} epoch, the currently activated edge mask (specifically for $\sigma(\hat{M}_k)$, denoted as $M^k_{snap} = \sigma(\hat{M}_k^{(T_{start})})$ 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 $I_{low} = \{i | m^k_{snap,i} < \tau_{low}\}$ (edges whose mask values are already close to 0 in the snapshot),
- For high confidence edge index set $I_{high} = \{j | m^k_{snap,j} > \tau_{high}\}$ (edges whose mask values are already close to 1 in the snapshot).

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

$$\begin{aligned} \mathcal{L}_S = &\sum_{i \in I_{low}} (\sigma(\hat{M}^{(t)}_{k,i}) - m^k_{snap,i})^2 \\ &+ \sum_{j \in I_{high}} (\sigma(\hat{M}^{(t)}_{k,j}) - m^k_{snap,j})^2. \end{aligned} \tag{8}$$

This penalty term targets edges whose mask values already showed a clear trend. This encourages the mask values to consolidate towards 0 or 1.

Finally, the total loss function $\mathcal{L}_{\textbf{ProvX}}$ is the sum of the primary counterfactual loss and the weighted solidification loss:

$$\mathcal{L}_{\text{ProvX}} = \mathcal{L}_{\text{CFX}} + \gamma_S \cdot \mathcal{L}_S, \tag{9}$$

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

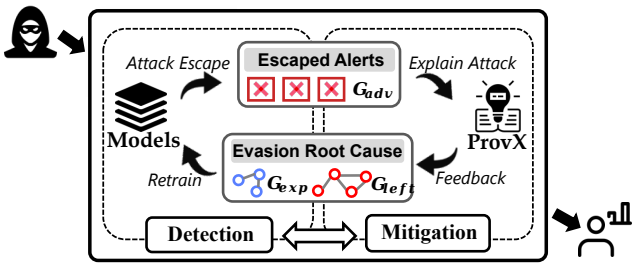


Figure 5: Mitigation-driven feedback loop for detection enhancement.

4.3 Choke Point Localization

In this phase, we transform the optimization results from the blocking strategy algorithm (4.2.2) into a core edge list that security analysts can prioritize for review and possible enforcement.

After the optimization process is complete, we obtain a final continuous event mask, $\sigma(\hat{M}_k^*)$. By calculating $(1-\sigma(\hat{M}_k^*))$, we derive a blocking priority score for each edge in the alert graph. A high score indicates that the edge is highly necessary for the detector's malicious prediction under the learned counterfactual objective. Therefore, high-scoring edges represent core behavioral links that analysts should prioritize for review and candidate blocking. This score reflects model-level decision necessity rather than the operational ease or safety of enforcing a host-level action.

We select the top-K edges with the highest blocking priority score. In our framework, K is a user-definable response budget. This budget allows analysts, based on their real-world operational pressure, to trade off between response cost and the completeness of alert refinement. For example, an analyst facing alert fatigue can set a low K to obtain the most compact list of candidate choke points. Our objective is to flip the detector's alert with the smallest simulated intervention, thereby producing a concise edge list for downstream review.

Finally, we extract the subgraph $\tilde{G}_k^*$ formed by these top-K core edges. This subgraph is the blocking-oriented core edge list output by ProvX. For model-level evaluation, we instantiate the simulated intervention by removing the selected edges and suppressing the features of their incident endpoint nodes. This joint structural-attribute intervention reflects both the disruption of selected event correlations and the removal of their local behavioral evidence from the detector's input. Security analysts can interpret the result as: *"Under the detector's current decision logic, this alert is most sensitive to the following K critical edges; these edges should be prioritized for review and possible blocking."*

5 Detection-Mitigation Feedback Loop

Although ProvX primarily analyzes malicious alerts for blocking-oriented refinement, the same counterfactual logic can also be used in reverse for false-negative analysis. When a malicious activity deceives a PIDS through camouflage or obfuscation [12] and is misclassified as benign, we refer to it as an adversarial evasion. In this setting, the question is no longer "which structures make the detector predict malicious?", but rather "which added or surrounding structures make the detector maintain a benign prediction for a known malicious behavior?"

We therefore explore a preliminary feedback framework guided by security analysts. Let G_m denote an original malicious provenance subgraph. Following the mimicry attack setting in [12], we construct an adversarial subgraph G_{adv} by injecting benign-looking provenance structures into G_m and retain the samples that evade the detector. For such a false-negative sample, ProvX is assigned a reversed objective: it searches for a minimal structure G_{exp} whose removal flips the detector's prediction from benign back to malicious. We interpret G_{exp} as candidate evasion-associated context, rather than as definitive proof of attacker intent.

After analyst verification, we remove the localized edges in G_{exp} from G_{adv} and denote the remaining residual graph as G_{left}. We then use both G_{adv} and G_{left} as malicious training examples. These two samples play different roles. G_{adv} teaches the detector the complete obfuscated attack pattern, while G_{left} preserves the residual malicious behavior after removing the localized evasion-associated context. This paired feedback is intended to improve robustness against similar false negatives while keeping analyst validation in the loop.

Analyst verification is important because the localized G_{exp} may contain ambiguous benign context or low-confidence structures rather than a definitive evasion cause. We therefore do not directly treat every localized structure as an automatic training signal. Instead, analyst verification filters unreliable feedback and reduces the risk of increasing false positives during retraining. In addition, using the full adversarial graph G_{adv} together with the residual graph G_{left} avoids training only on a dismantled subgraph: G_{adv} exposes the detector to the complete obfuscated attack, while G_{left} reinforces the remaining attack core. This design aims to reduce false negatives against similar evasion patterns without unnecessarily weakening the detector's boundary between benign and malicious behavior.

We present this feedback loop as a preliminary extension of ProvX, not as a fully general adversarial-training framework. Its effectiveness is evaluated in Section 6.7.

6 Evaluation

In this section, we evaluate the effectiveness of ProvX. Our goal is to answer the following research questions:

- **RQ1**. **Localization Ability:** How accurately does PROVX's core edge list hit real-world attack steps?
- **RQ2**. **Blocking Effectiveness:** What is the blocking capability of the core edges localized by PROVX?
- **RQ3**. **SecOps Cost-Benefit Analysis:** How does PROVX's *Cost* vs. *Benefit* trade-off compare to SOTA PIDS?
- **RQ4**. **Hyperparameter Sensitivity:** What is the impact of key hyperparameters on PROVX's mitigation performance?
- **RQ5**. **System Overhead:** What is the computational overhead of the PROVX framework?
- **RQ6**. **Adversarial Analysis:** How can PROVX be used to analyze and defend against adversarial evasion attacks?

6.1 Experimental Setup

6.1.1 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)[2]. The DARPA TC dataset [3] 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 [22], we select the **Cadets**, **Theia** and **Trace** scenarios. The DARPA OpTC dataset [2] provides a broad view of benign and malicious audit records from approximately 1,000 hosts. 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*.

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 during detection.

Table 2: DARPA OpTC detection performance on three GNNs.

[2]Existing PIDS based on provenance graphs all use one [46] or more [8, 43] of these datasets to verify APT detection performance. Our work refers to [22, 33, 34] and selects the above datasets.

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

Table 3: DARPA E3 detection performance on three GNNs.

6.1.2 Implementation Details

We detail the implementation of our framework's two main components: **Phase I: Threat Detection**. We implemented subgraph-level detection models using GCN, GAT, and GraphSAGE. Each model uses a 2-layer GNN architecture with ReLU/Dropout, followed by an average pooling layer and a 2-layer MLP classifier. These detectors were trained for 50 epochs using an Adam optimizer with a 0.001 learning rate. As shown in Tables 2 and 3, the models achieve high performance, providing a solid and reliable foundation for our Phase II task. **Phase II: Attack Mitigation**. Each triggered alert subgraph is processed independently. We train the core edge mask for 200 epochs with a learning rate of 0.01. Key hyperparameters for the Staged Solidification strategy were set as follows: solidification ratio $R_S = 0.6$, factor $\gamma_S = 0.5$, and confidence bounds $\tau_{low} = 0.05$ and $\tau_{high} = 0.95$. A detailed analysis of the trade-off parameter α and the response budget K is presented in Section 6.5.

All experiments were performed on a server running Ubuntu 22.04.5 LTS with an Intel(R) Xeon(R) Silver 4208 CPU (14 cores, 14 threads, base frequency 2.10 GHz) and 256 GB of RAM.

6.1.3 Baselines for Comparison

Our evaluation is designed to validate PROVX's performance in the novel task of attack mitigation. This requires a two-pronged comparison, as no single baseline fully addresses this challenge. First, to validate the algorithmic superiority of our localization ability(**RQ1**), we compare our counterfactual-based algorithm against established triage tools and SOTA explainable artificial intelligence (XAI) explainers. Second, to validate the mitigation ability(**RQ2**), we compare with XAI explainers. These XAI explainers are fact-based, including perturbation-based, decomposition-based, and gradient-based methods [26, 35, 36, 38, 44, 45]. Unlike these methods, which primarily seek subgraphs that preserve or explain the original prediction, PROVX searches for a minimal intervention that changes the detector's prediction and ranks edges by blocking-oriented necessity. And, to demonstrate our framework's operational value and SecOps advantage (**RQ3**), we

GNN Type	Metrics[1]	DARPA OpTC	DARPA TC E3		
			Cadets	Theia	Trace
GCN	Acc	0.8402	0.9643	0.9375	1.0000
	Pr	0.1009	0.9024	0.8934	0.9747
	Rec	0.7573	0.1470	0.1182	0.1226
	F1	0.1683	0.2517	0.1970	0.1982
GAT	Acc	0.9148	0.9643	0.9688	1.0000
	Pr	0.1293	0.9125	0.9033	1.0000
	Rec	0.8169	0.2249	0.2244	0.2758
	F1	0.2118	0.3101	0.3342	0.3524
GraphSAGE	Acc	0.9033	1.0000	1.0000	1.0000
	Pr	0.1306	0.9798	0.9431	0.9511
	Rec	0.7035	0.3068	0.2914	0.3318
	F1	0.2006	0.4531	0.3826	0.4773

[1] **ProvX** uses different security model, so the number of malicious subgraphs captured is different.

Table 4: Localization Ability of **ProvX** in DARPA OpTC(with K=10) and TC-E3(with K=30) on GCN, GAT, and GraphSAGE.

compare our cost-benefit output against raw alerts produced by SOTA PIDS detectors. This group includes prominent systems such as MAGIC [22], SLOT [33], FLASH [34], SHADEWATCHER [46], and THREATRACE [43].

6.2 Localization Ability *(RQ1)*

The core output of the PROVX framework is an actionable core edge list. A critical operational question is: to what extent does this list hit the real-world attack steps? Therefore, our first research question evaluates the localization ability of PROVX. We use the collection of malicious labels from existing works [22, 34, 43] as the ground truth for evaluation. After PROVX analyze an alert subgraph, we use Accuracy, Precision, Recall and F1-score as evaluation metrics for localization hit ability.

Specifically, if an edge within the Top-K core edge list provided by PROVX corresponds to a real attack, we consider it a Hit. Accuracy is defined as the proportion of malicious alert graphs in the test set for which PROVX achieves at least one such hit. 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 F1-score is computed per alert subgraph and then averaged across alert subgraphs.

Table 4 presents the evaluation results of PROVX across all datasets and GNN models, while Table 5 shows the comparison between PROVX and the baseline localization algorithms. 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. We found that the prevalence of star structures (such as those related to `Nginx`) in the DARPA TC dataset significantly affects the learning trend

GNNs	Explainers[1]	Acc	Pr	Rec	F1
GCN	GradCam	0.2165 ▼74.2%	0.0944 ▼6.4%	0.2473 ▼67.3%	0.1418 ▼15.7%
	DeepLIFT	0.2639 ▼68.6%	0.0355 ▼64.8%	0.1832 ▼75.8%	0.0730 ▼56.6%
	GNN-LRP	0.2526 ▼69.9%	0.0891 ▼11.7%	0.1774 ▼76.6%	0.1186 ▼29.5%
	PGExplainer	<u>0.7938</u> ▼5.5%	0.0978 ▼3.1%	0.7147 ▼5.6%	0.1426 ▼15.3%
	SubgraphX	0.7783 ▼7.4%	<u>0.1019</u> ▲1.0%	0.6413 ▼15.3%	0.1011 ▼39.9%
	GNNExplainer	0.8402 0.0%	**0.1025** ▲1.6%	<u>0.7492</u> ▼1.1%	**0.1698** ▲0.9%
	ProvX	**0.8402** ▲60.3%[2]	0.1009 ▲16.2%	**0.7573** ▲67.5%	<u>0.1683</u> ▲35.2%
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%	<u>0.1188</u> ▼8.1%	<u>0.8007</u> ▼1.9%	<u>0.2021</u> ▼5.1%
	SubgraphX	<u>0.9015</u> ▼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%
Graph SAGE	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	0.9033 0.0%	<u>0.1298</u> ▼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	<u>0.9033</u> ▲11.7%	**0.1306** ▲27.1%	<u>0.7035</u> ▲8.8%	<u>0.2006</u> ▲24.5%

- **Bold** denotes the best results, and <u>underline</u> denotes the second-best results. ▼% represents the percentage of reduction, ▲% represents the percentage of improvement.

[1] We adopt the hyperparameter configuration of previous work [20] to set up the baselines explainers.

[2] Improvement compared to the average of all detection systems except PROVX.

Table 5: Comparison of **ProvX** and XAI Baselines' performance in explaining attack correlation on three GNNs, in DARPA OpTC.

of the detection model. The K key edges found by PROVX under the given response budget cannot fully cover the numerous interactions of the hub nodes. The attack distribution in OpTC is more scattered and complex; PROVX can obtain multiple important links of the scattered attack links, but these important edges may be crucial to triggering the alert (model prediction), rather than the real steps annotated by experts.

In the comparative experiments (Table 5), PROVX outperforms the baseline algorithms in the vast majority of cases, demonstrating the effectiveness of counterfactual logic in the APT alert refinement domain. Among the baselines, the perturbation-based methods (GNNExplainer, PGExplainer, SubgraphX) perform slightly better than the decomposition-based or gradient-based methods, but PROVX's localization accuracy remains consistently ahead. The main reason is the difference in optimization objective. Fact-based explainers tend to identify edges that are salient or correlated with the existing malicious prediction, which can include benign but highly connected background structures in noisy provenance graphs. In contrast, PROVX explicitly optimizes for a minimal intervention that flips the detector's prediction, making the selected edges more aligned with model-level necessity.

Method	Accuracy	Precision	Recall	F1-score
NoDoze [17]	0.8929	0.8595	0.2661	0.3994
DepImpact [10]	0.9643	0.9472	**0.3643**	0.4926
ProvX (Ours)	**1.0000**	**0.9798**	0.3068	**0.4531**

- Results for NoDoze and DepImpact are aligned to the same budget $K = 30$ for structural comparison. ProvX results are based on the GraphSAGE backbone.

Table 6: Comparison of **ProvX** against Security Triage Baselines (NoDoze and DepImpact) in DARPA TC-E3 Scenarios.

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

Table 7: Mitigation Efficacy of DARPA OpTC and TC-E3 on GNNs, with K=10 & 30.

In addition, staged solidification stabilizes mask learning by locking high-confidence retain/remove trends in later epochs, which reduces oscillation around high-degree provenance hubs.

Compared to established security triage baselines(Table 6) and under an equivalent alert reduction budget, ProvX significantly outperforms NoDoze and DepImpact in localization certainty, achieving a perfect Accuracy of 1.0000 and a Precision of 0.9798. Unlike traditional tools that provide a broader but less certain list of candidates—often leading to further analyst fatigue —ProvX 's interventional reasoning provides a definitive and concise blocking strategy. This accuracy is paramount for automated response, where "cutting the wrong wire" can be as damaging as the attack itself.

6.3 Blocking Effectiveness *(RQ2)*

RQ1 validates the localization ability of our framework, i.e., how well the edges localized by ProvX correlate with real-world attack steps. However, from a security operations perspective, a more critical question remains: are these localized edges necessary for the detector's original alert? An edge that overlaps with attack ground truth is not necessarily the decisive evidence used by the GNN detector. Therefore, we further evaluate whether intervening on the top-K core edge list is sufficient to flip the original detector prediction. Unlike methods that use Fidelity+ with surrogate models [29], we perform this validation directly on the fixed GNN detector. Inspired by concepts from causal inference [9, 31, 39, 40], we introduce the **Mitigation Efficacy Rate (MER)** to measure the model-level blocking efficacy of the core edge list.

Let an alert subgraph be $G_k = (A_k, X_k)$, where A_k is the adjacency matrix and X_k is the node feature matrix. Given

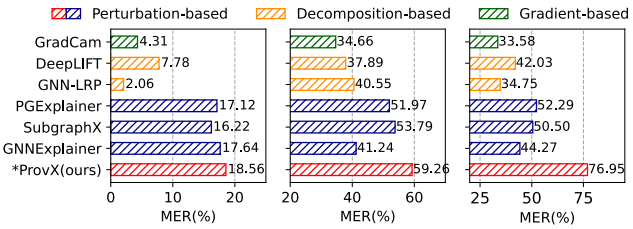


Figure 6: ProvX 's MER(%) performance on DARPA OpTC, from left to right are the results of **GCN**, **GAT**, and **GraphSAGE**. ProvX's component is also a perturbation-based scheme, but is the only counterfactual-based explainer among these works.

the top-K core edge set $\tilde{G}_k^*$ output by ProvX, we define a simulated intervention operator $I_K(\cdot)$ that removes these selected edges from A_k and suppresses the features of their incident endpoint nodes in X_k. The resulting intervened graph is then fed back into the same detector without retraining. MER is defined as follows:

$$\text{MER} = \frac{1}{N}\sum_{k=1}^{N} p_k, \quad p_k = \begin{cases} 1, & \text{if } f(I_K(G_k)) \neq f(G_k), \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Here, $f(G_k)$ is the detector's original prediction on the alert subgraph, and $f(I_K(G_k))$ is the prediction after applying the simulated structural-attribute intervention. A high MER indicates that the selected edge set and its local endpoint evidence are necessary for the detector to maintain its malicious judgment. MER is therefore a model-level metric: it evaluates whether the recommended core edges can flip the detector's alert under a controlled intervention, rather than proving that the corresponding host-level enforcement action is always safe or sufficient in deployment.

Table 7 presents the MER results of ProvX on all datasets, while Figure 6 compares ProvX with baseline algorithms under the same intervention protocol. We observe that, with $K = 10$ ($K = 30$ for DARPA TC E3), ProvX achieves an MER greater than 18% on all datasets, even achieves average MER of 51.59% in OpTC. In other words, on the test set, the top-K core edge list localized by ProvX flips over half of the original detector alerts under the simulated intervention. Compared with fact-based baselines, ProvX achieves higher MER across the three GNN backbones because it directly optimizes for a minimal counterfactual intervention that changes the detector's prediction, rather than only identifying subgraphs correlated with the original prediction.

6.4 Cost-Benefit Analysis *(RQ3)*

The core motivation for our work is the alert fatigue crisis, characterized by voluminous, coarse-grained alerts that lack the prescriptive detail needed for response. While SOTA PIDS

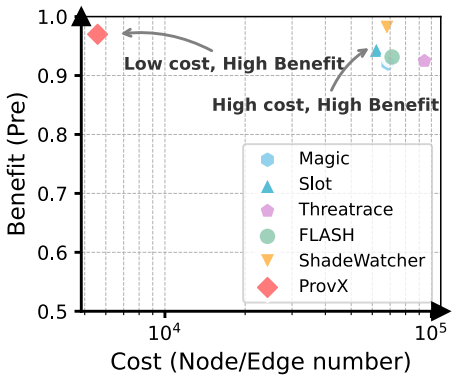


Figure 7: Cost-Benefit analysis of PROVX against SOTA PIDS [22, 33, 34, 43, 46] in DARPA E3 Trace. The X-axis represents the **Operational Cost** (average entities an analyst must review per alert). The Y-axis represents the **Benefit**. PROVX is the only framework in the ideal *Low Cost, High Benefit* quadrant.

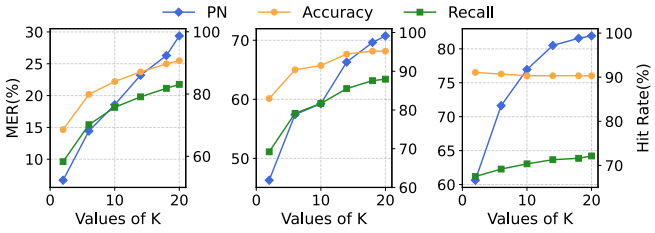


Figure 8: PROVX's MER(%) (left y-axis) and Hit Rate(%) (right y-axis) as K varies, from left to right are the results of GCN, GAT and GraphSAGE.

may correctly flag a subgraph as malicious, they often output tens of thousands of unprioritized system entities, imposing an immense review burden on analysts. In the context of defense, the critical trade-off is between Mitigation Precision (the safety of the intervention) and Operational Response Cost (the analyst's effort).

This RQ evaluates PROVX's ability to provide high-precision blocking strategies at a minimal cost. To conduct a fair comparison across systems with different output granularities, we define two metrics: Mitigation Precision (Y-axis): The proportion of real attack steps within the output list. High precision is paramount for operational safety, ensuring that blocking actions do not inadvertently disrupt benign business processes. Operational Response Cost (X-axis): The average number of entities (nodes or edges) an analyst must review to execute a blocking action, plotted on a logarithmic scale. For PROVX, this cost is fixed at $K = 30$ edges, representing a targeted intervention.

The results are shown in Figure 7. We observe a stark contrast between SOTA PIDS and PROVX. Raw PIDS alerts reside in a High Cost, Low Precision category regarding specific entity-level actions; while they correctly identify the presence of a threat, their massive output (62,125 to 95,000 entities) is saturated with benign background noise, making the precision of any individual blocking action near zero without manual triage. Investigating such a volume would take a 10-analyst team roughly 6.9 to 11.2 months per alert, which is operationally impossible for active defense. In contrast, PROVX resides in the ideal Low Cost, High Benefit quadrant. It achieves a near-perfect Mitigation Precision of 97.98% while reducing the operational cost by over 89% compared to investigating raw alerts.

6.5 Hyperparameter Investigation *(RQ4)*

We investigate the impact of several different hyperparameters on the explanation performance of PROVX:

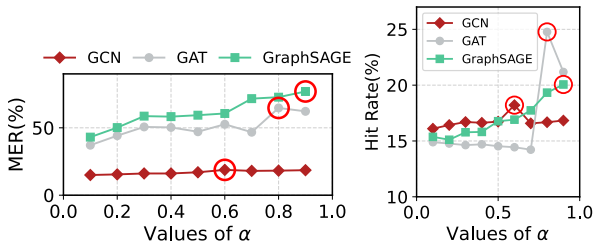
Response Budget K: K defines the number of edges included in the final core edge list, making it a direct proxy for the operational cost of analysis. We evaluate K on the OpTC dataset, varying its value from 1 to 20 (Figure 8). Increasing the K budget directly improves the MER. On GraphSAGE, when $K = 20$, the MER can reach 81.91%. However, K should not be excessively large, as this would diminish the significance of mitigation; removing a large portion of a graph will obviously cause a prediction flip.

Counterfactual Trade-off Parameter α: The parameter α balances the predict flip loss ($\mathcal{L}_{\textbf{Pred}}$) and the mask distance Loss ($\mathcal{L}_{\textbf{dist}}$). The former aims to neutralize the alert, while the latter aims to minimize the blocking cost. The experimental results are shown in Figure 9. Figure 9(a) displays the change in MER with respect to α, and Figure 9(b) shows the change in F1-score. On the three GNN detectors, the optimal α values are 0.6, 0.8, and 0.9, respectively, validated by both MER and F1-score. When α exceeds the optimal point, both metrics decline.

Solidification Start Ratio T_{start} & Penalty Strength γ_S: Solidifying the localization results is one of the core features of PROVX. The T_{start} ratio determines when to activate the mechanism, while the γ_S strength determines the penalty for deviating from already-confident judgments. Figure 10 shows the analysis. Figure 10(a) shows the trend of MER as T_{start} varies. A value of 0.6 provides the best balance, allowing sufficient exploratory time before the penalty phase begins. Figure 10(b) shows the trend of MER as γ_S varies. When γ_S is 0 (no penalty), performance drops significantly. The best performance is achieved when γ_S is 0.7, 0.6, and 0.6 for GCN, GAT, and GraphSAGE, respectively.

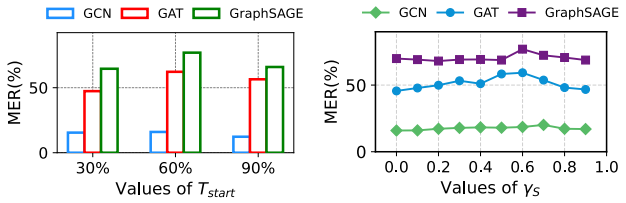
6.6 System Overhead *(RQ5)*

PROVX is an defense framework whose workflow encompasses two phase. We evaluate its performance across three GNN backbones in a CPU-only environment, reflecting re-



(a) Mitigation Efficacy Rate (b) F1-score

Figure 9: MER and F_1-score of PROVX when the α changes.



(a) Solidification ratio study (while $\gamma_S = 0.6$) (b) Solidification factor study (while $T_{start} = 0.6$)

Figure 10: MER of PROVX when the T_{start} and γ_S changes.

alistic SecOps scenarios where dedicated GPU support for forensic analysis is often unavailable. As shown in Table 8, the detection training in Phase I is highly efficient. For Phase II, the average peak memory remains remarkably low at 1.43 GB. Critically, once the strategy is synthesized, the final mitigation evaluation (identifying the specific choke points) is completed in just 3-4 seconds.

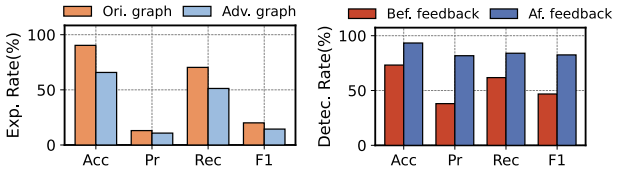
Overall, PROVX demonstrates high operational feasibility with minimal hardware requirements. We argue that the minute-level synthesis time is a justifiable trade-off for high-precision, analyst-facing blocking candidates, especially when compared to the hours or days an analyst would otherwise spend manually investigating voluminous raw alerts. By reducing the computational barrier, PROVX allows security teams to rapidly deploy and iterate on defense strategies within existing standard server architectures, providing a practical pathway for timely alert analysis and blocking-oriented response.

6.7 Adversarial Analysis *(RQ6)*

In this section, we provide a preliminary evaluation of the feedback idea described in Section 5. Starting from existing malicious provenance subgraphs, we inject benign-looking structures following the mimicry attack setting of [12] and retain the generated samples that evade the detector as G_{adv}. We then apply PROVX with the reversed objective: instead of finding structures whose removal flips a malicious prediction to benign, it finds structures whose removal flips the false-negative prediction from benign back to malicious.

Phase		Time consumption (s)			Peak Memory consumption (MB)		
		GCN	GAT	Graph SAGE	GCN	GAT	Graph SAGE
Generation		113	174	140	1377	1405	1379
Mitigation	Training	733	1196	994	1491	1519	1391
	Evaluating	3	4	4	1541	1561	1461

Table 8: Overhead analysis of the PROVX framework, detailing the **Phase I (Detection)** and **Phase II (Mitigation)** components.



(a) Localization performance before and after adding adversarial structure. (b) Detection performance before and after feedback.

Figure 11: Demonstration of the **Feedback Loop**. (a) PROVX's **localization performance** is impacted by adversarial structures. (b) However, feeding these **localized analysis results** back for retraining significantly enhances the detector's adversarial robustness.

The localized structure is treated as candidate evasion-associated context. After this step, we construct G_{left} by removing the localized structure from G_{adv}. We feed both G_{adv} and G_{left}, with malicious labels, back into detector training. We then report two effects: (1) how adversarial camouflage affects PROVX's localization performance, and (2) how feedback changes the detector's performance on adversarial samples, as shown in Figure 11(a) and Figure 11(b).

As shown in Figure 11(a), adding adversarial structures decreases localization performance because the injected benign-looking context changes the detector's decision boundary and weakens the visibility of the attack core. Nevertheless, PROVX still identifies candidate structures associated with the evasion. After feedback training, the detector's performance on adversarial samples improves: Accuracy, Precision, Recall, and F1 improve by 20.1%, 43.7%, 22.2%, and 35.5%, respectively (Figure 11(b)). These results suggest that PROVX's localized feedback can provide useful training signals for detector hardening, while a complete feedback framework remains future work.

7 Discussion & Future Work

Positioning and Industrial Feasibility. PROVX is designed as a forensic decision-support tool rather than a real-time, in-line enforcement system. Its optimization is intended to operate post-detection, transforming voluminous, unaction-

able alerts into analyst-reviewable candidate choke points. In practical SecOps, the bottleneck is often not only sub-second detection latency, but the long manual effort required to investigate complex attack chains. By providing compact blocking-oriented guidance in minutes, PROVX helps bridge the gap between threat discovery and response planning. However, mapping a provenance edge to a safe host-level action, such as process termination, file quarantine, syscall blocking, or network restriction, remains deployment-specific and must account for collateral cost and business context.

Alert-Subgraph Acquisition. PROVX assumes that an alert-associated provenance subgraph is available as input. In our experiments, we approximate this SOC input by using a Louvain-based procedure to obtain evaluation subgraphs from DARPA provenance graphs, followed by a GNN-based detector to identify malicious alerts. This setting enables controlled evaluation on public datasets, but it does not fully reproduce how alerts are generated, correlated, and expanded in a live SOC. In deployment, the input subgraph may come from mature PIDS, EDR/SIEM alarms, network indicators mapped to host events, or analyst-guided provenance expansion. We also observe that the quality of this upstream subgraph acquisition can noticeably affect downstream blocking-oriented refinement. If a subgraph is too small, the relevant attack dependency may be split across windows or communities; if it is too large, benign background behavior may dilute the structural signal and make core-edge localization harder. Thus, obtaining alert subgraphs with stronger structural discriminability is an important direction for improving PROVX. Beyond the Louvain-based instantiation used in this paper, preliminary exploration with time-window-based subgraphs and ego-network expansion around suspicious seed entities also produced meaningful inputs for the blocking module. We therefore view Phase I as a replaceable alert-acquisition layer: the current counterfactual blocking module can benefit from more mature SOC subgraph construction strategies, and designing such strategies is part of our future work.

Applicability to Non-Differentiable PIDS. The current design of PROVX is tailored for explaining GNNs, which are (typically) differentiable models. Our methodology, which may rely on probing model-internal states or structural sensitivities, does not directly apply to the wide array of non-differentiable PIDS currently in use, such as those based on decision forests, isolation forests, or complex rule-based expert systems. This limits the generalizability of PROVX as a universal explainer for all provenance-based detection. To address the aforementioned limitation, a key future direction is to adapt PROVX into a truly model-agnostic framework. We plan to explore techniques inspired by LIME or SHAP, which treat the detector as a black box and query it to build a local, interpretable approximation. This would allow PROVX to be integrated as a modular explanation component for a much wider range of existing PIDS.

Formalizing the Alert-Feedback Loop. In the main text, we present the feedback loop as a preliminary extension of PROVX. A complete deployment-ready feedback framework still requires further formalization and implementation. Future work should address several key questions: how analyst feedback should be encoded, how feedback samples should be weighted during retraining or fine-tuning, and how the detector can benefit from new adversarial examples without catastrophic forgetting. Exploring active learning or reinforcement learning from human feedback may be a promising direction for systematically improving both detector robustness and explanation quality.

8 Conclusion

We propose PROVX, an active APT defense framework designed to bridge the critical operational gap between threat detection and mitigation. PROVX employs an interventional reasoning technique based on counterfactual logic. This method reformulates the search for critical attack edges into a continuous, differentiable optimization task. By maximizing a prediction flip loss while minimizing a mask distance Loss, and incorporating a novel staged solidification strategy, PROVX effectively isolates the minimal structural subset—the "fuses" that are most necessary for the detector's malicious judgment and should be prioritized for blocking-oriented response.

Evaluations on real-world APT datasets demonstrate that PROVX achieves good performance, ensuring that blocking recommendations are both highly relevant to actual attack steps and safe for system continuity. This work also validates a preliminary closed-loop feedback framework, positioning PROVX as a forensic decision-support tool that helps make existing PIDS outputs more actionable while reducing analyst burden.

Acknowledgment

We thank anonymous reviewers for their valuable comments. This work was supported in part by the National Key Research and Development Program of China (2023YFB2904000), the National Natural Science Foundation of China under Grant (No. 62272370), the Natural Science Basic Research Program of Shaanxi (No. 2025JC-JCQN-073), China Higher Education Institutions Industry-University-Research Innovation Fund (Project No. 2024IT095), Qinchuangyuan Scientist + Engineer Team Program of Shaanxi (No. 2024QCY-KXJ-149), Songshan Laboratory (No. 241110210200), Fundamental Research Funds for the Central Universities (No. YJSJ26003), National Research Foundation, Singapore, and DSO National Laboratories under the AI Singapore Programme (AISG Award No. AISG2-GC-2023-008), the National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN), the National Research Foundation, Prime Minis-

ter's Office, Singapore under Campus for Research Excellence and Technological Enterprise (CREATE) programme, Ripple under University Blockchain Research Initiative (UBRI) [11], the National Natural Science Foundation of China under Grant 62402202, and project ZR2024QF036 supported by Shandong Provincial Natural Science Foundation.

Ethical Considerations

This section examines the ethical implications of PROVX by identifying affected stakeholders, assessing potential benefits and harms, outlining safeguards for responsible deployment, and explaining our decision to conduct and publish this research.

Stakeholders. The stakeholders affected by this research include system administrators and SOC analysts who may deploy PROVX, end users whose activities may be captured in provenance logs, organizations and system owners protected by PIDS, security tool vendors who may adopt similar techniques, the broader security research community, adversaries whose attack strategies may be disrupted or informed by this work, and society at large, including users of critical infrastructure, financial systems, and government services targeted by APT campaigns.

Positive Impacts. PROVX is designed to improve APT defense by moving beyond detection-only alerts toward *analyst-reviewable blocking guidance*. For SOC analysts, PROVX can reduce the cognitive burden of incident response by identifying a compact set of provenance edges that are important to the detector's malicious judgment. For organizations, this can support faster containment and reduce adversary dwell time. For end users and the public, stronger APT defenses can help protect the confidentiality, integrity, and availability of systems that process sensitive data and support critical services. For the research community, this work contributes a counterfactual reasoning framework and the Mitigation Efficacy Rate (MER) metric for evaluating *model-level blocking efficacy*.

Potential Negative Impacts. Despite its defensive intent, PROVX may introduce several risks if deployed carelessly. First, acting on blocking recommendations in production may disrupt benign activity. A false positive mitigation could terminate critical processes, sever legitimate network connections, or block access to essential files, causing service outages, lost work, or cascading operational failures. Second, because PROVX prioritizes a subset of alert edges, analysts may overfocus on the recommended core edge list and overlook genuine threats outside the prioritized set. This could introduce new false negatives at the triage stage, even when the underlying PIDS has detected the broader suspicious activity. Third, PROVX's counterfactual reasoning pipeline introduces processing overhead, which may limit usefulness in time-critical response settings or large-scale deployments. Fourth, provenance logging systems such as Linux Audit, ETW, and CamFlow collect fine-grained process, file, and network activity. Such logs may raise privacy concerns if repurposed for employee surveillance or other non-security uses. Finally, the counterfactual reasoning methodology has *dual-use potential*: an adversary with access to comparable provenance data and detector behavior might use similar reasoning to make attacks more resilient to edge-level mitigations.

Mitigations. We mitigate these risks in several ways. Our evaluation uses only publicly available DARPA TC and OpTC benchmark datasets collected in controlled environments; we do not collect, store, or analyze real user data or personally identifiable information. For real-world deployment, organizations should apply *data minimization*, log rotation, aggregation or anonymization where feasible, clear retention limits, and access controls for provenance logs. Operationally, PROVX should be deployed as a *decision-support tool* rather than a fully automated response system. Its recommendations should be reviewed by human analysts before any production mitigation is executed, and automated enforcement should first be evaluated in dry-run or shadow mode. To reduce false-negative risk from triage, organizations should preserve the full PIDS alert corpus alongside PROVX's prioritized subset and periodically review deprioritized alerts. Future work will also investigate streaming and incremental computation to reduce latency. We acknowledge the dual-use nature of this research, but note that PROVX primarily benefits defenders because it assumes access to detailed provenance logs and a trained detection model, which are typically available to the defending organization.

Unmitigated Risks. Some risks cannot be fully eliminated. Provenance logging infrastructure may still be repurposed for surveillance despite our recommendations. PROVX's prioritization may still cause analysts to miss threats outside the recommended core edge list. Counterfactual reasoning may remain too slow for some time-critical scenarios. Publication of the method may also inform adversarial evasion strategies. We consider these risks inherent to security research that produces *dual-use knowledge*.

Decision to Conduct and Publish. Our decision to conduct and publish this research is guided by *beneficence*, *respect for persons*, *justice*, and *respect for law and public interest*. APT attacks cause substantial harm to governments, enterprises, and individuals, while current incident response remains heavily manual and time-consuming. PROVX aims to improve this situation by providing analysts with actionable, human-reviewed guidance. The study does not involve human subjects and relies only on public benchmark datasets, so no informed consent issue arises in our experiments. For deployment, however, organizations should notify users about provenance logging and establish transparent data governance policies. We also recognize that organizations need provenance logging infrastructure and GNN-based PIDS to benefit from PROVX, which may limit accessibility for some resource-constrained organizations. We comply with applicable dataset terms of use. Overall, we believe the defensive value of im-

proving APT response outweighs the residual dual-use and operational risks, particularly because open publication enables safer follow-on work.

Open Science

To support future research, we provide an accompanying artifact repository [4] for ProvX. The artifact contains the code implementation, including the GNN detector, the ProvX interventional mask learning module, the staged solidification procedure, and evaluation utilities.

References

[1] Adversarial tactics, techniques and common knowledge. https://attack.mitre.org/wiki/MainPage.

[2] Darpa operationally transparent cyber (optc) data release. https://github.com/FiveDirections/OpTC-data. 2019.

[3] Darpa transparent computing program engagement 3 data release. https://github.com/darpa-i2o/Transparent-Computing. 2020.

[4] ProvX Artifact. URL: https://zenodo.org/records/20310415, doi:10.5281/zenodo.20310415.

[5] Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. {ATLAS}: A sequence-based learning approach for attack investigation. In *30th USENIX security symposium (USENIX security 21)*, pages 3005–3022, 2021.

[6] Adam Bates, Dave Jing Tian, Kevin RB Butler, and Thomas Moyer. Trustworthy {Whole-System} provenance for the linux kernel. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 319–334, 2015.

[7] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[8] Zijun Cheng, Qiujian Lv, Jinyuan Liang, Yan Wang, Degang Sun, Thomas Pasquier, and Xueyuan Han. Kairos: Practical intrusion detection and investigation using whole-system provenance. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3533–3551. IEEE, 2024.

[9] Zhaoyang Chu, Yao Wan, Qian Li, Yang Wu, Hongyu Zhang, Yulei Sui, Guandong Xu, and Hai Jin. Graph neural networks for vulnerability detection: A counterfactual explanation. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 389–401, 2024.

[10] Pengcheng Fang, Peng Gao, Changlin Liu, Erman Ayday, Kangkook Jee, Ting Wang, Yanfang Fanny Ye, Zhuotao Liu, and Xusheng Xiao. {Back-Propagating} system dependency impact for attack investigation. In *31st USENIX security symposium (USENIX Security 22)*, pages 2461–2478, 2022.

[11] Yebo Feng, Jiahua Xu, and Lauren Weymouth. University blockchain research initiative (ubri): Boosting blockchain education and research. *IEEE Potentials*, 41(6):19–25, 2022.

[12] Akul Goyal, Xueyuan Han, Gang Wang, and Adam 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.

[13] Akul Goyal, Gang Wang, and Adam Bates. R-caid: Embedding root cause analysis within provenance-based intrusion detection. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3515–3532. IEEE, 2024.

[14] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. Unicorn: Runtime provenance-based detector for advanced persistent threats. In *In Network and Distributed System Security Symposium (NDSS'20).*, 2020.

[15] Xueyuan Han, Xiao Yu, Thomas Pasquier, Ding Li, Junghwan Rhee, James Mickens, Margo Seltzer, and Haifeng Chen. {SIGL}: Securing software installations through deep graph learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2345–2362, 2021.

[16] Wajih Ul Hassan, Adam Bates, and Daniel Marino. Tactical provenance analysis for endpoint detection and response systems. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1172–1189. IEEE, 2020.

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

[18] Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Dawei Wang, Zhengzhang Chen, Zhichun Li, Junghwan Rhee, Jiaping 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*, pages 165–178, 2020.

[19] Md Nahid Hossain, Sadegh M Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R Sekar, Scott Stoller, and VN Venkatakrishnan. Sleuth: Real-time attack scenario reconstruction from cots audit data. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 487–504, 2017.

[20] Yutao Hu, Suyuan Wang, Wenke Li, Junru Peng, Yueming Wu, Deqing Zou, and Hai 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*, pages 1407–1419, 2023.

[21] Muhammad Adil Inam, Yinfang Chen, Akul Goyal, Jason Liu, Jaron Mink, Noor Michael, Sneha Gaur, Adam Bates, and Wajih Ul Hassan. Sok: History is a vast early warning system: Auditing the provenance of system intrusions. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2620–2638. IEEE, 2023.

[22] Zian Jia, Yun Xiong, Yuhong Nan, Yao Zhang, Jinjing Zhao, and Mi Wen. {MAGIC}: Detecting advanced persistent threats via masked graph representation learning. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5197–5214, 2024.

[23] Baoxiang Jiang, Tristan Bilot, Nour El Madhoun, Khaldoun Al Agha, Anis Zouaoui, Shahrear Iqbal, Xueyuan Han, and Thomas Pasquier. Orthrus: Achieving high quality of attribution in provenance-based intrusion detection systems. In *Security Symposium (USENIX Sec'25). USENIX*, 2025.

[24] Zitong Li, Xiang Cheng, Lixiao Sun, Ji Zhang, and Bing Chen. A hierarchical approach for advanced persistent threat detection with attention-based graph neural networks. *Security and Communication Networks*, 2021:1–14, 2021.

[25] Yushan Liu, Mu Zhang, Ding Li, Kangkook Jee, Zhichun Li, Zhenyu Wu, Junghwan Rhee, and Prateek Mittal. Towards a timely causality analysis for enterprise security. In *NDSS*, 2018.

[26] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020.

[27] Sadegh M Milajerdi, Birhanu Eshete, Rigel Gjomemo, and VN 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*, pages 1795–1812, 2019.

[28] Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1137–1152. IEEE, 2019.

[29] Kunal Mukherjee, Joshua Wiedemeier, Tianhao Wang, Muhyun Kim, Feng Chen, Murat Kantarcioglu, and Kangkook Jee. Interpreting gnn-based ids detections using provenance graph structural features. *arXiv e-prints*, pages arXiv–2306, 2023.

[30] Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eyers, Margo Seltzer, and Jean Bacon. Practical whole-system provenance capture. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 405–418, 2017.

[31] Judea Pearl. Causal inference in statistics: An overview. 2009.

[32] Kexin Pei, Zhongshu Gu, Brendan Saltaformaggio, Shiqing Ma, Fei Wang, Zhiwei Zhang, Luo Si, Xiangyu Zhang, and Dongyan Xu. Hercule: Attack story reconstruction via community discovery on correlated log graph. In *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, pages 583–595, 2016.

[33] Wei Qiao, Yebo Feng, Teng Li, Zhuo Ma, Yulong Shen, JianFeng Ma, and Yang Liu. Slot: Provenance-driven apt detection through graph reinforcement learning. *arXiv preprint arXiv:2410.17910*, 2024.

[34] Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. Flash: A comprehensive approach to intrusion detection via provenance graph representation learning. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 139–139. IEEE Computer Society, 2024.

[35] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. Higher-order explanations of graph neural networks via relevant walks. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7581–7596, 2021.

[36] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[37] Yun Shen, Enrico Mariconti, Pierre Antoine Vervier, and Gianluca Stringhini. Tiresias: Predicting security events through deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 592–605, 2018.

[38] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMlR, 2017.

[39] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *Proceedings of the ACM web conference 2022*, pages 1018–1027, 2022.

[40] Juntao Tan, Shuyuan Xu, Yingqiang Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. Counterfactual explainable recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1784–1793, 2021.

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

[42] Qi Wang, Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Junghwan Rhee, Zhengzhang Chen, Wei Cheng, Carl A Gunter, et al. You are what you do: Hunting stealthy malware via data provenance analysis. In *NDSS*, 2020.

[43] Su Wang, Zhiliang Wang, Tao Zhou, Hongbin Sun, Xia Yin, Dongqi Han, Han Zhang, Xingang Shi, and Jiahai Yang. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Transactions on Information Forensics and Security*, 17:3972–3987, 2022.

[44] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[45] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International conference on machine learning*, pages 12241–12252. PMLR, 2021.

[46] Jun Zengy, Xiang Wang, Jiahao Liu, Yinfang Chen, Zhenkai Liang, Tat-Seng Chua, and Zheng Leong Chua. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 489–506. IEEE, 2022.