SLOT: Provenance-Driven APT Detection through Graph Reinforcement Learning

Wei Qiao*
Xidian University
State Key Laboratory of Integrated Services Networks (ISN)
Xi'an, China
qiaoweiXidian@gmail.com

Yebo Feng*
Nanyang Technological University
Singapore
yebo.feng@ntu.edu.sg

Teng Li†
Xidian University
State Key Laboratory of Integrated Services Networks (ISN)
Xi'an, China
litengxidian@gmail.com

Zhuo Ma†
Xidian University
Xi'an, China
mazhuo@mail.xidian.edu.cn

Yulong Shen
Xidian University
Xi'an, China
ylshen@mail.xidian.edu.cn

Jianfeng Ma
Xidian University
Xi'an, China
jfma@mail.xidian.edu.cn

Yang Liu
Nanyang Technological University
Singapore
yangliu@ntu.edu.sg

Abstract

Advanced Persistent Threats (APTs) represent sophisticated cyberattacks characterized by their ability to remain undetected within the victim system for extended periods, aiming to exfiltrate sensitive data or disrupt operations. Existing detection approaches often struggle to effectively identify these complex threats, construct the attack chain for defense facilitation, or resist adversarial attacks. To overcome these challenges, we propose SLOT, an advanced APT detection approach based on provenance graphs and graph reinforcement learning. SLOT excels in uncovering multi-level hidden relationships, such as causal, contextual, and indirect connections, among system behaviors through provenance graph mining. SLOT implements semi-supervised learning with limited labels through efficient label similarity computation, significantly enhancing both detection performance and model robustness. By pioneering the integration of graph reinforcement learning, SLOT dynamically adapts to new user activities and evolving attack strategies, enhancing its resilience against adversarial attacks. Additionally, SLOT automatically constructs the attack chain according to detected attacks with clustering algorithms, providing precise identification of attack paths and facilitating the development of defense strategies. Evaluations with real-world datasets demonstrate SLOT's outstanding accuracy, efficiency, adaptability, and robustness in APT detection, with most metrics surpassing state-of-the-art methods. Additionally, case studies conducted to assess SLOT's effectiveness in supporting APT defense further establish it as a practical and reliable tool for cybersecurity protection.

CCS Concepts

• **Security and privacy** → **Intrusion detection systems**; **Network security**; **Systems security**; • **Computing methodologies** → *Machine learning*.

Keywords

advanced persistent threat (APT), APT Detection, intrusion detection, reinforcement learning, graph neural network

1 Introduction

An Advanced Persistent Threat (APT) is a sophisticated cyberattack in which a malicious actor gains unauthorized access to a network, remaining undetected for a long time to steal sensitive data or disrupt system operations [49]. These attacks are highly targeted, using advanced techniques to maintain stealth and persistence within the victim's network [7], resulting in severe damage to the victim's system. For example, the latest SektorCERT report from Denmark [48] reveals that Sandworm, an APT hacking group, launched coordinated attacks on 22 Danish energy companies, causing widespread disconnection of remote devices. Additionally, Symantec researchers have uncovered a four-month APT operation by Grayling [51], targeting manufacturing, IT, and biomedical sectors in Taiwan, the US, and Vietnam, involving theft of sensitive data.

To effectively combat APTs, it is crucial to detect them early with both timeliness and precision. Traditional detection methods may struggle with these complex and prolonged attacks, often resulting in either inadequate accuracy or excessive time to produce detection results [13, 15]. Recently, researchers have increasingly utilized provenance graphs to trace APT activities, progressively achieving practical efficacy and efficiency.

Based on their methodologies, existing provenance-graph-based approaches for APT detection can be classified into three categories: **1) Statistics-based approaches** [22, 34, 56] consider infrequent

*Wei Qiao and Yebo Feng contributed equally to this research.
†Corresponding authors: Teng Li and Zhuo Ma.

events as suspicious and use provenance graphs to model and identify these events. However, these approaches focus solely on direct event connections, overlooking the deep semantics and hidden relationships within provenance graphs, which can result in a high rate of false positives and diminished reliability. **2) Specification-based approaches** [21, 23, 24, 38, 39] leverage expert knowledge to develop heuristic rules for APT detection. While this method effectively maintains a low false positive rate in cases it covers, creating these heuristic rules demands substantial prior expert knowledge, which must be distilled and refined by experts, making it a resource-intensive process. **3) Learning-based approaches** [6, 20, 29, 33, 44, 47, 57, 60, 62] use various deep learning techniques to model APT attack patterns and system behaviors, subsequently utilizing these models for APT detection. These methods can achieve satisfactory detection performance, particularly with the integration of provenance graphs and graph neural network (GNN), which can uncover subtle connections between system events to enhance APT detection. However, they remain vulnerable to adversarial attacks, where attackers can easily mimic legitimate users to mislead the model into producing incorrect detection results.

To address the aforementioned gaps, we introduce Slot, an advanced APT detection approach based on provenance graphs and graph reinforcement learning. Unlike existing methods, Slot achieves enhanced APT detection accuracy by leveraging Latent Behavior Mining and Graph Reinforcement Learning to deeply excavate multi-level hidden relationships (e.g., causal dependencies, contextual interactions, and indirect attack traces) from system behavior data. Additionally, by incorporating graph reinforcement learning, Slot can autonomously learn and adapt to new user activities and attack strategies without relying on threat reports derived from expert insights, thereby enhancing its adaptability and resilience against adversarial attacks. Furthermore, Slot can automatically construct the attack chain, accurately identifying the attack path, which simplifies the defense process.

Slot is comprised of five integrated modules that work together to detect highly stealthy APT attacks and assist administrators in formulating effective defense strategies. (1) It begins with the Graph Construction Module, which processes system logs to create an initial provenance graph that reflects system behaviors. (2) The Latent Behavior Mining Module then applies attention mechanisms and graph transformation techniques to discover hidden relationships within the graph, enhancing the depth and comprehensiveness of subsequent analysis. (3) The Embedding Module utilizes a graph reinforcement learning algorithm (i.e., Bernoulli multi-armed bandit [63]) to embed semantic and topological features of graph nodes as vectors, aggregates similar nodes, and produces updated feature vectors to filter out camouflaged entities, thereby improving accuracy and robustness. (4) The Threat Detection Model, which combines a multi-layer perceptron (MLP) [46] and Isolation Forest (iForest) [11] to comprehensively identify APTs. (5) Finally, the Attack Chain Reconstruction module traces the attack pathway by clustering correlative nodes, aiding in the development of effective defense strategies.

We evaluated Slot using three publicly available APT datasets from well-respected institutions and research communities. The evaluation results show that Slot achieves an overall detection accuracy of approximately 99%, outperforming all existing state-of-the-art (SOTA) techniques. Furthermore, Slot completes the detection process more quickly. Additionally, we tested Slot against adversarial strategies proposed by Goyal et al. [17], which involve altering the neighborhood of nodes in the attack graph to resemble benign nodes. Leveraging its graph reinforcement learning mechanism, Slot exhibits enhanced resistance compared to SOTA approaches. Ultimately, Slot has proven its effectiveness in tracing attack pathways and aiding in the development of defense strategies in case studies.

In summary, this paper makes the following contributions.

- We propose Slot, an accurate method for Advanced Persistent Threat (APT) detection that maintains robustness in adversarial environments. It effectively identifies malicious activities within large-scale audit logs while reconstructing attack chains to support the development of targeted defense strategies.
- We propose an advanced graph mining technique capable of efficiently uncovering multi-level hidden relationships—such as causal, contextual, and indirect connections—within the graph.
- Slot implements semi-supervised learning with limited labels through efficient label similarity computation, significantly enhancing both detection performance and model robustness.
- We integrate reinforcement learning into provenance graph analysis. By embedding semantic and topological features and leveraging the adaptive dynamics of reinforcement learning, Slot effectively endures highly adversarial environments.
- We performed comprehensive evaluations using real-world datasets, and the results underscore Slot's effectiveness in detecting APTs, its resilience against adversarial attacks, and its capability to support the development of effective defense strategies.

2 Related Work and Its Limitations

This section categorizes and summarizes the related work on APT detection and graph reinforcement learning, and discusses their respective limitations.

Table 1: Obstacles that limit the performance of APT detection approaches.

Model	Threat Report	Semantic Encoding	Attack Reconstruction	Adversarial Robustness	Granularity
Slot	✗	✓	✓	✓	Node
Nodoze[22]	✗	✗	✗	✗	Node
Poirot[38]	✓	✓	✓	✗	Graph
Unicorn[20]	✗	✓	✗	✗	Graph
ShadeWatcher[62]	✗	✓	✗	✗	Edge
FLASH[47]	✗	✓	✓	✓	Node
MAGIC[26]	✗	✗	✗	✗	Node
ThreaTrace[57]	✗	✗	✗	✗	Node

2.1 APT Detection

To effectively detect sophisticated APTs, provenance graphs are used to model system events and trace attacks. Recent detection methods can be classified into three categories [62]: statistics-based, specification-based, and learning-based methods. And it lists the existing limitations of current approaches in Table 1.

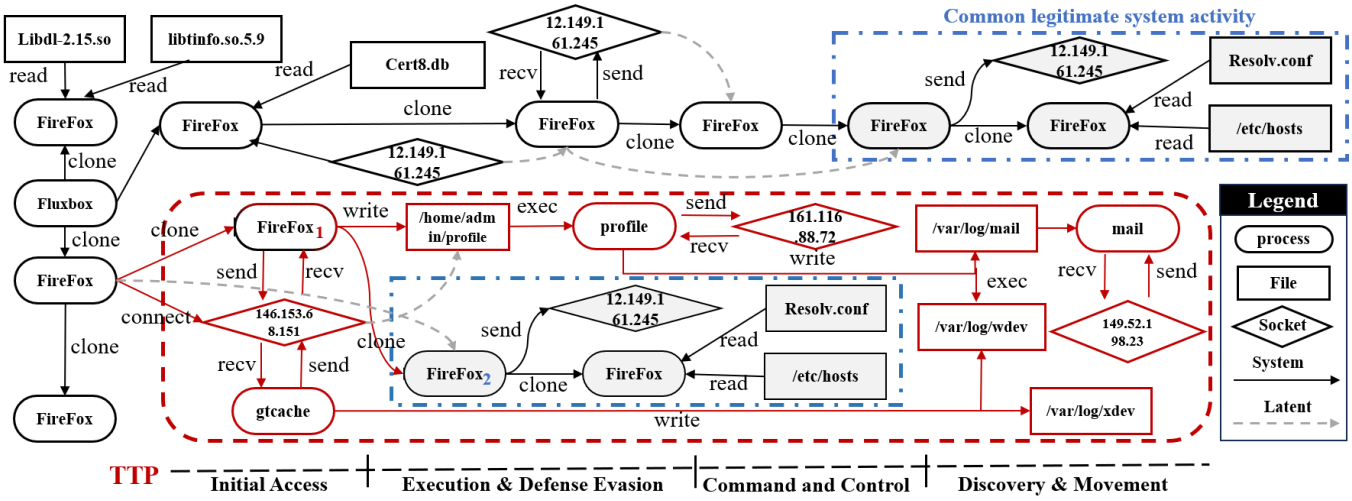


Figure 1: An example of an attack provenance graph from the DARPA E3 dataset [1]. Attackers launched a series of attacks exploiting the Firefox backdoor vulnerability, along with camouflage activities to cover up the attacks. SLOT effectively identifies all critical attack components, as indicated by the red nodes in the figure.

Statistics-based approaches: Statistical methods [22, 34, 56] assume that attacks correlate with unusual system activities, quantifying suspiciousness based on interaction frequencies among system entities. However, rare events aren't always anomalies. For instance, in our scenario 1, Firefox loads the benign module *Org.chromium.iyhyah* for the first time, which a statistical method flags as a false positive. This example shows that such methods fail to capture deep semantics and latent relationships, leading to high false positive rates.

Specification-based approaches: Specification-based methods match audit logs with threat report [21, 38] or use expert knowledge for abnormality scoring [24, 39], raising alerts when anomalies exceed a set threshold. While expert-driven detection generally achieves a low false positive rate, it requires specialized expertise to design effective strategies. As systems evolve and attack techniques grow more sophisticated, continuous expert intervention is needed, making it difficult for the system to adapt dynamically to changes in the network environment.

Learning-based approaches: Learning-based methods have proven effective for classification and anomaly detection by modeling system behaviors from logs. For example, Unicorn [20] uses graph similarity matching to identify anomalous graphs, but a single alert can involve thousands of logs, complicating verification. At the edge level, ShadeWatcher [62] models system interactions using GNNs, but high computational costs arise from numerous edges. At the node level, FLASH [47] and ThreaTrace [57] utilize GNN to detect anomalies at the node level by learning the structural information of nodes and detecting deviations from this learned behavior. MAGIC [26] utilizes masked autoencoder and KNN to identify abnormal nodes. While node-level detection effectively identifies anomalies, current methods fail to utilize heterogeneous entity information and overlook the impact of benign masquerading by neighboring nodes on embeddings. As exemplified in our attack scenario (Figure 1), embedding benign DNS resolution behaviors within the malicious red attack chain can disrupt the embedding representation of the root malicious node, "Firefox1", thereby enabling adversarial mimicry and evasion.

2.2 Graph Reinforcement Learning

Reinforcement Learning (RL) has recently achieved success in addressing challenges across various fields [14], including robotics [30], gaming [9], and natural language processing (NLP) [54]. Researchers have found that RL methods enable effective exploration of the topological structures and attribute information of graphs by analyzing key components such as nodes, links, and subgraphs. SUGAR [50] employs Q-learning to adaptively select significant subgraphs to represent discriminative information of the graphs. Bachu et al. [8] develop perturbation strategies for local explanations of graph data by optimizing multi-objective scores. Other related works [28, 61] have demonstrated impressive performances in graph representation learning through RL, proving its efficacy in this domain. But currently, graph reinforcement learning has not yet been attempted in the field of log provenance detection.

3 Problem Formalization

In this section, we present real-world attack scenarios from the DARPA dataset [1] to highlight the limitations of current provenance-based APT detection systems. These scenarios demonstrate how existing systems may fail to detect complex and evolving attacks. We also define the specific threat scenarios that the SLOT system aims to address, focusing on improving detection accuracy, reducing false positives, and enhancing response times.

3.1 Attack Scenario

Figure 1 illustrates a Firefox backdoor attack. The red subgraph represents a simplified version of the original attack. A victim computer running the vulnerable Firefox version 54.0.1 unknowingly interacts with a malicious ad server located at 146.153.68.151. This

server exploits a backdoor in Firefox, injecting the binary executable '*Drakon*' 1 into the process memory. '*Drakon*' subsequently spawns a new process with root privileges (/home/admin/profile), which connects to an attacker's server at 161.116.88.72, thus granting the attacker full access to the victim's computer.

The existing APT detection systems based on learning distinguish between attack activities (red entities) and benign behaviors (black entities) by analyzing behavior patterns in provenance graphs. Common DNS resolution behaviors (blue subgraph) can be classified more clearly as benign structures after repeated learning sessions. However, in APT attacks, attackers may familiarize themselves with and mimic common benign activities of the system over a long period of information infiltration, thereby launching adversarial attacks that embed these benign substructures into their attack tactics without altering the actual attack logic. By repeatedly performing these common benign activities, malicious processes can be misjudged as benign during the detection process, thus achieving attack evasion.

3.2 Threat Model

Building on prior APT detection research [20, 47, 57, 62], our study focuses on scenarios where attackers exploit software vulnerabilities and communication backdoors for system control and persistence. We exclude hardware trojans and side-channel attacks, which are undetectable through system audits. We assume attackers know the target hosts' benign activities, enabling mimicry attacks. Additionally, we consider audit log data secure, supported by robust security tracing [42] and anti-tampering [5, 41], making our provenance graphs reliable for effective threat detection and analysis.

3.3 Design Goals

To defend against APT attacks, Slot models and analyzes system call data recorded by audit logs. The ultimate goal of Slot is to provide security analysts with more effective and streamlined attack insights, thereby accelerating the alert handling process. We believe that APT detection should achieve the following objectives when raising alarms: (1) For the complex multi-stage characteristics of APT attacks, Slot must have precise detection capabilities to identify attack-related entities; (2) Amid vast volumes of logs, Slot should maintain a low false positive rate to prevent alert fatigue; (3) In the face of highly covert adversarial attacks, Slot must be highly robust, capable of dealing with attackers' camouflaging behaviors; (4) Finally, Slot should provide security analysts with more streamlined and effective traceable analysis results, enabling the validation of alarm events within a reasonable time cost.

4 Methodology

4.1 Overview

Slot is a fine-grained, adversarial threat provenance scheme. It leverages attention mechanisms and graph transformation techniques to uncover deep hidden relationships, and then combines reinforcement learning with the provenance graph to guide relational aggregation in GNN. Slot features flexible policy extension capabilities and dynamic adaptability, allowing it to address adversarial attacks in real-world scenarios. Slot not only provides malicious detection through model learning but also proposes a method for considering unknown anomalies. Finally, By integrating LPA and ATT&CK [3] encoding, it offers a concise and low false-positive attack chain, effectively accelerating manual verification. Figure 2 depicts Slot' architecture consisting of five major components:

① **Graph Construction (§ 4.2).** Slot constructs a graph based on entity relationships from system call logs and extracts semantic information from the logs as initial features for the nodes.

② **Latent Behavior Mining (§ 4.3).** Slot further explored the behavioral relationships on the original system provenance graph. It combined system call relationships to construct deeper path connections. This approach enriches event behaviors and accelerates the propagation of information within the graph.

③ **Embedding with Graph Reinforcement Learning (§ 4.4).** In the graph embedding stage, Slot captures system behavior patterns through graph reinforcement learning. Specifically, Slot combines the semantic features and the topological features of nodes to calculate the similarity between entities in the provenance graph. Subsequently, it uses a reinforcement learning strategy to guide the selection and aggregation of relationships in the GNN based on this similarity, thereby achieving effective modeling of system behavior and obtaining the true feature vectors of the nodes.

④ **Threats Detection (§ 4.5).** Using the trained model, Slot performs node representation on the test logs. For previously learned behavior patterns, Slot can directly classify node vectors as benign or malicious. However, to address unknown behaviors in APT attacks, Slot uses anomaly detection to identify entities that deviate from known behavior patterns. Slot treats both malicious entities and anomalous entities as attack activities.

⑤ **Attack-chain Reconstruction (§ 4.6).** To assist security analysts in alert verification, Slot constructs scattered nodes into a complete attack activity chain, eliminating the need for tracing across thousands of nodes. Slot classifies attack nodes into different attack chains by clustering, using the TTP (Tactics, Techniques, and Procedures) phase tags from the ATT&CK framework [3] along with the feature embeddings of the nodes.

4.2 Graph Construction

Table 2: System behaviors extracted from audit logs.

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

Slot constructs a comprehensive system provenance graph using audit data collected from logging infrastructures such as Windows ETW [12], Linux Audit [2], and CamFlow [43]. It adheres to the existing definitions of provenance graphs and employs recent graph preprocessing techniques [47, 57], abstracting events into subjects, objects, and relationships, as illustrated in Table 2. Here, source nodes like Process/Thread represent subjects, and target nodes like Files or sockets represent objects, with system call events serving as the relationships to form graph $\mathcal{G}(S, R, O)$. In addition to focusing on system behaviors, Slot further enriches

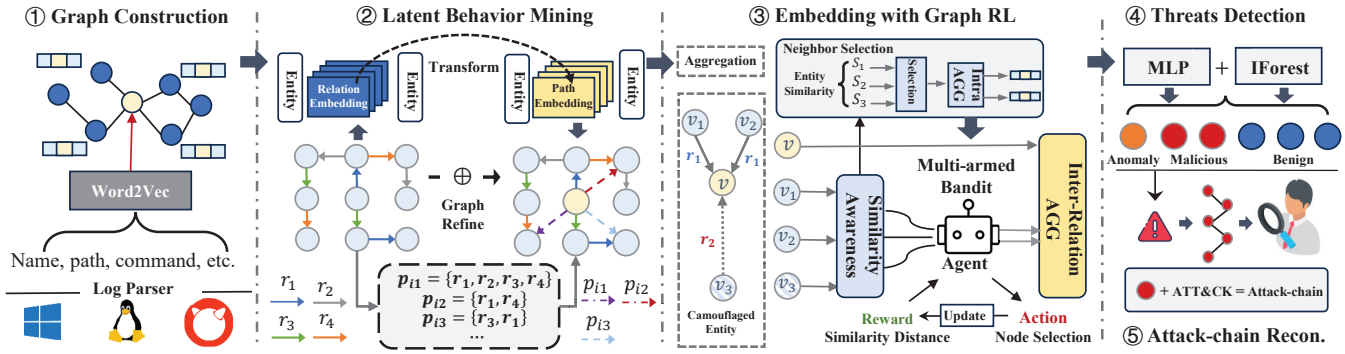


Figure 2: The workflow of SLOT. SLOT initializes event node features through semantic embedding, enhances graph relationships via latent relation mining, and then employs graph reinforcement learning to achieve robust node embedding representations, ultimately accomplishing malicious activity detection and attack reconstruction.

the relationships between behaviors and explores multi-hop latent relations in Section 4.3. Regarding node features, SLOT fully considers meaningful semantic information such as process names and command-line parameters for process nodes, file paths for file nodes, network IP addresses and ports for netflow nodes. SLOT effectively mines deep semantic patterns in provenance logs by constructing node "summary sentences" through aggregating semantic attributes (process names, file paths, IPs) and causal events (system calls) from 1-hop neighbors, incorporating timestamp-sorted event sequences with positional encoding to preserve temporal relationships, and leveraging the Word2Vec model [37] to learn vector representations of semantic features.

4.3 Latent Behavior Mining

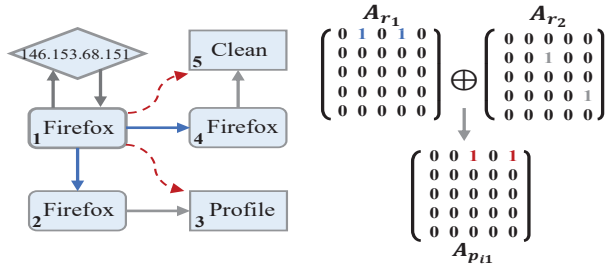


Figure 3: Latent behavior adjacency matrix construction, with colored edges indicating different relationships.

A complete user behavior or attack campaign typically involves multiple system calls, which manifest as multi-hop node relationships in provenance graphs. To accelerate node information propagation and uncover implicit dependencies between distant nodes, SLOT introduces a Latent Relation Mining Module. This module automatically constructs hidden relationships across multi-hop nodes through a relational attention mechanism, prioritizing causal chains and attack-relevant contextual patterns.

In the mining of latent relationships, SLOT leverages attention mechanisms [55] and graph transformation [59] techniques to automatically mine advanced behavioral paths. For basic system call behavior r_i, it mines deep event-behavior relationships $p_{ik} \in P_i$, and then integrates low-dimensional overall system relationships with path-based advanced behavioral relationships to generate a new provenance graph. We define $p_{ik} = r_{i1}, r_{i2}, ..., r_{iL}$, represented as an L-hop event path, where each $r_{il} \in p_{ik}$ is a basic system call behavior in the event path. In the path generation module, we use multiple attention mechanisms to softly select L relationships and combine these relationships to form path P_{ik}. Specifically, for sub-behaviors within events, we generate corresponding relationship embedding using attention mechanisms: $\mathbf{r}_{il} = \sum_{j=1}^{n} \alpha_{ij}^{l} \mathbf{r}_j$, where α_{ij}^{l} is calculated through scores normalized by the softmax function, which is computed as follows:

$$\alpha_{ij}^{l} = \frac{\exp(\mathbf{r}_j^T \sigma(\mathbf{r}_i \mathbf{W}^l + \mathbf{b}^l))}{\sum_{j'=1}^{n} \exp(\mathbf{r}_{j'}^T \sigma(\mathbf{r}_i \mathbf{W}^l + \mathbf{b}^l))}, \tag{1}$$

where W and b are parameters used in the path generation process, and σ represents the activation function. By incorporating attention mechanisms, the model can effectively select multi-hop paths most relevant to a given relationship r_i, thereby enhancing the representation of complex relationships between entities in the provenance graph. For each path p_{ik} composed of L hops, its adjacency matrix $A_{p_{ki}}$ is obtained by the product of the adjacency matrices of the corresponding relationships: $A_{p_{ik}} = A_{r_{i1}} \cdot A_{r_{i2}} \dots A_{r_{iL}}$, where each $A_{r_{il}}$ is the adjacency matrix for the relationship r_{il}. Direct computation of this product can be resource-intensive, so our method uses an efficient representation to approximate this process. First, each relationship's adjacency matrix is approximated by $A_{ril} \approx E \cdot \mathrm{diag}(r_{il}) \cdot E^T$, where E represents the matrix of entity embeddings, and $diag(r_{il})$ transforms the relationship embedding r_{il} into a diagonal matrix form. As shown in Figure 3, the adjacency matrix for the path $A_{p_{ik}}$ can be calculated as follows to capture the composite effect of multiple relationships along the path:

$$\begin{aligned} A_{p_{ik}} &\approx (\mathbf{E} \cdot \mathrm{diag}(\mathbf{r}_{i1}) \cdot \mathbf{E}^T) \cdot \ldots \cdot (\mathbf{E} \cdot \mathrm{diag}(\mathbf{r}_{iL}) \cdot \mathbf{E}^T) \\ &\approx \mathrm{diag}(\mathbf{r}_{i1}) \cdot \mathbf{E}^T \cdots \mathbf{E} \cdot \mathrm{diag}(\mathbf{r}_{iL}) \\ &\approx \mathbf{E} \cdot \mathbf{p}_{ik} \cdot \mathbf{E}^T. \end{aligned} \tag{2}$$

4.4 Embedding with Graph Reinforcement Learning

SLOT utilizes graph reinforcement learning to obtain high-quality node embeddings from the provenance graph. The embedding module consists of three stages: 1) In the similarity-aware phase, the node's semantic features and topological features are embedded into vectors. 2) Based on the similarity calculation of node feature vectors, we design an adaptive Bandit neighbor selector using reinforcement learning [45]. Finally, we aggregate features by integrating multiple relationships (including system call and latent relationships) to generate updated node feature vectors.

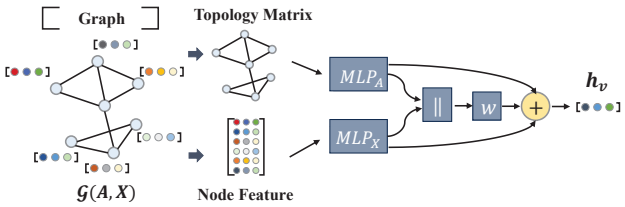


Figure 4: Embed semantic features and topological features as vectors.

4.4.1 **Feature-Topology Similarity Awareness.** Previous research has explored various types of attacker camouflages from both behavioral [27, 40] and semantic [58] perspectives. These camouflages can make the features of malicious activities similar to those of benign entities, further inducing GNN to generate incorrect node embeddings. To address these issues of node feature camouflage, we believe that an effective similarity metric is necessary to filter out disguised neighbors before applying GNNs. SLOT achieves lightweight node similarity awareness by simultaneously considering node attribute features and topological features [31] shown in Figure 4.

MLP on node features. A simple approach to node classification is to ignore graph topology and simply train an MLP on node features. Research [65] has demonstrated that MLPs can actually perform relatively well on heterogeneous graphs-achieving higher or roughly equivalent efficacy compared to various GNNs. Thus, we represent the node features as follows:

$$\mathbf{h_X} = \mathrm{MLP_X}(\mathbf{X}) \in \mathbb{R}^{d \times n}, \tag{3}$$

where $\mathbb{R}^{d \times n}$ denote the matrix of node features with input dimension d.

LINK regression on graph topology. Another extreme is LINK [64], a simple baseline that solely utilizes graph topology. We extend LINK to compute feature embeddings for the topological relationship A as follows:

$$\mathbf{h_A} = \mathrm{MLP_A}(\mathbf{A}) \in \mathbb{R}^{d \times n}. \tag{4}$$

Finally, we let $[h_1; h_2]$ denote concatenation of vectors h_1 and h_2 and map the entity feature vectors as:

$$\mathbf{h_v} = \mathrm{MLP}_f\left(\sigma\left(\mathbf{W}[\mathbf{h_A}; \mathbf{h_X}] + \mathbf{h_A} + \mathbf{h_X}\right)\right). \tag{5}$$

We use the L1 distance between the feature vectors of two nodes as their measure of similarity. For a center node v under relation r at the $l-th$ layer and edge $(v, v') \in \mathcal{E}_r^{(l-1)}$, we can define the similarity measure as:

$$\mathcal{D}^{(l)}(v, v') = \left\| \sigma\left(MLP^{(l)}(\mathbf{h}_v^{(l-1)})\right) - \sigma\left(MLP^{(l)}(\mathbf{h}_{v'}^{(l-1)})\right) \right\|_1. \tag{6}$$

4.4.2 **Adaptive Bandit Neighbor Selector.** Given relation camouflage, attackers may connect to varying numbers of benign entities under different relationships [17]. We should select similar neighbors (i.e., filter disguised neighbors) to enhance the capability of GNN. To adaptively select appropriate neighbors, we utilize a similarity-aware neighbor selector to filter out nodes exhibiting inappropriate behaviors due to adversarial actions or inaccurate feature extraction [45]. More specifically, for each central node, the selector utilizes Top-p sampling and adaptive filtering thresholds to construct similar neighbors under each relationship.

Top-p Sampling.

Before aggregating information from the central node v and its neighbors, we perform Top-p sampling to filter dissimilar neighbors based on different relationships. The filtering threshold p_r^l for the $l-th$ layer relationship r is defined within the range $[0, 1]$, representing the selection ratio from all neighbors.

Finding the Optimal Thresholds with RL.

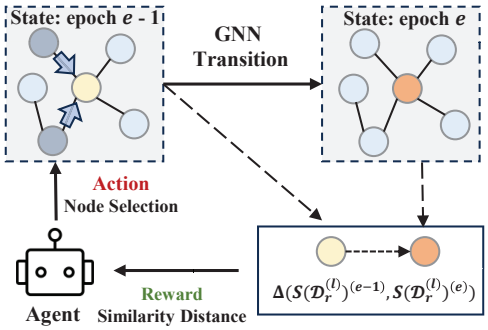


Figure 5: RL-guided graph neighbor node selection.

The logic behind reinforced neighborhood selection is to mine appropriate neighbor nodes for the central node. Previous GNN-based Works [47, 57] are built on homogeneous benchmark graphs, devoid of noise caused by disguised attackers. Inspired by neighborhood sampling [45], we found that the scenario of resisting neighborhood interference satisfies the conditions for applying a Bernoulli multi-armed bandit reinforcement learning approach, allowing for the adaptive selection of suitable neighborhoods for each central node. We represent the RL module as a Markov Decision Process $MDP < A, S, R, T, F >$, used for the filtering threshold of relationships. A is the action space, S is the node state, R is the reward function, T is the state transition, and F is the termination condition.

- **Action** The action in the RL model dictates the updates to $p_r^{(l)}$ according to the received rewards. Given that $p_r^{(l)}$ falls within the range $[0, 1]$, we define the action $a_r^{(l)}$ as modifying $p_r^{(l)}$ by adding or subtracting a fixed increment $\tau \in [0, 1]$.
- **State** Due to the inability to use the GNN's loss as the environment state, we use the average node distance, computed through

similarity-aware calculations, as the state. The average neighbor distance in train node $\mathcal{V}_{train}$ at layer l under relation r during epoch e is:

$$\mathcal{S}(\mathcal{D}_r^{(l)})^{(e)} = \frac{\sum_{v \in \mathcal{V}_{train}} \mathcal{D}_r^{(l)}(v, v')^{(e)}}{|\mathcal{V}_{train}|}. \tag{7}$$

- **Reward** The optimal $p(l)^r$ at layer l under relation r aims to find the most similar neighbors to the central node. A decrease in the average neighbor distance signifies that the selected neighbors exhibit higher similarity to the central node. We design a binary reward based on the difference in average distances between two consecutive epochs. We define the reward for epoch e as:

$$f(p_r^{(l)}, a_r^{(l)})^{(e)} = \begin{cases} +1, \mathcal{S}(\mathcal{D}_r^{(l)})^{(e-1)} - \mathcal{S}(\mathcal{D}_r^{(l)})^{(e)} \geq 0, \\ -1, \mathcal{S}(\mathcal{D}_r^{(l)})^{(e-1)} - \mathcal{S}(\mathcal{D}_r^{(l)})^{(e)} < 0. \end{cases} \tag{8}$$

  When the distance between two consecutive epochs is positive, the reward is positive; otherwise, the reward is negative. Finally, we greedily update the actions based on the rewards.
- **Transition** State transitions are achieved through the forward propagation of GNN. When the neighbor selection threshold changes, the embeddings of the nodes are updated, leading to an update of the entire system's state.
- **Terminal** We define the terminal condition for RL as:

$$\sum_{e-10}^{e} f(p_r^{(l)}, a_r^{(l)})^{(e)}| \leq 2, where \quad e \geq 10. \tag{9}$$

  It means that the RL converges in the recent ten epochs and indicates an optimal threshold $p_r^{(l)}$ is discovered.

4.4.3 **Contextual Relationship Integrator.** After filtering the neighbors for each relationship through reinforcement learning, we directly use the optimal filtering threshold $p_r^{(l)}$, learned from the RL process as the aggregation weight between relationships. We then use a GNN to aggregate information from neighbors across different relationships, generating updated node feature vectors. Formally, at the $l-th$ layer for relationship r, after applying top-p sampling, the neighbor aggregation for node v is defined as follows:

$$\mathbf{h}_{v,r}^{(l)} = \text{ReLU}\left(\text{GNN}_r^{(l)}\left(\left\{\mathbf{h}_{v'}^{(l-1)} : (v, v') \in \mathcal{E}_r^{(l)}\right\}\right)\right). \tag{10}$$

Then, we define the inter-relation aggregation as follows:

$$\mathbf{h}_v^{(l)} = \text{ReLU}\left(\text{GNN}_{all}^{(l)}\left(\mathbf{h}_v^{(l-1)} \oplus \{p_r^{(l)} \cdot \mathbf{h}_{v,r}^{(l)}\}|_{r=1}^{R}\right)\right). \tag{11}$$

4.5 Threats Detection

After obtaining the embeddings of the system entities, we perform threat detection using a trained MLP and anomaly-based Isolation Forest, classifying the system entities into benign, malicious, and anomalous categories.

Learning Detection. Existing node-level approaches [26, 47, 57], whether based on anomaly-based unsupervised detection or self-supervised methods relying on data structure or attributes, have shown very high false positive rates. Gui et al. [18] argued that in highly adversarial and data-imbalanced scenarios, semi-supervised learning with a small amount of labeled data can more effectively capture imbalanced data patterns, thereby increasing the embedding distance between benign and malicious behavior patterns. To classify the embedded features, we trained an MLP classifier. Slot uses only 10% of the attack data to assist in characterizing the benign patterns, and through oversampling, semi-supervised learning is performed with the benign training set. For jointly training the similarity measure with GNNs, a heuristic approach involves appending it as a new layer before the GNN aggregation layer. We define the cross-entropy loss for the MLP at the first layer as follows:

$$\mathcal{L}_{\text{Simi}}^{(1)} = \sum_{v \in \mathcal{V}} -\log\left(y_v \cdot \sigma\left(MLP^{(l)}(\mathbf{h}_v^{(l)})\right)\right). \tag{12}$$

For each node v, its final embedding is the output of the GNN at the last layer $\mathbf{z}_v = \mathbf{h}_v^{(L)}$. We can define the loss of GNN as a cross-entropy loss function:

$$\mathcal{L}_{\text{GNN}} = \sum_{v \in \mathcal{V}} -\log\left(y_v \cdot \sigma(MLP(\mathbf{z}_v))\right). \tag{13}$$

By combining the losses of the similarity modeling and the GNN modeling, we minimize the following objective function to learn parameters in our threats detection model:

$$\mathcal{L} = \mathcal{L}_{\text{GNN}} + \lambda_1 \mathcal{L}_{\text{Simi}}^{(1)} + \lambda_2 ||\Theta||_2, \tag{14}$$

where $||\Theta||_2$ is the $L2$-norm of all model parameters, λ_1 and λ_2 are weighting parameters. Finally, we can classify the node feature vectors using the trained MLP.

Anomaly Detection. Learning-based scheme is effective in detecting known benign or malicious patterns but struggles with identifying unknown anomalies. To address this limitation, we have additionally introduced an unsupervised detector on top of the supervised learning classifier to identify unknown anomalies, or outliers. The key intuition here is that anomalous entities, compared to learned behavior patterns, tend to be sparse and distant from other data points.

Slot utilizes an Isolation Forest [11] to partition node feature vectors and compute node anomaly scores. Initially, the feature vector h in the detection representation is normalized as $\mathbf{h}' = \frac{\mathbf{h} - \mu_{\mathbf{h}}}{\sigma_{\mathbf{h}}}$. Subsequently, the node anomaly score is calculated as:

$$Score(\mathbf{h}, n) = 2^{-\frac{E(l(\mathbf{h}'))}{c(n)}}, \tag{15}$$

where n represents the total number of samples, and $E(l(\mathbf{h}'))$ denotes the average path length of the data point across all trees, which indicates the average depth at which the data point is isolated (i.e., the path ends) in the tree. $c(n)$ represents the average path length of a data point in a completely random tree.

After parallel detection, Slot combines the results of both classifiers to make the final decision. If the supervised learning model makes a high-confidence prediction about a behavior (either benign or malicious), that prediction is adopted. On the other hand, if the anomaly detection model indicates that the behavior might be anomalous, it is labeled as anomalous.

4.6 Attack-chain Reconstruction

Considering that node-based alerts are dispersed in extremely large provenance graphs, verifying an event activity requires a forward or backward traceability investigation of nodes. Dispersed nodes are inconvenient for security analysts to validate the analysis, and node-level based alerts are accompanied by a large number of false-positive nodes, which can lead to severe alert fatigue.

To address this challenge, we propose a novel reconstruction scheme based on alert nodes, which, unlike previous approaches [10, 47] that directly employed clustering or path association, incorporates the ATT&CK framework [3] into attack chain reconstruction for the first time. This method involves mapping attack techniques from the ATT&CK framework and assigning appropriate TTP (Tactics, Techniques, and Procedures) codes $\mathbf{t}_i$ to each node. We represent TTPs using one-hot encoding; if the ATT&CK framework includes N distinct TTPs, each TTP is encoded as an N-dimensional vector, with the position corresponding to the TTP set to 1 and the rest set to 0, with each technique or tactic corresponding to one code. We concatenate the node features with the TTP codes to obtain the initial node label $x_i = [\mathbf{h}_i; \mathbf{t}_i]$, and then update the node labels using the Label Propagation Algorithm (LPA):

$$\mathbf{y}_i^{(t+1)} = \text{mode}(\mathbf{y}_j^{(t)} \in \mathcal{N}(i)). \tag{16}$$

where y_i represents the label of node i during the $i-th$ iteration, $N(i)$ denotes the set of neighboring nodes of i, and mode indicates the selection of the most frequent label. The label propagation process is repeated until the labels of all nodes no longer change. This combination of TTP patterns effectively filters out false positives within the attack chain and generates the final attack path.

5 Evaluation

In this section, experiments are performed to validate Slot's advantage and answer the following key research questions:

- **RQ1**: Can similarity awareness provided by a small number of labels enable effective APT detection under semi-supervision?
- **RQ2**: How effective is Slot in detecting APTs compared to current state-of-the-art appraoches?
- **RQ3**: What is the system overhead associated with using Slot?
- **RQ4**: How effective are the components of our Slot in achieving their intended functions?
- **RQ5**: How do different hyperparameters influence the detection capabilities of Slot?
- **RQ6**: How resilient is Slot to adversarial attacks?
- **RQ7**: How effectively does Slot facilitate the manual validation of alerts?

5.1 Experimental Setups

5.1.1 **Datasets.** The evaluation of Slot was carried out using three open-source datasets: StreamSpot [4], Unicorn Wget [20] and DARPA E3 [1]. Both StreamSpot and Unicorn consist of batch audit logs generated in controlled environments. The StreamSpot dataset includes 600 batch logs monitoring system calls across six distinct scenarios, five of which simulate benign user behavior, while one represents a drive-by download attack. The Unicorn dataset contains 150 batch logs collected via Camflow [43], with 125 logs corresponding to benign activity and 25 involving supply chain attacks. These attacks are stealthy in nature, designed to mimic normal system workflows, and are challenging to detect. The DARPA dataset is part of the DARPA Transparent Computing program, gathered during adversarial engagements in enterprise networks. The Red Team conducted APT attacks using various vulnerabilities to exfiltrate sensitive information, while the Blue Team focused on identifying these attacks through network host audits and causal analysis. We conducted experimental comparisons using datasets commonly shared across multiple SOTA algorithms [26, 47, 57, 62] and recognized open-source labels [26, 47].

Table 3: Overview of the experimental datasets.

Dataset	of Nodes	of Edges (in millions)	Attack batch/node	Size (GB)
StreamSpot	999,999	89.8	100 batches	2.8
Unicorn Wget	39,606,900	145.9	25 batches	76.6
DARPA E3-TRACE	3,288,676	4	68,082 nodes	15
DARPA E3-CADETS	1,627,035	2.8	25,319 nodes	18
DARPA E3-THEIA	1,623,966	3.3	12,846 nodes	18

5.1.2 **Baseline Methods.** To compare the efficacy and efficiency of Slot with state-of-the-art methods, we selected the following seven advanced APT detections at both the graph-level, node-level and edge-level granularity for evaluation. Specialized expert rules or threat reports for APT detection schemes are difficult to compare, so they are not taken into consideration.

1) graph-level detection

- **StreamSpot** [36]: It introduces a new similarity function for graphs, comparing them based on the relative frequency of local substructures represented as short strings.
- **Unicorn** [20]: UNICORN uses graph sketching to build an incrementally updatable, fixed size, longitudinal graph data structure 1 that enables efficient computation of graph statistics.

2) node-level detection

- **Threatrace** [57]: It uses GraphSAGE to execute node-level anomaly detection, learning the structural information of nodes and identifying anomalies based on deviations from this learned behavior.
- **Log2vec** [33]: It transforms the user's log entries into heterogeneous graphs based on rules and expresses behavioral relationships through graph embedding.
- **FLASH** [47]: It uses word2vec for semantic learning and GraphSAGE [19] for graph structure to jointly represent log behavior for anomaly detection.
- **MAGIC** [26]: It learns system behavior through a masked autoencoder [25] and then uses KNN to calculate node distances, identifying abnormal nodes.

3) edge-level detection

- **ShadeWatcher** [62]: It leverages TransR [32] and GNN to detect APTs based on recommendation.

5.1.3 **Implementation.** We implement Slot using Pytorch, comprising approximately 3,000 lines of code. All models operate on Python 3.9, utilizing the Gensim library for the Word2Vec model. Regarding hyperparameters, we set all embedding dimensions to 64, and the learning rate to 0.01. We use Adam as the optimizer, with the similarity loss weight (λ_1) set at 2 and the $L2$ regularization weight (λ_2) set at 0.01. The RL action step size (τ) is configured to 0.02. In Section 5.6, we provide a detailed discussion on the impact of hyperparameter settings, offering a thorough analysis and explanation of the rationale behind the settings and the experimental logic. All experiments are performed on a server running Ubuntu

18.04 LTS with an Intel 13th Gen Core i7-1360P CPU (12 cores, 16 threads, base frequency 2.20 GHz), 32 GB RAM.

5.2 Efficacy of Semi-supervision Detection (RQ1)

Table 4: Slot 's detection results on different datasets. For batched log level detection, the detection targets are log pieces. And for system entity level detection, system entities are the targets.

Granularity	Dataset	Ground Truth		#TP	#FP	#TN	#FN
		#Benign	#Malicious				
Batched log	StreamSpot	100	90	89	1	99	1
	Unicorn Wget	25	22	21.8	0.9	24.1	0.2
System entity	DARPA E3 Trace	613,050	60,644	60,604	1,521	611,529	40
	DARPA E3 CADETS	343,428	11,561	11,550	478	342,950	11
	DARPA E3 THEIA	315,212	22,825	22,822	400	314,812	3

Slot performs node-level fine-grained APT detection on batch logs and event-level logs. In order to reasonably perform result statistics at the batch processing level, we calculate the detection performance of Slot based on the proportion of malicious attacks in the batch processing. Slot uses only 10% of the malignant structures through oversampling to provide supervision signals for similarity labels and achieve semi-supervised learning. Table 4 shows that Slot accurately detected APT attacks in various scenarios. A small number of malicious structures effectively assist the model in learning and characterizing benign patterns. The node-level Slot achieves very low false negatives and false positives in fine-grained detection.

Through similarity awareness, Slot can effectively select neighbors that are more similar to the central node, allowing the Graph Neural Network (GNN) to better capture patterns in system behavior. Table 5 demonstrates the advantages of semi-supervised learning. While a greater number of malicious signals lead to significant improvements in performance, the results show that even a small amount of malicious supervision signals is sufficient to train an effective model.

5.3 Overall Detection Efficacy Comparsion (RQ2)

The effectiveness of Slot was tested using both batch and entity-level logs. To demonstrate Slot's precise ability to distinguish between benign and malicious behaviors, a fair comparison was made with relevant SOTA (state-of-the-art) methods, as shown in Table 6. Based on these results, we draw the following observations:

Table 5: Impact of Malignant Structure Percentage in Batch and Event-Level Training Datasets on Detection Efficacy.

Dateset	Train%	Accuracy	Precison	Recall	F1-Score	AUC
StreamSpot	5	0.8924	0.8922	0.9115	0.8911	0.9213
	10	0.9999	0.9999	0.9999	0.9999	0.9999
	20	0.9999	0.9999	0.9999	0.9999	0.9999
	40	0.9999	0.9999	0.9999	0.9999	0.9999
CADETS	5	0.9069	0.9016	0.9203	0.9050	0.9780
	10	0.9986	0.9602	0.9990	0.9792	0.9970
	20	0.9907	0.9895	0.9911	0.9903	0.9981
	40	0.9999	0.9999	0.9999	0.9999	0.9999

Table 6: The comparison results between Slot and state-of-the-art approaches.

Datasets	System	Accuracy	Precison	Recall	F1-Score	FPR
StreamSpot	StreamSpot	93%▼6.06%	73%▼26.2%	91%▼8.08%	81%▼18.2%	6.6%▲3200%
	Unicorn	99%▼0.00%	95%▼4.04%	97%▼2.02%	96%▼3.03%	1.6%▲700%
	Threatrace	99%▼0.00%	98%▼1.01%	99%▼0.00%	99%▼0.00%	0.4%▲100%
	Slot	**99%**▲2.06%	**99%**▲11.7%	**99%**▲3.48%	**99%**▲7.60%	**0.2%**▼93.0%
Unicorn Wget	Unicorn	90%▼9.09%	86%▼10.4%	95%▼4.04%	90%▼7.21%	15.5%▲330%
	Threatrace	95%▼4.04%	93%▼3.12%	98%▼1.01%	95%▼2.06%	7.4%▲105%
	Slot	**99%**▲7.02%	**96%**▲7.26%	**99%**▲2.59%	**97%**▲4.86%	**3.6%**▼68.5%
DARPA E3 TRACE	Log2vec	97.63%▼2.13%	54.41%▼44.2%	78.25%▼21.6%	64.18%▼34.9%	1.8%▲800%
	ThreaTrace	98.92%▼0.84%	71.59%▼26.6%	99.99%▲0.06%	83.43%▼15.4%	1.1%▲450%
	ShadeWatcher	**99.96%**▲0.20%	**99.98%**▲2.49%	**99.99%**▲0.06%	**99.98%**▲1.27%	0.3%▲50%
	FLASH	99.75%▼0.01%	94.66%▼2.96%	**99.99%**▲0.06%	97.20%▼1.53%	0.3%▲50%
	MAGIC	99.04%▼0.72%	92.00%▼5.68%	98.99%▼0.94%	95.43%▼3.33%	0.9%▲350%
	Slot	99.76%▲0.70%	97.55%▲18.2%	99.93%▲4.70%	98.72%▲12.1%	**0.2%**▼77.2%
DARPA E3 CADETS	Log2vec	98.16%▼1.70%	49.20%▼48.7%	84.55%▼15.3%	62.23%▼36.4%	1.6%▲1600%
	ThreaTrace	99.81%▼0.05%	90.42%▼5.83%	99.97%▲0.07%	94.95%▼3.03%	0.2%▲100%
	FLASH	99.73%▼0.13%	93.43%▼2.69%	99.94%▲0.04%	96.58%▼1.36%	0.3%▲200%
	MAGIC	99.18%▼0.68%	82.02%▼14.5%	99.07%▼0.83%	89.75%▼8.34%	0.8%▲700%
	Slot	**99.86%**▲0.64%	**96.02%**▲21.8%	**99.90%**▲4.19%	**97.92%**▲14.0%	**0.1%**▼86.2%
DARPA E3 THEIA	Log2vec	99.55%▼0.33%	62.49%▼36.4%	66.05%▼33.9%	64.23%▼35.2%	0.3%▲150%
	ThreaTrace	99.87%▼0.01%	87.03%▼11.4%	99.74%▼0.24%	92.95%▼6.23%	**0.1%**▼16.6%
	FLASH	99.38%▼0.50%	91.73%▼6.65%	99.82%▼0.16%	95.63%▼3.53%	0.8%▲566%
	MAGIC	99.86%▼0.02%	98.23%▼0.04%	99.99%▼0.00%	99.10%▼0.03%	0.14%▲16.6%
	Slot	**99.88%**▲0.22%	**98.27%**▲15.7%	**99.99%**▲9.39%	**99.13%**▲12.6%	0.12%▼64.2%

Bold denotes the best results, and *underlined* denotes the second-best results.
▼% represents the percentage of decrease, ▲% represents the percentage of improvement.
[1] Improvement compared to the average of all detection systems except Slot.
[2] FPR means False Positive Rate. A detection system with a lower false positive rate can better avoid false positive attacks during detection.

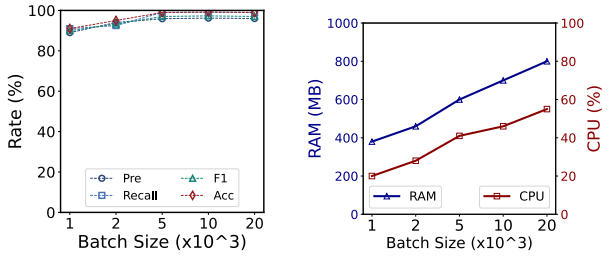
- **Obs.1**: **Comparison in Batch Logs**. For batch audit log detection, we used the datasets provided by Streamspot and Unicorn as baselines and compared the results with node-level detection methods, including Threatrace. In the relatively simple Streamspot logs, where attack patterns are straightforward and behavioral differences are clear, all detection methods performed well. However, Slot achieved near-perfect detection results. On the Unicorn dataset, where highly camouflaged attacks significantly reduced the detection efficacy of other methods, Slot excelled by distinguishing node similarities and strengthening relationship selection. This effectively magnified the differences between benign and malicious behaviors, resulting in superior detection efficacy.
- **Obs.2**: **Comparison in Entity-Level Logs**. In entity-level log detection, we compared Slot with other node-level methods. Log2vec treats logs as nodes rather than entities, fragmenting entity information and weakening aggregation. GNN-based methods like MAGIC, Threatrace, and FLASH also failed to address benign behaviors camouflaging malicious entities, making them vulnerable to evasion. In contrast, Slot used reinforcement learning to filter out benign-camouflaged neighbors, resulting in superior detection efficacy compared to the SOTA node-level system, FLASH. In Figure 6, compared to FLASH, the Slot ROC curve demonstrates superior model classification performance.

5.4 System Overhead (RQ3)

Slot is a detection scheme based on GNN, and therefore, the efficiency overhead during the detection phase depends on the size of the constructed graph, which is determined by the batch size

Table 7: The time cost and proportion of each stage for SLOT on the training graph with 300K nodes.

Phase	Graph Construction	Reinforcement Learning	GNN
Time cost	46.12s	331.94s	64.69s
Percentage	10.41%	74.97%	14.67%



(a) Detection rates with changes in batch size.

(b) RAM and CPU overheads with changes in batch size.

Figure 6: Detection efficacy and efficiency of SLOT across various batch sizes.

parameter. We comprehensively analyzed the changes in efficiency, CPU and memory overhead of SLOT by adjusting the batch size. And compared the time overhead when selecting a batch size of 5000.

Figure 6(a) shows the impact of batch size on detection effectiveness. Since larger graph batches facilitate learning a more complete local structure in GNN processing, the detection efficiency improves as the batch size increases. However, beyond a certain threshold (The maximum number of entities required to complete most of the system activities), the detection capability stabilizes. Considering that batch size has a significant impact on detection accuracy, we analyzed the CPU and memory overhead under different batch sizes. As shown in Figure 6(b), larger batch sizes provide more complex graph structures and node information, which requires more memory and node computation during the graph information propagation process. In terms of detection time, while the introduction of reinforcement learning incurs additional overhead, as shown in Table 7, our tests on graphs with 300K nodes demonstrate that although reinforcement learning consumes more time, the overhead can be considered acceptable.

5.5 Module Ablation Study (RQ4)

In our ablation experiments, we re-evaluate the efficacy of SLOT by varying the content of various components to show their impact on the effectiveness of the system. The results are reported in Figure 7.

Effect of semantic features and topological features. In the similarity-aware module, we conducted ablation studies on semantic and topological structures to explore the value of node semantic information and node topology in the selection of node neighbors. Experimental results show that both have a certain impact on node neighbor aggregation. During feature embedding, integrating semantic features and structural features simultaneously

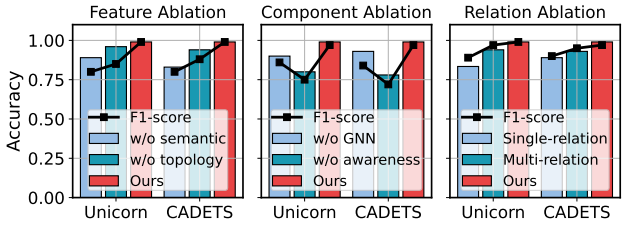


Figure 7: The impact on accuracy and F1 score from separately ablating features, components, and relations, evaluated on the Unicorn and CADETS datasets.

can effectively distinguish between benign and malicious nodes. This approach significantly improves the accuracy of similarity segmentation.

Effect of similarity awareness & GNN. SLOT characterizes node features through similarity awareness and GNN. We conducted ablation tests on the main modules of SLOT, and experiments revealed that using similarity awareness alone cannot deeply explore graph structure information, while GNN without the guidance of neighbor selection is susceptible to the influence of noisy neighbors, misleading node embeddings. Experimental results demonstrate that GNN guided by similarity significantly enhances detection efficacy.

Effect of provenance relation. Current APT detection [47, 57, 62] based on GNN often treat provenance graphs as homogeneous, failing to differentiate the heterogeneity of node neighbors, with all relationships being merged into the central node. As illustrated in Figure 7, we compared the detection efficacy of three methods: homogeneous aggregation, multi-relation aggregation, and deep relational aggregation based on hidden relation mining. The experimental results indicate that neighbor relationship filtering based on reinforced selection effectively reduces the induced expression of benign neighbors on malicious nodes. Additionally, by incorporating hidden relational information, the model can mine deeper feature information, thereby accelerating the effective propagation of information flow and enhancing detection efficiency.

5.6 Hyperparameter Investigation (RQ5)

Previously, we employed a fixed set of optimal parameters; in this section, we discuss the impact of several critical hyperparameters on the threat detection efficacy of SLOT. We examined the logic behind the choice of hyperparameters separately on CADETS shown in Figure 8. When testing one of these parameters, we maintained the others consistent with the baseline. We summarize the results with the following observations:

- **Embedding Size** d: In the embedding process, node features are embedded and mapped to low-dimensional vectors while preserving as much of their original feature attributes as possible. As shown in Figure 8, as the embedding dimension increases, the detection metrics improve significantly, indicating that a larger embedding dimension effectively captures more information. However, an excessively large dimension can lead to feature sparsity, which severely impacts detection efficacy and results

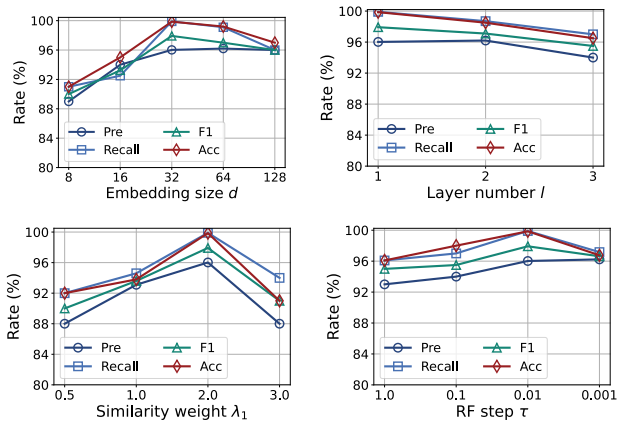


Figure 8: Detection efficacy (i.e., accuracy, precision, recall, F1) with different hyperparameters on the CADETS dataset.

in higher memory consumption. When the dimension is set to 32, the model achieves the fastest convergence and reaches high efficacy metrics.

- **Number of Layers** l: Generally, increasing the number of layers in a model can capture deeper information. However, as shown in Figure 8, there is no significant improvement in efficacy with an increase in the number of layers. Particularly, when the model exceeds three layers, there is a clear occurrence of overfitting. Therefore, choosing a single-layer model can achieve the best balance between detection efficacy and efficiency cost.
- **Similarity Loss Weight** λ_1: Figure 7 demonstrates the significant impact of similarity calculations on Slot. Figure 8 shows the effect of different similarity loss weights on detection efficacy. It is evident that when the weight of the similarity loss is twice that of the GNN loss, there is a balanced integration of feature similarity perception and GNN information aggregation, resulting in optimal efficacy.
- **Reinforcement Learning step** τ: In reinforcement learning, selecting the appropriate action step size allows all thresholds to be updated and converge over multiple epochs. When the thresholds oscillate for several rounds, reaching Formula 9 is the termination condition. Experiments have demonstrated that Slot achieve optimal efficacy when the step size is set to 0.01.

5.7 Resilience to Adversarial Attacks (RQ6)

Adversarial Attack: Mimicry attacks against provenance-based APT detection are carried out by altering the original data to incorporate more benign features, thereby 'mimicking benign behavior' in order to evade detection. To evaluate the resilience of the Slot system against adversarial attacks, we employed the attack methods described in [17] and [47], which involve inserting benign structures into the attack graph. This allowed us to assess the robustness of Slot.

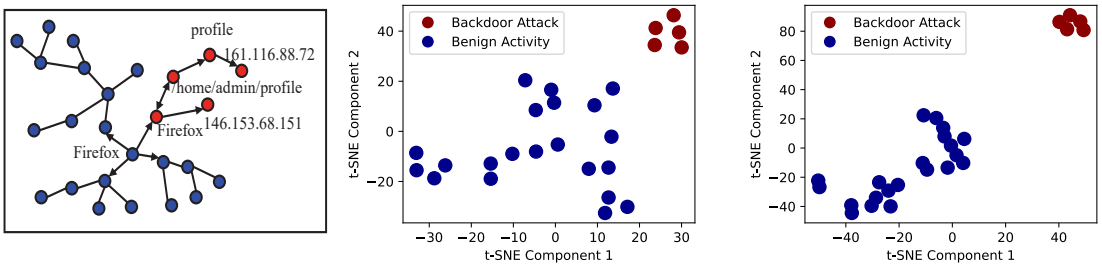
To demonstrate the robustness of Slot against adversarial attacks, we use the THEIA dataset as an example to show how Slot can distinguish between benign imitative structures and malicious behaviors, preventing attackers from misleading the model by mimicking benign behaviors and inducing incorrect representations of malicious nodes. In Figure 9(a), we present the system behavior near the exploitation of a Firefox backdoor vulnerability. The attack activity has minimal interaction with benign behaviors, allowing for clear separation between benign and attack behaviors when using GNN embeddings, as shown in Figure 9(b). In contrast, Figure 9(c) illustrates that Slot, by leveraging semi-supervised similarity labels, enables a greater distinction between different types of nodes in the embedding space. In Figure 9(d), we use the Clone technique to repeatedly insert common benign behaviors (DNS resolution) near attack nodes without affecting the attack activity, in an attempt to interfere with the model's embedding judgment of the attack. As seen in Figure 9(e), the attack behaviors, influenced by benign imitative structures, are mistakenly represented as benign when they are close to benign behaviors. However, as shown in Figure 9(f), Slot counters this by adapting its aggregation weights based on neighborhood similarity, allowing it to selectively aggregate different types of relationships in Firefox's neighborhood. This mechanism enables the model to resist the misleading effects of multiple Clone operations, resulting in a better separation of the embeddings despite the presence of benign structure imitations.

As shown in Figure 10, both Slot and FLASH exhibit a certain level of robustness against imitation attacks when only a small number of benign events are introduced. However, as more benign events are incorporated, we observe a significant decline in FLASH's detection efficacy, while Slot remains unaffected by the imitation attack, demonstrating greater robustness compared to the baseline. We attribute this to FLASH and other graph-based representation learning methods for APT detection, which fail to adequately consider the heterogeneous relationships between nodes during propagation and aggregation. A small number of benign embeddings have minimal impact at the node level, but when more benign structures aggregate near attack nodes, it leads to erroneous representations of the attack nodes. In contrast, Slot effectively guides neighbor selection by distinguishing behavioral similarities, making it resilient to the degree of adversarial influence. By employing multi-armed bandit-based reinforcement learning, Slot dynamically filters out benign camouflage related to malicious nodes, demonstrating strong resistance against imitation attacks.

5.8 Alert Validation (RQ7)

The ultimate goal of Slot is to provide security analysts with more effective and concise attack insights, thereby accelerating the alert-handling process. We believe that APT detection should achieve the following when reconstructing attack chains: (1) Provide concise and readable attack chains, (2) Include as few false positive nodes in the attack chain as possible.

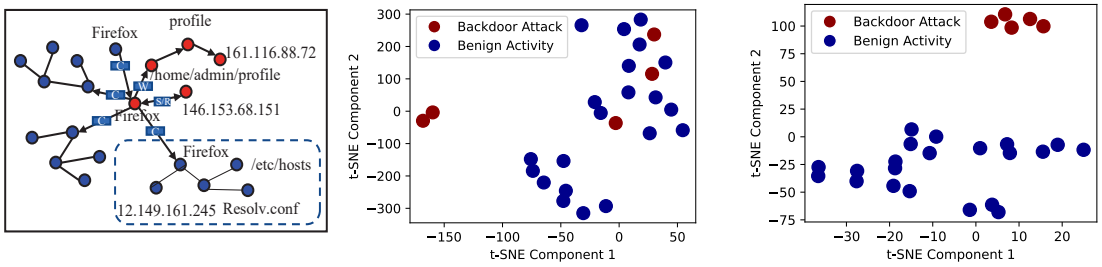
As shown in Figure 11, Slot rapidly filters and identifies the most relevant paths during the attack chain decision-making process, using TTP to exclude paths that may appear related but are common in normal operations. This enhances false positive filtering in attack chains. Additionally, as described by FLASH [47], the reconstructed attack graph exhibits strong event representation, eliminating the need for security analysts to trace numerous anomalous nodes. As shown in Figure 12, the benign structure in CADETS, generated by



(a) System Activities Near Backdoor Attack Events.

(b) Node Embedding of System Events Near Backdoor Attack Activities Detected by FLASH.

(c) Node Embedding of System Events Near Backdoor Attack Activities Detected by SLOT.



(d) System Event Structure After Cloning Multiple Benign DNS Resolution Activities at Attack Nodes.

(e) Node Embedding of System Events Near Backdoor Attack Activities Detected by FLASH under Adversarial Attacks.

(f) Node Embedding of System Events Near Backdoor Attack Activities Detected by SLOT under Adversarial Attacks

Figure 9: t-SNE Visualization of Adversarial Attack Embeddings.

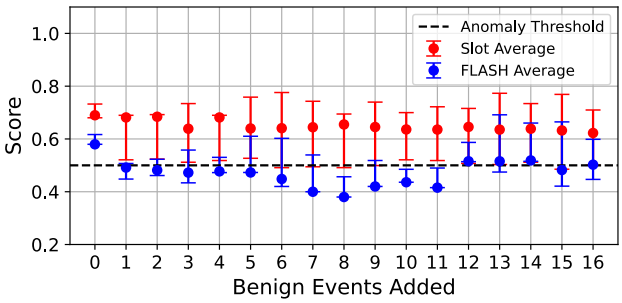


Figure 10: Resilience against adversarial attacks.

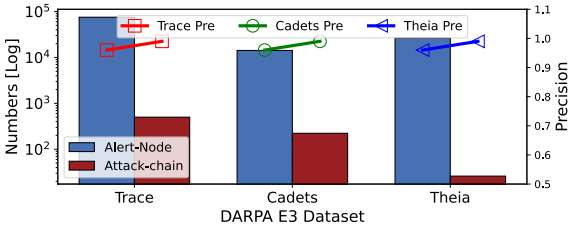


Figure 11: The number of alerts produced with the DARPA E3 dataset and the false positive rate within the generated attack chain.

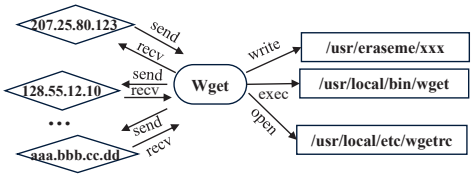


Figure 12: An example of benign activities reconstructed within the alert nodes.

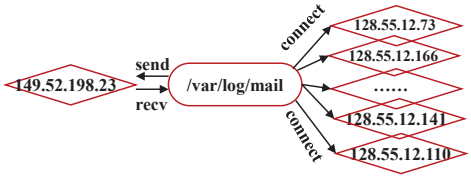


Figure 13: An example of attack activities reconstructed within the alert nodes.

"*wget*", is highly aggregated, accelerating the filtering and selection of benign entity graphs. In Figure 13, the attack scanning behavior in THEIA initiated by the malicious process "*/var/log/email*" involves over 6000 scanned network entities. Aggregating the attack chain reduces the time analysts need to manually trace these

activities. The use of attack chains significantly reduces the items analysts need to review, as they are far fewer than anomalous nodes.

Case Study. Using the attack scenario from Section 3.1, we demonstrate how Slot detects APT attacks from audit logs and reconstructs the attack chain. In this scenario, the attacker leverages a Firefox backdoor and embeds benign DNS resolutions to mask their activities. Slot employs semantic and topological embeddings to derive primary node representations and calculate similarities, distinguishing between benign behaviors (e.g., Firefox accessing common IPs) and malicious activities. By segmenting similarities between Firefox1 and Firefox2 and using reinforcement learning, Slot filters out benign activities disguised within attack events. Using GNN, it learns the genuine behavioral patterns of nodes. In the final Attack-chain Reconstruction phase, TTP encoding further separates benign peripheral nodes, resulting in the red-highlighted attack chain. Core malicious activities include communication with IP 146.153.68.151, downloading and executing a malicious file (*/home/admin/profile*), and interacting with IP 149.52.198.23 to manipulate processes for attack scanning.

5.9 Discussion

Unknown Attacks. Slot uses graph reinforcement learning to model both benign and malicious behaviors. However, APT attacks involve numerous unknown 0-day attacks. Although Slot provides considerations for detecting unknown anomalies, relying solely on feature isolation for anomaly detection tends to result in a high false positive rate, and requires appropriate confidence levels to balance model-based learning detection and anomaly detection. In the future, we aim to extend unsupervised graph reinforcement learning to APT detection.

Concept drift. As user business evolves, system behaviors also change accordingly, leading to new behaviors that the model has not previously encountered. This causes the behavior patterns learned by the model to become outdated. Such concept drift may result in misclassification by the model. Similar to existing strategies [26, 62], Slot can also adopt current small-batch incremental learning methods [52] and historical data forgetting mechanisms [20] to adapt to these changes, thereby enhancing the adaptability and accuracy of the model.

Adversarial attacks. In Section 5.7, we consider mimicry adversarial attack, which is a form of black-box attack. Adversarial attacks also include gradient-based white-box attacks, but these require the attacker to have full access to the target model, including knowledge of the model architecture, weights, and training data, which is often impractical in real-world scenarios. Moreover, existing methods such as adversarial training [53] and gradient masking [35] have shown to be effective defenses, and Slot can integrate and extend these methods accordingly.

6 Conclusion

The persistent and evolving threat posed by advanced persistent threats (APTs) necessitates innovative detection solutions that can navigate the complexities of modern cyberattacks. Existing methods often fall short in addressing challenges like limited detection efficacy, vulnerability to adversarial attacks, and insufficient support for developing defense strategies. To bridge these gaps, we introduced Slot, an APT detection approach leveraging provenance graphs and graph reinforcement learning. Using advanced provenance graph digging, Slot uncovers multi-level hidden relationships among system behaviors. Slot boosts semi-supervised learning with limited labels via efficient label similarity computation, enhancing both detection accuracy and model robustness. Additionally, it leverages graph reinforcement learning to adapt to new user activities and attack strategies, improving accuracy and resilience in adversarial environments. By automatically constructing attack chains, Slot identifies attack paths with precision and supports defense strategy development. Our evaluations show Slot's effectiveness, achieving 99% APT detection accuracy and outperforming state-of-the-art approaches. Further experiments highlight Slot's high efficiency, resilience to adversarial attacks, and ability to support defense strategy development.

7 Acknowledgments

This research is funded by the National Key Research and Development Program of China (2023YFE0111100), Natural Science Basic Research Program of Shaanxi (No. 2025JC-JCQN-073), National Natural Science Foundation of China under Grant (No. 62272370), Young Elite Scientists Sponsorship Program by CAST (2022QNRC001), the China 111Project (No.B16037), Qinchuangyuan Scientist + Engineer Team Program of Shaanxi (No. 2024QCY-KXJ-149), Songshan Laboratory (No. 241110210200), Open Foundation of Key Laboratory of Cyberspace Security, Ministry of Education of China (No.KLCS20240405) and the Fundamental Research Funds for the Central Universities (QTZX23071), the 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 Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, and Ripple under its University Blockchain Research Initiative (UBRI) [16].

A Appendix

A.1 The multi-class classification performance of StreamSpot.

Slot achieves near-perfect attack detection in the StreamSpot dataset. However, since the StreamSpot dataset also contains ground truth for benign activities, I have similarly performed multi-class classification on the node events. As shown in Figure 14, the six behavior classes exhibit clear clustering, which strongly demonstrates Slot's powerful capability in similarity awareness for multi-class recognition.

A.2 Impact of Oversampling Ratio

APT attack logs are a severely imbalanced dataset, with the number of positive samples being much smaller than the number of negative samples. The model may tend to predict the negative class because it better fits the majority class in the data. By oversampling the positive samples, the proportion of positive samples in the training data can be increased, thereby forcing the model to learn how to

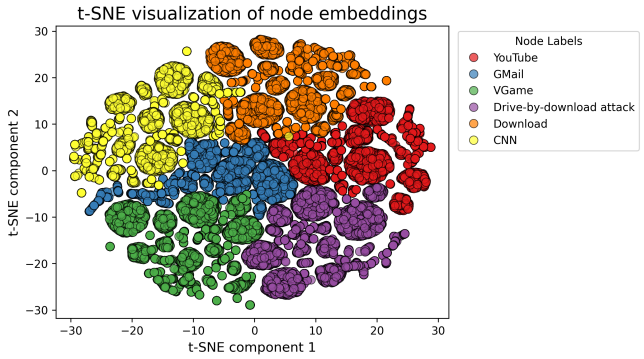


Figure 14: Visualization of t-SNE embeddings in the StreamSpot dataset. The dataset includes five user behavior scenarios and one attack scenario: watching YouTube, checking Gmail, playing VGame, experiencing drive-by-download attack, downloading regular files, and watching CNN. Each point represents a system call behavior from the dataset.

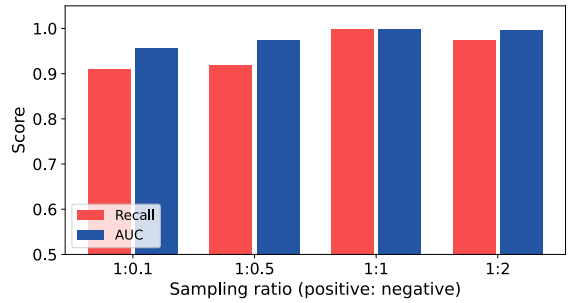


Figure 15: The best AUC and Recall results at different sampling ratios over 30 epochs.

distinguish between the two classes, rather than simply memorizing the negative class. We chose Recall and AUC to measure the impact of sampling ratios on model detection. The reason is that Recall measures the proportion of actual positive samples that are correctly predicted as positive. For oversampling the positive class, Recall reflects how many positive samples the model can identify. AUC measures the model's ability to distinguish between positive and negative samples.The experiment shown in Figure 15 demonstrates that Slot, with a 1:1 positive-to-negative sampling ratio, achieves better detection performance.

References

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

[2] [n. d.]. The Linux audit daemon. https://linux.die.net/man/8/auditd.

[3] [n. d.]. MITRE ATT&CK Framework. https://attack.mitre.org/

[4] [n. d.]. The streamspot dataset. https://github.com/sbustreamspot/sbustreamspot-data. 2016.

[5] Adil Ahmad, Sangho Lee, and Marcus Peinado. 2022. Hardlog: Practical tamper-proof system auditing using a novel audit device. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1791–1807.

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

[7] Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. 2019. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials* 21, 2 (2019), 1851–1877.

[8] Davide Bacciu and Danilo Numeroso. 2022. Explaining deep graph networks via input perturbation. *IEEE Transactions on Neural Networks and Learning Systems* 34, 12 (2022), 10334–10345.

[9] Miaojiang Chen, Wei Liu, Tian Wang, Shaobo Zhang, and Anfeng Liu. 2022. A game-based deep reinforcement learning approach for energy-efficient computation in MEC systems. *Knowledge-Based Systems* 235 (2022), 107660.

[10] Zijun Cheng, Qiujian Lv, Jinyuan Liang, Yan Wang, Degang Sun, Thomas Pasquier, and Xueyuan Han. 2023. Kairos:: Practical Intrusion Detection and Investigation using Whole-system Provenance. *arXiv preprint arXiv:2308.05034* (2023).

[11] Zhangyu Cheng, Chengming Zou, and Jianwei Dong. 2019. Outlier detection using isolation forest and local outlier factor. In *Proceedings of the conference on research in adaptive and convergent systems*. 161–168.

[12] Microsoft Corporation. [n. d.]. Event tracing. https://docs.microsoft.com/en-us/windows/desktop/ETW/event-tracing-portal.

[13] Yebo Feng, Jun Li, Jelena Mirkovic, Cong Wu, Chong Wang, Hao Ren, Jiahua Xu, and Yang Liu. 2025. Unmasking the Internet: A Survey of Fine-Grained Network Traffic Analysis. *IEEE Communications Surveys & Tutorials* (2025).

[14] Yebo Feng, Jun Li, and Thanh Nguyen. 2020. Application-layer DDoS defense with reinforcement learning. In *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–10.

[15] Yebo Feng, Jun Li, Devkishen Sisodia, and Peter Reiher. 2023. On explainable and adaptable detection of distributed denial-of-service traffic. *IEEE Transactions on Dependable and Secure Computing* 21, 4 (2023), 2211–2226.

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

[17] Akul Goyal, Xueyuan Han, Gang Wang, and Adam Bates. 2023. Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems. In *30th Network and Distributed System Security Symposium*.

[18] Qian Gui, Hong Zhou, Na Guo, and Baoning Niu. 2024. A survey of class-imbalanced semi-supervised learning. *Machine Learning* 113, 8 (2024), 5057–5086.

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[20] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. 2020. UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats. In *27th Annual Network and Distributed System Security Symposium, NDSS*.

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

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

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

[24] Md Nahid Hossain, Sanaz Sheikhi, and R Sekar. 2020. Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1139–1155.

[25] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 594–604.

[26] Zian Jia, Yun Xiong, Yuhong Nan, Yao Zhang, Jinjing Zhao, and Mi Wen. 2024. MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning. In *33rd USENIX Security Symposium (USENIX Security 24)*. 5197–5214.

[27] Chao Jiang, Yi He, Richard Chapman, and Hongyi Wu. 2022. Camouflaged poisoning attack on graph neural networks. In *Proceedings of the 2022 International Conference on Multimedia Retrieval*. 451–461.

[28] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. 2018. Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202* (2018).

[29] Maya Kapoor, Joshua Melton, Michael Ridenhour, Siddharth Krishnan, and Thomas Moyer. 2021. PROV-GEM: automated provenance analysis framework using graph embeddings. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 1720–1727.

[30] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2018. Learning hand-eye coordination for robotic grasping with deep learning

and large-scale data collection. *The International journal of robotics research* 37, 4-5 (2018), 421–436.

[31] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. 2021. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems* 34 (2021), 20887–20902.

[32] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29.

[33] Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. 2019. Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 1777–1794.

[34] Yushan Liu, Mu Zhang, Ding Li, Kangkook Jee, Zhichun Li, Zhenyu Wu, Junghwan Rhee, and Prateek Mittal. 2018. Towards a Timely Causality Analysis for Enterprise Security.. In *NDSS*.

[35] Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *stat* 1050, 9 (2017).

[36] Emaad Manzoor, Sadegh M Milajerdi, and Leman Akoglu. 2016. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1035–1044.

[37] Tomas Mikolov. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

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

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

[40] Dorcas Ofori-Boateng, I Segovia Dominguez, C Akcora, Murat Kantarcioglu, and Yulia R Gel. 2021. Topological anomaly detection in dynamic multilayer blockchain networks. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21*. Springer, 788–804.

[41] Riccardo Paccagnella, Pubali Datta, Wajih Ul Hassan, Adam Bates, Christopher Fletcher, Andrew Miller, and Dave Tian. 2020. Custos: Practical tamper-evident auditing of operating systems using trusted execution. In *Network and distributed system security symposium*.

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

[43] Thomas Pasquier, Xueyuan Han, Thomas Moyer, Adam Bates, Olivier Hermant, David Eyers, Jean Bacon, and Margo Seltzer. 2018. Runtime analysis of whole-system provenance. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 1601–1616.

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

[45] Hao Peng, Ruitong Zhang, Yingtong Dou, Renyu Yang, Jingyi Zhang, and Philip S Yu. 2021. Reinforced neighborhood selection guided multi-relational graph neural networks. *ACM Transactions on Information Systems (TOIS)* 40, 4 (2021), 1–46.

[46] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems* 8, 7 (2009), 579–588.

[47] Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. 2024. FLASH: A Comprehensive Approach to Intrusion Detection via Provenance Graph Representation Learning. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 139–139.

[48] SektorCERT. 2023. The attack against Danish critical infrastructure. Available online. https://sektorcert.dk/wp-content/uploads/2023/11/SektorCERT-The-attack-against-Danish-critical-infrastructure-TLP-CLEAR.pdf

[49] Amit Sharma, Brij B Gupta, Awadhesh Kumar Singh, and VK Saraswat. 2023. Advanced persistent threats (apt): evolution, anatomy, attribution and countermeasures. *Journal of Ambient Intelligence and Humanized Computing* 14, 7 (2023), 9355–9381.

[50] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Yuanxing Ning, Philip S Yu, and Lifang He. 2021. Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In *Proceedings of the web conference 2021*. 2081–2091.

[51] Threat Hunter Team. 2023. *Grayling: Previously Unseen Threat Actor Targets Multiple Organizations in Taiwan*. https://symantec-enterprise-blogs.security.com/threat-intelligence/grayling-taiwan-cyber-attacks

[52] Songsong Tian, Lusi Li, Weijun Li, Hang Ran, Xin Ning, and Prayag Tiwari. 2024. A survey on few-shot class-incremental learning. *Neural Networks* 169 (2024), 307–324.

[53] Florian Tramer and Dan Boneh. 2019. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems* 32 (2019).

[54] Victor Uc-Cetina, Nicolás Navarro-Guerrero, Anabel Martin-Gonzalez, Cornelius Weber, and Stefan Wermter. 2023. Survey on reinforcement learning for language processing. *Artificial Intelligence Review* 56, 2 (2023), 1543–1575.

[55] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[56] Qi Wang, Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Junghwan Rhee, Zhengzhang Chen, Wei Cheng, Carl A Gunter, et al. 2020. You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis.. In *NDSS*.

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

[58] Rui Wen, Jianyu Wang, Chunming Wu, and Jian Xiong. 2020. Asa: Adversary situation awareness via heterogeneous graph convolutional networks. In *Companion Proceedings of the Web Conference 2020*. 674–678.

[59] Zhiwen Xie, Runjie Zhu, Jin Liu, Guangyou Zhou, and Jimmy Xiangji Huang. 2022. An efficiency relation-specific graph transformation network for knowledge graph representation learning. *Information Processing & Management* 59, 6 (2022), 103076.

[60] Fan Yang, Jiacen Xu, Chunlin Xiong, Zhou Li, and Kehuan Zhang. 2023. {PROGRAPHER}: An Anomaly Detection System based on Provenance Graph Embedding. In *32nd USENIX Security Symposium (USENIX Security 23)*. 4355–4372.

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

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

[63] Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. 2022. Hierarchical graph transformer with adaptive node sampling. *Advances in Neural Information Processing Systems* 35 (2022), 21171–21183.

[64] Elena Zheleva and Lise Getoor. 2009. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*. 531–540.

[65] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems* 33 (2020), 7793–7804.