Brewing Vodka: Distilling Pure Knowledge for Lightweight Threat Detection in Audit Logs

Anonymous Author(s)*

ABSTRACT

Advanced Persistent Threats (APTs) are continuously evolving, leveraging their stealthiness and persistence to put increasing pressure on current provenance-based Intrusion Detection Systems (IDS). This evolution exposes several critical issues: (1) The dense interaction between malicious and benign nodes within provenance graphs introduces neighbor noise, hindering effective detection; (2) The complex prediction mechanisms of existing APTs detection models lead to the insufficient utilization of prior knowledge embedded in the data; (3) The high computational cost makes detection impractical.

To address these challenges, we propose Vodka, a lightweight threat detection system built on a knowledge distillation framework, capable of node-level detection within audit log provenance graphs. Specifically, Vodka applies graph Laplacian regularization to reduce neighbor noise, obtaining smoothed and denoised graph signals. Subsequently, Vodka employs a teacher model based on GNNs to extract knowledge, which is then distilled into a lightweight student model. The student model is designed as a trainable combination of a feature transformation module and a personalized PageRank random walk label propagation module, with the former capturing feature knowledge and the latter learning label and structural knowledge. After distillation, the student model benefits from the knowledge of the teacher model to perform precise threat detection. Finally, Vodka reconstructs attack paths from anomalous nodes, providing insight into the attackers' strategies. We evaluate Vodka through extensive experiments on three public datasets and compare its performance against several state-of-the-art IDS solutions. The results demonstrate that Vodka achieves outstanding detection accuracy across all scenarios and the detection time is 1.4 to 5.2 times faster than the current state-of-the-art methods.

CCS CONCEPTS

• **Security and privacy** → **Intrusion detection systems**.

KEYWORDS

Threat Detection, Host Provenance, Knowledge Distillation

1 INTRODUCTION

Advanced Persistent Threats (APTs)[6] represent a complex form of cyber attack characterized by high stealth and strong targeting. In these attacks, perpetrators gain unauthorized access to a victim's machine through methods such as network or software backdoors, and persist for extended periods to steal sensitive data or take control of the target machine. APTs have already infiltrated many highly secured large enterprises and institutions that are based on web services, causing substantial financial losses[3, 25, 38]. Notable examples include the Equifax[2] breach, which resulted in a record number of user data being stolen, and the SolarWinds[5] attack, which had a vast scope and severe impact.

To combat these sophisticated APTs attacks, host-based Intrusion Detection intrusion detection systems (IDS) have become a widely deployed defense mechanism. However, with the evolving nature of attack techniques and the growing scale of threats, traditional IDS solutions can no longer meet the demands of the changing threat landscape. Currently, provenance -based[28] detection methods are considered effective for capturing APTs. These methods apply system audit logs and structure system entities (such as processes, files, and network flows) into a graph structure known as a provenance graph. Based on how audit logs are utilized, these detectors can be divided into three categories: statistics-based detection[17, 18, 30] quantifies the suspiciousness of audit logs by analyzing the rarity in the provenance graph; rule-based detection[16, 20, 32] matches audit logs with attack patterns using expert security knowledge bases; and learning-based detection[10, 15, 23, 37, 45, 50] employs machine learning techniques to learn from the provenance graph, identifying abnormal system behaviors and attack patterns. Among these approaches, learning-based detection has been considered the most promising in recent years. However, we have observed several persistent challenges that impact real-world detection:

- **Neighbor Noise**: In provenance graphs constructed from audit logs, malicious nodes often exhibit long-distance, multi-hop distributions, meaning that a complete attack can involve more benign system entities. The interaction between malicious and benign nodes generates interference and confounds the classifier in detection systems. For node-level classification tasks, the neighboring nodes adjacent to the target detection node introduce dense neighbor noise, which can obscure true anomalies, leading to false positives or missed detections, and ultimately reducing detection accuracy.
- **High Computational Cost**: Graph-based algorithms enable learning from provenance graphs to capture the complex relationships within them. While these methods can achieve impressive detection performance, they come at the cost of significant memory and time overhead. As a result, these approaches lose the capability for real-time detection and can only function as offline systems to analyze graph data[45, 50].
- **Insufficient Utilization of Prior Knowledge**: Recent APTs detection methods[10, 23, 50] primarily rely on large-scale graph algorithms, such as GNNs, to extract structural and node feature information from graphs. However, the prediction mechanism of GNNs is highly complex, as it tightly integrates graph topology, node features, and projection matrices. This entanglement makes it challenging to clearly interpret the relationships between these factors within the model. This complexity prevents the effective utilization of prior knowledge in terms of labels, features, and structure[27, 43].

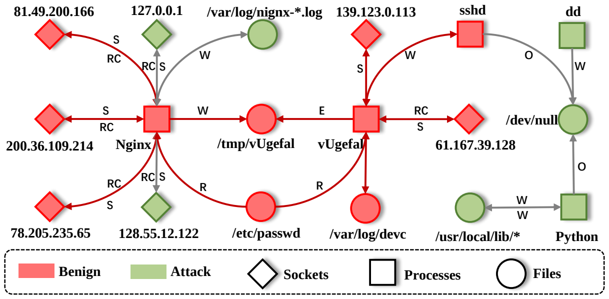


Figure 1: Attack scenario from DARPA E3 CADETS. Green indicates benign entities and red signifies malicious entities. R = Read, W = Write, O = Open, E = Execute, S = Send and Rc = Receive.

Thus, we propose Vodka, a lightweight APTs detection method that achieves both high detection accuracy and computational efficiency. Specifically, Vodka incorporates the following key functionalities: **(1)** Vodka constructs a provenance graph from audit logs that include various APTs attacks and uses Word2Vec to assign initial features to nodes. To address the issue of neighbor noise, Vodka designs a graph Laplacian regularization-based graph signal denoising method tailored to the provenance graph. This algorithm smooths node signals without disrupting the original graph topology. **(2)** Vodka introduces a knowledge distillation framework where, during training, knowledge from a pre-trained large GNN teacher model is distilled into a lightweight student model. The lightweight student model is then used for anomalous node detection, significantly reducing detection costs. **(3)** For the student model, Vodka proposes a new hybrid mechanism that combines the Feature Transformation (FT) mechanism with a personalized PageRank random walk label propagation (PRL). This allows more effective learning of prior knowledge from features, labels, and structures while utilizing the GNN knowledge from the teacher model. **(4)** After detecting anomalous nodes, Vodka uses a community division algorithm to identify malicious communities.

We performed a comprehensive evaluation of Vodka using datasets widely adopted by the research community, including StreamSpot[1], Unicorn Wget[15], and DARPA-E3[4]. We first analyzed its detection performance and compared it against various existing baselines, demonstrating that Vodka consistently outperforms almost all these systems under the same evaluation metrics. Furthermore, we analyzed Vodka's detection costs and explored its potential as a real-time detection system. In addition, hyperparameter analysis and ablation experiments on Vodka highlight the irreplaceability of its key parameters and main components. Lastly, we verified Vodka's inherent robustness against adversarial attacks. The primary contributions of our work are as follows:

- We propose Vodka, a general APTs detection method based on knowledge distillation, capable of transferring GNN knowledge from a large teacher model to a lightweight student model, thereby reducing resource and time costs during detection.
- We address the issue of neighbor noise in provenance graphs by designing a graph Laplacian regularization-based graph signal denoising method that smooths and denoises signals tailored for provenance graphs.
- We examine the problem of insufficient utilization of prior knowledge caused by complex prediction mechanisms in current approaches and design a lightweight student model to effectively learn and leverage prior knowledge.
- We comprehensively evaluate Vodka 's performance by implementing it on three widely used datasets and conducting various experiments to verify its effectiveness. Experimental results demonstrate that Vodka outperforms most existing methods while incurring lower detection costs.

Availability. We released the implementation of the core components of the model at https://anonymous.4open.science/r/Vodka-1E76 (Anonymized).

2 ATTACK SCENARIOS AND THREAT MODEL

2.1 Motivating Example

This is an attack case from the DARPA E3 CADETS: where the attacker establishes a connection with the Nginx server via IP address 81.49.200.166 and gains shell access. Using the shell, the attacker downloads a malicious payload to /tmp/vUgefal and executes it. The process vUgefal then moves laterally to 61.167.39.128, completes the infection, and further injects sshd into the system, downloading another payload to /var/log/devc. Figure 1 illustrates our motivating example, where three types of nodes represent system entities, and the lines indicate the directional interactions between them.

Table 1: Limitations of existing provenance-based intrusion detection systems.

Model	Neighbor Denoising	Lightweight Models	Utilization of Prior Knowledge	Attack Reconstruction	Granularity
Vodka (ours)	✓	✓	✓	✓	Node
MAGIC[23]	✓	✗	✗	✗	Node
KAIROS[10]	✓	✗	✗	✓	Node
ThreaTrace[45]	✗	✗	✗	✗	Node
ShadeWatcher[50]	✗	✗	✗	✗	Edge
Unicorn[15]	✗	✗	✗	✗	Graph

2.2 Limitations to Existing Solutions

Learning-based detection methods are currently the most promising in the field of threat detection. Their core principle is to build anomaly models by learning from large datasets and then infer whether anomalous behavior represents a malicious attack. Some of the most notable recent works include: **MAGIC[23]**, which performs efficient detection by masking the graph and reducing costs; **KAIROS[10]**, which identifies four dimensions of intrusion detection systems and uses an encoder-decoder architecture to detect attacks and reconstruct attack chains; **THREATRACE[45]**, the first detector to propose behavior pattern construction for each type of node in the provenance graph to achieve node-level detection; **ShadeWatcherShadeWatcher[50]**, which introduces recommendation system techniques into provenance graph detection and achieves edge-level detection; **Unicorn[15]**, which builds a provenance graph enriched with semantics and historical information to capture long-term stealthy attacks. Although these works demonstrate excellent performance in threat detection, they still face limitations when tested in real-world environments. These limitations are what Vodka aims to address, as shown in Table 1.

- **Neighbor Denoising**: Provenance graphs constructed from audit logs contain rich information, with nodes and edges representing different meanings. However, current approaches seem too eager to move directly to graph representation learning without adequately addressing the noise in the initial graph. We observed that only Magic acknowledged this issue.
- **Lightweight Models**: One of the major challenge with current log provenance algorithms lies in the large size and difficulty of deployment of the models. Most existing approaches rely on GNN models, which, due to their multi-layer architectures and large parameter inputs, result in bloated models. Furthermore, the high memory and time overhead makes these detectors impractical and limits their deployment in real-world environments. Notably, only Magic[23] has optimized performance costs to address this issue.
- **Utilization of Prior Knowledge**: Labels, node features, and graph structures offer rich prior knowledge that can significantly enhance model performance. Label information helps the model capture the true class relationships of system entities. Node features reflect the attribute differences among system entities, and the graph structure reveals the topological relationships and latent patterns between interactions. However, existing APTs detection methods often over-rely on complex network topologies and automated feature learning, overlooking the explicit prior knowledge embedded in these components.
- **Attack Reconstruction**: A simple and complete attack chain helps security analysts better understand the attack behavior. However, from a large number of anomalous nodes detected at the node level, we need to reconstruct these attack paths based on the dependency relationships between kernel objects, as exemplified by KAIROS, to enable analysts to respond quickly.
- **Granularity**: Different detection granularities yield different levels of attack discovery. For example, graph-level granularity, as in Unicorn, is easy to partition but lacks the precision to pinpoint specific malicious entities in the graph. Edge-level granularity, as in ShadeWatcher, offers more detailed insights into specific edge relationships but incurs higher detection costs. Node-level granularity methods like Magic, KAIROS, and THREATRACE, can target specific entities without requiring detailed learning of edge relationships.

2.3 Threat Model

To build a more comprehensive threat model, we make several assumptions about attacker behavior and system characteristics. We consider APTs attackers, who aim to infiltrate a target host through network channels, pre-installed backdoors, or software vulnerabilities. This process is often highly stealthy and can persist for an extended period. During the intrusion, attackers attempt to blend their malicious activities with legitimate background data to obscure their intent. For instance, they may use Living-off-the-land techniques[33], injecting malicious code into legitimate processes, which then spawn new legitimate processes to continue their malicious actions.

Although the attackers' behavior is highly covert, audit logs capture complete, traceable evidence and footprints of their activities. As in most of the existing works in this field[15, 37, 45], we assume that the underlying operating system, audit framework, and system hardware are intact and trustworthy[8, 35]. Additionally, we assume that attackers cannot directly tamper with the contents of the audit logs, ensuring that the provenance graph constructed by Vodka remains reliable[34].

3 METHODOLOGY

In this section, we provide a detailed overview of Vodka's overall design framework and the design of each module. The framework consists of the following components: ①**Graph Construction**. ②**Log Distillation**. ③**Threat Detection**. ④**Attack Reconstruction**. The overall framework is illustrated in Figure 2.

3.1 Graph Construction

3.1.1 Provenance Graph Construction.

Vodka processes host audit logs from various sources(such as Windows ETW, Linux Audit and CamFlow[35]) and transforms them into a graph structure known as a Provenance Graph (PG). In this graph, nodes represent system entities, and edges denote the interactions between these entities.

Since system logs contain rich attributes related to various system entities, Vodka encodes these attributes into a vector space to serve as initial input for model learning. We use Word2Vec[31], a neural network-based technique effective at learning word vector representations, which generates dense, low-dimensional vectors for each word. In our cataloging process, we use attributes such as file paths for files, process names for process nodes, system entity types, and network IP addresses for sockets to generate feature vectors. These are combined with semantic attributes and system call types within one hop of the neighbors to form sentences. During training, Vodka captures semantic relationships between words within a fixed length, outputting embedded vectors as the raw signal for subsequent modules.

3.1.2 Neighbor Denoising.

Given that in the provenance graph constructed from audit logs, malicious nodes (i.e., system entities manipulated by attackers) are rare, and adversarial activities—such as deceptive access to benign nodes or the manipulation of compromised benign nodes to perform misleading actions—are common, there exists a significant issue of neighbor noise for anomaly detection tasks. To address this, Vodka incorporates a graph Laplacian regularization-based signal denoising method, which is designed to leverage the graph's topological structure to smooth signals and reduce the impact of noise.

Specifically, Vodka defines the weights based on the similarity and connection relationships between entity nodes and combines them into a weight matrix W. By calculating the node degree matrix D, we can then derive the Laplacian matrix $L = D - W$. This enables the formulation of a convex quadratic optimization problem to minimize noise and smooth signals as follows:

$$\min_{\mathbf{x}} \left\{ \frac{1}{2} \|\mathbf{x} - \mathbf{x}^{(0)}\|^2 + \frac{\gamma}{2} \mathbf{x}^\top \mathbf{L} \mathbf{x} \right\}. \tag{1}$$

Where $|| \cdot ||$ represents the Euclidean norm, x is the raw signal, and γ is the regularization parameter that controls the trade-off between data fidelity and signal smoothing. To achieve this, we solve the following linear system to find the optimal solution and

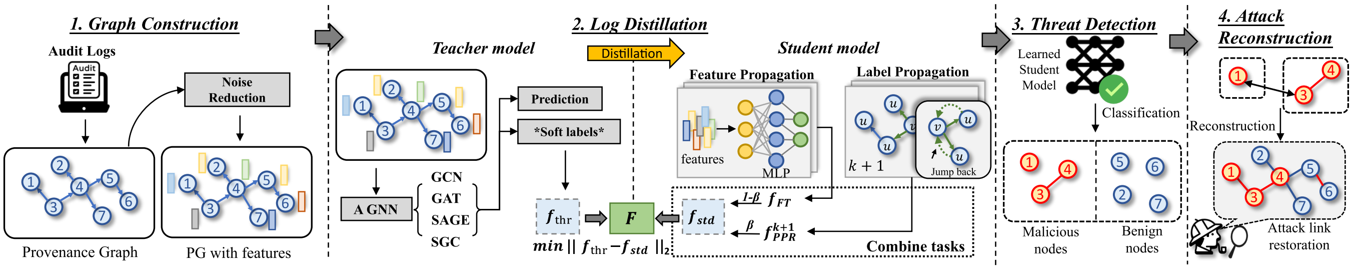


Figure 2: The framework of Vodka.

obtain the denoised node signals:

$$(\mathbf{I} + \gamma \mathbf{L})\mathbf{x} = \mathbf{x}^{(0)}, \quad \mathbf{x} = (\mathbf{I} + \gamma \mathbf{L})^{-1}\mathbf{x}^{(0)}. \tag{2}$$

After the neighbor denoising process, we obtain denoised and smoothed signal vectors. It is worth noting that Vodka's neighbor denoising algorithm does not alter the graph's topology in any way, ensuring that no new noise is introduced due to changes in the structure. This preserves the integrity of the original graph topology while effectively reducing the noise, allowing for more accurate and reliable detection in subsequent steps.

3.2 Log Distillation

In this section, we describe the knowledge distillation mechanism designed for Vodka 's log provenance graph, which explains the training process of the teacher and student models and how knowledge is distilled from the large-scale teacher model to the lightweight student model.

3.2.1 Teacher Model.

The teacher model is the cornerstone of Vodka 's APTs threat detection system, essential for reducing detection time costs. The teacher model is pre-trained before real-time detection occurs. For Vodka 's teacher model, we employ various large-scale GNN models, such as GCN[26], GAT[42], GraphSAGE[14], and SGC[46]. These GNN models are well-suited for leveraging the complex interactions between system entities found in audit logs and capturing causal connections and semantic information across entities. This characteristic is commonly utilized by the most advanced IDS today.

Unlike other IDS, Vodka 's teacher model is trained using supervised labels on system entities fed into the GNN model, producing both the predicted result f_thr and soft labels. Soft labels represent the aggregated anomaly probabilities for all test nodes and serve as an important self-supervised signal for the subsequent student model. By pre-training the teacher model on the large-scale graph topology derived from the log data and generating soft labels, Vodka ensures robust support for real-time detection.

3.2.2 Student Model.

The core requirement for Vodka 's student model is to compress detection time while maintaining high accuracy and effectively utilizing prior knowledge. Thus, the student model is designed to be lightweight and efficient, built on two foundational mechanisms: 1) feature transformation for individual nodes, and 2) label propagation[21] between nodes. The former focuses on enhancing the model's ability to fit based on each node's intrinsic features, while the latter propagates label information across nodes.

Vodka 's student model integrates both mechanisms, combining Feature Transformation **(FT)** and Personalized PageRank Random walk Label propagation **(PRL)**. Feature Transformation enables each node to adjust and refine its feature representation to improve detection accuracy, while PRL helps spread labels effectively across the graph to capture anomaly information in a more efficient and informed manner. This hybrid approach allows the student model to leverage the knowledge distilled from the teacher model while significantly reducing detection time and computational costs.

1. Feature Transformation We use a simple Multilayer Perceptron (MLP)[39] for feature transformation to perform nonlinear transformations on the input node features, enhancing the model's ability to fit complex data. In general, MLP takes the raw node features as input and outputs the predicted probability for malicious nodes. Vodka employs a two-layer MLP.

Assuming the input feature vector for a node is x_v, with an input dimension of d, the MLP transforms it into a feature vector h_v as follows:

$$h_v = \sigma(W_1 X_v + b_1). \tag{3}$$

The prediction result from the FT module is

$$f_{\text{FT}}(v) = \text{softmax}(W_2 h_v + b_2), \tag{4}$$

where W_1 and W_2 are the learnable weight matrices corresponding to the first and second linear transformations, respectively. b_1 and b_2 are the bias terms, and σ represents a non-linear activation function. The output layer passes through a softmax function to generate the final prediction.

2.Label Propagation We designed a label propagation method based on Personalized PageRank[7] in the form of random walks[47]. This approach propagates labels from labeled nodes to unlabeled nodes by simulating random walks between them. PRL introduces a "jump-back" mechanism, which allows the label information to propagate from neighbor nodes while also having a probability of returning to the starting node, preserving more local information.

The process can be broken down into the following steps:

1) Label Initialization For any node $v \in V$, its initial label is mapped to a probability distribution based on the supervised signal from the labeled data.

2) Iterative Propagation In each iteration, PPR updates the label distribution of each node based on its current distribution and that of its neighbors. Node v retains its previous label distribution from the last iteration (denoted as iteration k) with a probability of $1 - \alpha$, meaning that it primarily relies on its own information without depending on its neighbors.

Then, with a probability α, node v receives label information from its neighboring nodes N_v. The label contribution from each neighbor $u \in N_v$ is an equally weighted average of the random walk label distribution f_{RW}. This ensures that each neighbor node contributes equally to the label propagation process, with all neighbors participating equally in updating the label distribution.

Finally, the updated label distribution $f_{\mathrm{RW}}^{k+1}(v)$ for node v at iteration $k+1$ is the sum of its retained distribution and the propagated distribution from its neighbors:

$$f_{\mathrm{RW}}^{k+1}(v) = (1-\alpha) f_{\mathrm{RW}}^{k}(v) + \alpha \sum_{u \in N_v} \frac{f_{\mathrm{RW}}^{k}(u)}{|N_v|}. \tag{5}$$

where $f_{\mathrm{RW}}^{k}(v)$ represents the label distribution of node v during the k-th iteration, α is a parameter that controls the balance between jumping back and propagating the label, N_v is the set of neighboring nodes of node v, and $|N_v|$ is the number of neighbors of node v, and $f_{\mathrm{RW}}^{k}(u)$ is the label distribution of node u.

3) Stopping Iteration The iteration stops when a fixed number of iterations is reached or when the label distribution converges. The result $f_{\mathrm{PRL}}^{k+1}(v) = f_{\mathrm{RW}}^{k+1}(v)$ is then returned.

3. Combine tasks After completing both FT and PRL, we combine them to form the complete student model task. During the learning process for each node v, we dynamically balance FT and PPR to make predictions. The prediction function is as follows:

$$f_{std}^{k+1}(v) = \beta_v f_{FT}(v) + (1-\beta_v) f_{\mathrm{PRL}}^{k+1}(v) \tag{6}$$

Where $f_{FT}(v)$ is the prediction from the Feature Transformation module, $f_{\mathrm{PRL}}^{k+1}(v)$ is the result from the label propagation (PPR), and β is a balancing parameter that controls the contribution of each component to the final prediction.

3.2.3 Optimization Objective. The training logic of both the teacher model and the student model has been presented. In summary, the teacher model distills knowledge obtained from large-scale GNN training, while the student model is trained to mimic the soft label predictions of the pre-trained teacher model. Thus, our optimization objective is to make the student model's prediction $f_{std}(v)$ approximate the teacher model's soft labels $f_{thr}(v)$. The optimization goal is expressed as:

$$\min_{\Theta} \sum_{v \in V} distance(f_{thr}(v), f_{std;\Theta}(v)) \tag{7}$$

Assuming that the student model performs K iterations, the final optimization objective F can be expressed as:

$$\min_{\Theta} \sum_{\forall v \in V\}_U} \| f_{thr}(v) - f_{std;\Theta}^{K}(v) \|_2 \tag{8}$$

In this context, $||\cdot||$ represents the L2 norm, and the parameter set θ includes the balancing parameters between PPR and FT, denoted as $\beta_v, v \in V$, the label jump-back and propagation balance parameter α in PPR, and the parameters θMLP from the MLP in the FT module.

3.3 Threat Detection

Our student model operates in two modes: a training mode and a detection mode. As shown in Figure 3, during the knowledge distillation process, we transfer knowledge from the complex teacher model to the lightweight student model. In this phase, the student model is in training mode, focusing on learning the distilled knowledge from the teacher model. Once the knowledge distillation step is complete, Vodka employs the trained student model as a detection model, which infers and classifies anomalous nodes from the provenance graph constructed from logs. At this point, the student model switches to detection mode. The detection mode workflow is as follows: we input the provenance graph (with initial features) to the student model. After lightweight K-layer PRL and feature transformation steps, the student model predicts an anomaly score for each node v. The student model efficiently passes messages and aggregates information across the input graph, classifying nodes based on the learned lightweight feature representations. The anomaly score output for each node is then compared to a pre-set threshold, and nodes exceeding this threshold are flagged as potential malicious nodes and collected into the malicious node set.

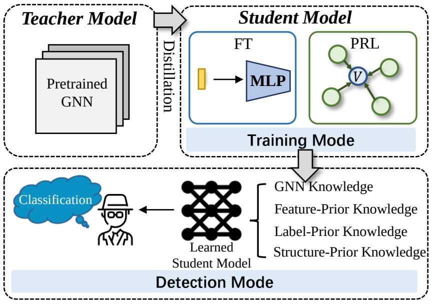


Figure 3: Vodka working modes.

3.4 Attack Reconstruction

After threat detection, security personnel may have a set of detected malicious nodes provided by Vodka, but these nodes are often dispersed across the graph, making it difficult to directly derive the attacker's path. To alleviate the burden on security analysts, Vodka reconstructs a more detailed attack trace, capturing the complete attack chain without relying on prior knowledge. Specifically, Vodka leverages Infomap[12], an algorithm based on random walks that simulates how information propagates between nodes and divides the graph into tightly connected communities. Malicious nodes and their one-hop neighbors are assigned higher weights to distinguish them from benign node connections. Next, information flow propagation is performed, where information spreads across the graph according to weighted probabilities, merging nodes to form communities.

The goal is to maximize 'intra-community propagation' while minimizing 'inter-community propagation'. If most malicious nodes cluster within a single community, it indicates that the community may represent a core region of the APTs attack. If malicious nodes are distributed across multiple communities, it suggests that they likely serve as bridge nodes, linking different regions along the attack path.

4 EVALUATION

In this section, we will conduct various experiments to validate the advantages of Vodka and answer the following research questions:

Table 2: Detection results with teacher models as GCN and GAT.

Datasets	Teacher (GCN)	Student				+ACC Impv.	Teacher (GAT)	Student				+ACC Impv.
	ACC	ACC	PR	RC	F1		ACC	ACC	PR	RC	F1	
StreamSpot	98.30%	99.57%	99.28%	99.81%	99.54%	1.27%	97.93%	99.82%	99.21%	99.35%	99.28%	1.89%
Unicorn Wget	98.21%	99.16%	99.63%	99.76%	99.69%	0.95%	98.27%	99.75%	99.34%	99.82%	99.58%	1.48%
DARPA CADETS	97.31%	99.77%	95.31	99.44%	97.33%	2.46%	96.56%	99.33%	95.53%	99.39%	97.42%	2.77%
DARPA TRACE	96.73%	98.35%	95.73	99.02%	97.34%	1.62%	96.22%	98.98%	95.11%	99.40%	97.21%	2.76%
DARPA THEIA	97.26%	99.94%	96.21	99.74%	97.94%	2.68%	97.43%	99.64%	96.40%	99.61%	97.98%	2.21%

- **RQ1**: How does VODKA 's detection performance compare to state-of-the-art methods?
- **RQ2**: How effective are the main components of VODKA?
- **RQ3**: How do VODKA 's hyperparameters affect its detection performance?
- **RQ4**: What are the cost overheads of VODKA as a detection system?
- **RQ5**: How robust is VODKA against adversarial attacks?

4.1 Datasets Settings

4.1.1 Datasets.

StreamSpot Dataset[1]: This dataset comprises provenance graphs collected by SystemTap from six controlled environments, including CNN, Download, Gmail, Vgame, YouTube, and Attack, with each environment containing 100 provenance graphs.

Unicorn Wget Dataset[15]: This dataset includes simulated attacks designed by UNICORN, consisting of 150 batches of log data collected by CamFlow, of which 125 batches are benign and 25 are malicious.

DARPA-E3 Dataset[4]: This dataset is part of the DARPA TC program's third evaluation, containing several sub-datasets. We selected the Cadets, Theia, and Trace datasets to evaluate VODKA, using the same ground truth labels as ThreaTrace. Detailed descriptions of these datasets can be found in the appendix.

4.1.2 Metrics. We use the evaluation metrics from Unicorn[15] and ThreaTrace[45] to compare performance across different granularities, enabling reasonable comparisons with other detection methods. The performance metrics include Precision (PR), Recall (RC), F1-score and Accuracy (ACC).

4.1.3 Baselines. To perform a comprehensive performance evaluation, we compared VODKA with the best detection methods. Unfortunately, some methods were excluded due to challenges in reproducibility or incompatibility. For example, KAIROS[10] could not be compared as it uses a time window-based approach that differs from other methods. We selected the following detection methods as our baselines:

- **StreamSpot[22]**: StreamSpot detects intrusions by analyzing information flow graphs. It extracts features from the graph to learn a benign model and uses clustering methods to detect anomalous graphs.
- **Unicorn[15]**: Unicorn utilizes graph rendering techniques to efficiently summarize long-running system executions. It classifies graphs as benign or malicious and filters anomalies.
- **Prov-Gem[24]**: Prov-Gem captures entity interactions using a unified relational-aware embedding framework, with the help of supervised signals.
- **ThreaTrace[45]**: ThreaTrace customizes a new GraphSAGE method to aggregate system entity nodes, enabling node-level detection.
- **Log2vec[29]**: Log2vec is a heterogeneous graph embedding-based method for detecting network threats within enterprises. It identifies abnormal logs using node embeddings and clustering methods.
- **DeepLog[11]**: DeepLog leverages Long Short-Term Memory (LSTM)[49] networks to model system logs as natural language sequences and performs log-level anomaly detection.
- **MAGIC[23]**: MAGIC is a multi-granularity detection method that achieves efficient detection and cost reduction through graph masking techniques.

4.1.4 Implementation. Details We implemented VODKA 's prototype in approximately 2800 lines of Python3.11 code. For log processing, we used a log parser to convert audit log files from various sources into JSON format and then preprocess them. The development environment was PyTorch[36], and the model implementation was supported by the Deep Graph Library (DGL)[44]. Detailed teacher and student model parameters are provided in the A Appendix.

4.2 Overall Detection Efficacy Comparsion(Q1)

Table 2 and Table 3 show VODKA 's performance across various datasets using different teacher models. We observe that VODKA 's student model consistently improves ACC after distillation from all teacher models. Table 4 compares VODKA with the baselines across multiple datasets. For StreamSpot, we compare VODKA with StreamSpot, ThreaTrace, and MAGIC; for Unicorn Wget, we compare VODKA with Unicorn, Prov-Gem, ThreaTrace, and MAGIC; for DARPA E3 datasets, we compare VODKA with Prov-Gem, ThreaTrace, Log2vec, DeepLog, and MAGIC. Results indicate that VODKA achieves the best or second-best results on all datasets. Specifically, on the **StreamSpot**, which involves relatively simple attack scenario graphs, VODKA —being an entity-level detector—handles it quite easily. In the **Unicorn Wget**, which is somewhat more complex compared to StreamSpot but still graph-level, VODKA only slightly lags behind Prov-Gem in Precision (PR), but outperforms all other baselines across all other metrics. In the three **DARPA E3**, VODKA ranks second only to MAGIC in the Theia and Trace datasets, slightly behind in PR and F1-score.

Table 3: Detection results with teacher models as SAGE and SGC.

Datasets	Teacher (SAGE)	Student				+ACC Impv.	Teacher (SGC)	Student				+ACC Impv.
	ACC	ACC	PR	RC	F1		ACC	ACC	PR	RC	F1	
StreamSpot	98.10%	99.00%	99.13%	99.27%	99.20%	0.90%	98.24%	99.77%	99.47%	99.26%	99.36%	1.53%
Unicorn Wget	97.34%	99.34%	99.23%	99.15%	99.19%	2.00%	98.00%	99.36%	99.66%	99.50%	99.58%	1.36%
DARPA CADETS	97.20%	99.03%	94.87%	99.78%	97.26%	1.83%	97.03%	99.15%	94.13%	99.71%	96.84%	2.12%
DARPA TRACE	96.03%	97.45%	95.22%	99.34%	97.23%	1.42%	96.18%	97.60%	93.28%	99.35%	96.24%	1.42%
DARPA THEIA	96.93%	98.94%	95.94%	99.40%	97.65%	2.01%	97.29%	99.23%	94.70%	99.22%	96.94%	1.94%

Table 4: Vodka ' comparison results.

Datasets	Systems	ACC	PR	RC	F1
StreamSpot	StreamSpot	66%	73%	91%	81%
	ThreaTrace	96%	95%	93%	96%
	MAGIC	99%	99%	99%	99%
	Vodka	**99%**	**99%**	**100%**	**99%**
Unicorn Wget	Unicorn	90%	86%	95%	90%
	Prov-Gem	NA	**100%**	80%	89%
	ThreaTrace	98%	93%	98%	95%
	MAGIC	99%	98%	96%	97%
	Vodka	**99%**	99%	**99%**	**99%**
DARPA CADETS	Log2vec	98%	49%	85%	62%
	DeepLog	95%	23%	74%	35%
	ThreaTrace	99%	90%	99%	95%
	MAGIC	99%	94%	99%	97%
	Vodka	**99%**	**95%**	**99%**	**97%**
DARPA TRACE	Log2vec	97%	54%	78%	64%
	DeepLog	96%	41%	68%	51%
	ThreaTrace	98%	72%	99%	83%
	MAGIC	**99%**	**99%**	99%	**99%**
	Vodka	98%	95%	**99%**	97%
DARPA THEIA	Log2vec	99%	62%	66%	64%
	DeepLog	98%	16%	14%	15%
	ThreaTrace	99%	87%	99%	93%
	MAGIC	99%	**98%**	99%	98%
	Vodka	**100%**	96%	**99%**	**98%**

- 🟥 Best performance
- 🟨 Second best performance

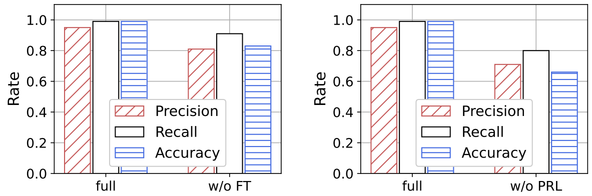
These results demonstrate that Vodka 's knowledge distillation approach effectively transfers knowledge from complex GNN teacher models to the student model, enabling the student model to retain high anomaly detection accuracy across multiple detection mechanisms.

4.3 Ablation Study (RQ2)

We conducted an ablation study on the CADETS dataset using GCN as the teacher model to investigate the individual contributions of Vodka 's different submodules, as shown in Figure 4.

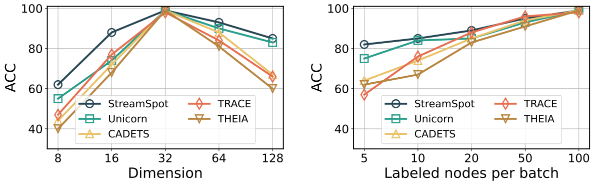
Impact of Feature Transformation: We removed the feature transformation mechanism from the student model, leaving only the label propagation mechanism. As shown in Figure 4.(a), removing the feature transformation significantly reduced performance—PR decreased by 14%, RC by 8%, and ACC by 16%,. This demonstrates that the feature-based prior knowledge is crucial for deep learning in node inference tasks. Without it, the student model loses the ability to learn deeply from system entity attributes.

Impact of Label Propagation: We removed the label propagation mechanism (PRL) from the student model, leaving only the feature transformation mechanism. As shown in Figure 4.(b), the absence of PRL led to an even larger performance drop—PR decreased by 24%, RC by 19%, and ACC by 33%. This highlights that PRL's contribution of structural and label-based prior knowledge is significant, enabling the student model to leverage contextual and temporal information effectively.



(a) FT (b) PRL

Figure 4: Ablation study



(a) Embedding Dimension (b) Training Ratio

Figure 5: Hyperparameter Investigation

4.4 Hyperparameter Investigation (RQ3)

We conducted the following experiments to explore how key hyperparameters affect ACC. Using GCN as the teacher model, we tested the entire dataset.

Embedding Dimension: The embedding dimension determines the length of the vector representing each node. We tested dimensions 8, 16, 32, 64, 128. Figure 5.(a) shows that lower dimensions result in poorer performance due to limited expressiveness. The optimal performance was achieved at 32 dimensions. Higher dimensions did not improve performance and even slightly decreased it due to redundant features, increasing overfitting risk.

Training Ratio: The training ratio determines the number of labeled nodes in each batch. To further demonstrate the effectiveness of Vodka, we tested values of 5, 10, 20, 50, 100. The experimental results are shown in Figure 5.(b). As we can observe, the classification performance of the student model improves as the number of

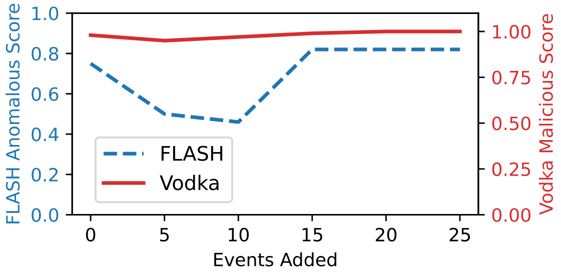


Figure 6: Robustness against adersarial attacks study.

labeled nodes increases, reaching near-optimal performance when the batch contains 100 labeled nodes. We attempt to analyze the reasons behind this phenomenon: Increasing the number of labeled nodes introduces more supervised signals, which in turn enhances the efficiency of label inference during PRL. For a semi-supervised system like Vodka, this improvement in label propagation is undoubtedly significant in boosting the model's performance.

4.5 Performance Overhead (RQ4)

Vodka aims to perform APTs detection with minimal overhead and flexible deployment in various environments. In real-world deployments, GPUs may be unavailable, and only CPUs may be used. Therefore, we primarily focused on Vodka 's detection time and memory usage under CPU-only conditions. Table 5 compares Vodka 's performance overhead with ShadeWatcher and MAGIC on the DARPA E3 TRACE dataset. Results show that Vodka has significant advantages in both detection time and memory usage. At the same training ratio, Vodka is 1.4 times faster than ShadeWatcher and 5.2 times faster than MAGIC.

Given its high detection speed, Vodka could be deployed as a real-time online detector. For example, with the TRACE dataset, which took two weeks to collect, generating 1.37GB of audit logs per day, Vodka only requires 23 seconds per day to complete the APTs detection. This means Vodka 's lightweight and high-speed detection capabilities make it suitable for daily APTs monitoring in small organizations and enterprises.

4.6 Robustness Against Adversarial Attacks (RQ5)

As more works focus on provenance-based data analysis, adversarial attacks[9, 41] designed to evade such provenance-based detectors are also increasing. One common adversarial attack is adversarial mimicry, which alters provenance data and adds benign representations to mimic benign behavior, disguising malicious system entities to evade detection. In this section, we simulate mimicry attacks to evaluate Vodka 's robustness against adversarial mimicry, using evaluation standards from [13] and [37]. Our mimicry attack steps are as follows: first, we select benign system entities from the normal training data. Next, we introduce events and combine them with the benign entities. Finally, we embed the structure of the benign nodes into the attack graph. We compare Vodka (using GCN as the teacher model and CADETS as the dataset) with FLASH[37]. Results (Figure 6) show that, as false events increase, FLASH's anomaly score significantly decreases, indicating its weak robustness against mimicry attacks. In contrast, Vodka maintains an anomaly score above 95% for all levels off false events, demonstrating its superior robustness.

Table 5: Vodka ' performance overhead.

System	Granularity	Time consumption(s)	Peak Memory consumption (MB)
ShadeWatcher	Edge	220	NA
MAGIC	Node	825	1,667
Vodka	Node	**158**	**982**

The reasons for this are twofold: first, Vodka is based on entity-level detection, making it highly sensitive to large-scale false event additions. Second, the provenance graph's neighbor denoising process smooths out some of the strong malicious expressions, making it less susceptible to manipulation.

5 RELATED WORK

APTs Detection. Current research primarily focuses on provenance-based methods to detect APTs, which can be categorized into three types based on the way audit logs are utilized: statistical-based methods, rule-based methods, and learning-based methods. **Statistical-based methods**, such as Nodoze[17], Swift[18], and PRIOTRACKER[30], quantify the rarity of statistical elements within the provenance graph, assigning suspiciousness scores to these elements, ultimately leading to anomaly detection. **Rule-based methods**, like Holmes[32], SLEUTH[20], and [16], collect prior attacks to help security personnel build attack rule libraries, enabling the detection of unknown attacks by matching them to predefined rules. **Learning-based methods** utilize machine learning techniques to model benign system behavior or attack patterns in order to identify unknown anomalous behaviors. Relevant work in this category includes [10, 15, 23, 37, 45, 50].

Knowledge Distillation. Knowledge distillation was first proposed by [19] and aims to transfer knowledge from a complex model (referred to as the teacher model) to a smaller model (referred to as the student model). This process enables the student model to reduce time and space complexity without compromising prediction quality, thus simplifying model inference and enhancing performance. While knowledge distillation has been widely applied in the field of computer vision, recent studies have extended the basic framework to other domains. Some works have also focused on the design of the student model. For example, [48] was the first to design a student model that is non-GCN-based, while [40] introduced contrastive learning into the distillation process to improve the student model's ability to learn representations.

6 CONCLUSION

In this paper, we innovatively propose a novel APTs detection method called Vodka, which integrates knowledge distillation and provenance-based APTs detection. Vodka introduces a graph Laplacian-based approach for neighbor denoising and signal smoothing on provenance graphs. Additionally, Vodka designs a knowledge distillation framework that distills GNNs knowledge from a complex teacher model into a lightweight student model. The student model combines feature transformation and label propagation, allowing for low-cost detection after training and reconstruction of the attack chain. We evaluate Vodka's performance on three widely used datasets, demonstrating excellent detection results.

REFERENCES

[1] 2016. The streamspot dataset. https://github.com/sbustreamspot/sbustreamspot-data.

[2] 2017. Equifax Information Leakage. https://en.wikipedia.org/wiki/Equifax.

[3] 2020. APT42 — Crooked Charms, Cons, and Compromises. https://www.mandiant.com/resources/podcasts/threat-trends-apt42-charms-cons-compromises.

[4] 2020. Darpa transparent computing program engagement 3 data release. https://github.com/darpa-i2o/Transparent-Computing.

[5] 2020. SolaWinds hack. https://en.wikipedia.org/wiki/SolarWinds.

[6] 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.

[7] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. 2010. Fast incremental and personalized pagerank. *arXiv preprint arXiv:1006.2880* (2010).

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

[9] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2021. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology* 6, 1 (2021), 25–45.

[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] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 1285–1298.

[12] Daniel Edler, Thaís Guedes, Alexander Zizka, Martin Rosvall, and Alexandre Antonelli. 2017. Infomap bioregions: interactive mapping of biogeographical regions from species distributions. *Systematic biology* 66, 2 (2017), 197–204.

[13] 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*.

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

[15] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. 2020. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525* (2020).

[16] 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.

[17] 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*.

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

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *stat* 1050 (2015), 9.

[20] 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.

[21] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. 2019. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5070–5079.

[22] Bart Jacob, Paul Larson, B Leitao, and SAMM Da Silva. 2008. SystemTap: instrumenting the Linux kernel for analyzing performance and functional problems. *IBM Redbook* 116 (2008).

[23] 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.

[24] 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.

[25] Eman J Khaleefa and Dhahair A Abdulah. 2022. Concept and difficulties of advanced persistent threats (APT): Survey. *International Journal of Nonlinear Analysis and Applications* 13, 1 (2022), 4037–4052.

[26] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[27] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. 2019. Label efficient semi-supervised learning via graph filtering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9582–9591.

[28] Zhenyuan Li, Qi Alfred Chen, Runqing Yang, Yan Chen, and Wei Ruan. 2021. Threat detection and investigation with system-level provenance graphs: A survey. *Computers & Security* 106 (2021), 102282.

[29] 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.

[30] 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*.

[31] Long Ma and Yanqing Zhang. 2015. Using Word2Vec to process big text data. In *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2895–2897.

[32] 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.

[33] Talha Ongun, Jack W Stokes, Jonathan Bar Or, Ke Tian, Farid Tajaddodianfar, Joshua Neil, Christian Seifert, Alina Oprea, and John C Platt. 2021. Living-off-the-land command detection using active learning. In *Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses*. 442–455.

[34] 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*.

[35] 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.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[37] 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.

[38] 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.

[39] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang. 2015. Extreme learning machine for multilayer perceptron. *IEEE transactions on neural networks and learning systems* 27, 4 (2015), 809–821.

[40] Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699* (2019).

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

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

[43] Hongwei Wang and Jure Leskovec. 2020. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755* (2020).

[44] Minjie Yu Wang. 2019. Deep graph library: Towards efficient and scalable deep learning on graphs. In *ICLR workshop on representation learning on graphs and manifolds*.

[45] 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.

[46] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[47] Feng Xia, Jiaying Liu, Hansong Nie, Yonghao Fu, Liangtian Wan, and Xiangjie Kong. 2019. Random walks: A review of algorithms and applications. *IEEE Transactions on Emerging Topics in Computational Intelligence* 4, 2 (2019), 95–107.

[48] Cheng Yang, Jiawei Liu, and Chuan Shi. 2021. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *Proceedings of the web conference 2021*. 1227–1237.

[49] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. 2019. A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation* 31, 7 (2019), 1235–1270.

[50] 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.

A APPENDIX

A.1 Provenance Graph Description

Table 6 provides a description of the system event types and the edge relationships in the provenance graph.

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

Table 6: System event and Rlation description.

A.2 Datasets

We provide a detailed description of the three datasets used(Table 7, Table 8, Table 9), including their specific contents and original sizes.

Scenarios	# of Graphs	Average # of Nodes	Average # of Edges
YouTube	100	8,292	113,229
Gmail	100	6,826	37,382
Video Game	100	8,636	112,958
Download	100	8,830	310,814
CNN	100	8,989	294,903
Attack	100	8,890	28,423

Table 7: Overview of StreamSpot Dataset.

Scenarios	# Log pieces	Average # of Entity	Avg # of Interaction	# of Raw DataSize(GB)
Benign	125	265,424	975,226	64.0
Attack	25	257,156	957,968	12.6

Table 8: Overview of Unicorn Wget Datasets.

Datasets	# of Node	#of Edges	# of Raw Data Size(GiB)
E3-THEIA	1,623,966	2,874,821	17.9
E3-CADETS	1,627,035	3,303,264	18.3
E3-TRACE	3,288,676	4,080,457	15.4

Table 9: Overview of Darpa-E3 Datasets.

A.3 Initial Parameters for Teacher and Student Models

We provide more information about the initial settings for both the teacher and student models.

Teacher Models are as follows:

- **GCN**: 2 layers, 64 hidden units, learning rate of 0.01, dropout probability of 0.8, learning rate decay of 0.01.
- **GAT**: 2 layers, 8 attention heads, 64 hidden units, learning rate of 0.01, dropout probability of 0.6, learning rate decay of 0.01, attention dropout probability of 0.3.
- **SAGE**: 128 hidden units, learning rate of 0.01, sample size of 5, batch size of 256, learning rate decay of 0.005.
- **SGC**: 2 layers, learning rate of 0.01, learning rate decay of 0.01.

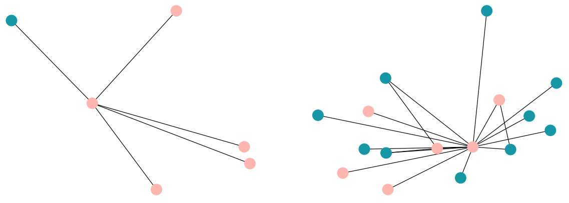
Student Model: The number of layers K is set to 5, the MLP hidden layer has 16 units, dropout rate is 0.2, the Adam optimizer has a learning rate of 0.01, and weight decay is set to 0.001.

A.4 Interpretability Analysis of the Student Model

We conduct an interpretability analysis of the student model. Recall that Vodka's student model consists of two main mechanisms: the Feature Transformation (FT) mechanism, powered by MLP, and the Personalized PageRank Random Walk Label Propagation (PRL) mechanism. The label propagation mechanism includes a "jump-back" feature, meaning there is a probability of returning to the previous node, which is controlled by the parameter α. Additionally, the final prediction result of the student model is $f_{std}^{k+1}(v) = \beta_v f_{FT}(v) + (1 - \beta_v) f_{\text{PRL}}^{k+1}(v)$, and this is controlled by the balancing parameter β. Our goal is to explore how these two parameters affect the bias of the student model's prediction results. The experiments were conducted with a GCN as the teacher model and using the DARPA CADETS dataset.

Callback Probability α: We dynamically adjust α and plot the neighborhood of the target node when α is at its maximum(Node 1) and minimum(Node 2) values, using different colors to represent labels. As shown in Figure 7, the neighbors of Node 1 are more diverse and numerous compared to those of Node 2. This indicates that nodes with a high α have a higher probability of jumping back to themselves, limiting the diversity of learning from neighboring nodes.

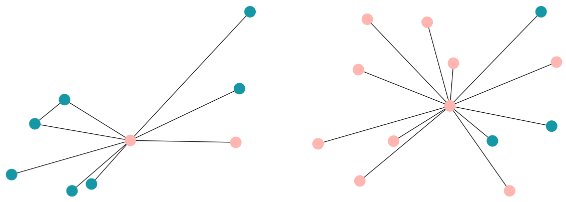
Balancing Parameter β: Similarly, we dynamically adjust β and plot the neighborhood of the target node when β is at its maximum(Node 3) and minimum(Node 4) values, with different colors representing labels. As shown in Figure 8, the labels of Node 4's neighborhood are almost identical, whereas those of Node 3 are almost entirely different. This reflects that the Personalized PageRank Random Walk (PRL) mechanism contributes more to node prediction compared to FT, as when the balancing parameter favors PRL, the prior knowledge from labels and structure plays a more significant role in improving prediction accuracy.



(a) Node 1

(b) Node 2

Figure 7: Study of α



(a) Node 3

(b) Node 4

Figure 8: Study of β