Hindawi
Security and Communication Networks
Volume 2021, Article ID 9961342, 14 pages
https://doi.org/10.1155/2021/9961342

Research Article

A Hierarchical Approach for Advanced Persistent Threat Detection with Attention-Based Graph Neural Networks

Zitong Li,[1] Xiang Cheng,[1] Lixiao Sun,[1] Ji Zhang,[2] and Bing Chen[1]

[1]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 21106, China*
[2]*School of Sciences, University of Southern Queensland, Toowoomba 4350, Australia*

Correspondence should be addressed to Bing Chen; cb_china@nuaa.edu.cn

Received 11 March 2021; Revised 15 April 2021; Accepted 26 April 2021; Published 4 May 2021

Academic Editor: Weizhi Meng

Advanced Persistent Threats (APTs) are the most sophisticated attacks for modern information systems. Currently, more and more researchers begin to focus on graph-based anomaly detection methods that leverage graph data to model normal behaviors and detect outliers for defending against APTs. However, previous studies of provenance graphs mainly concentrate on system calls, leading to difficulties in modeling network behaviors. Coarse-grained correlation graphs depend on handcrafted graph construction rules and, thus, cannot adequately explore log node attributes. Besides, the traditional Graph Neural Networks (GNNs) fail to consider meaningful edge features and are difficult to perform heterogeneous graphs embedding. To overcome the limitations of the existing approaches, we present a hierarchical approach for APT detection with novel attention-based GNNs. We propose a metapath aggregated GNN for provenance graph embedding and an edge enhanced GNN for host interactive graph embedding; thus, APT behaviors can be captured at both the system and network levels. A novel enhancement mechanism is also introduced to dynamically update the detection model in the hierarchical detection framework. Evaluations show that the proposed method outperforms the state-of-the-art baselines in APT detection.

1. Introduction

Advanced Persistent Threats (APTs) are becoming increasingly prominent in modern networks [1, 2]. Unlike conventional attacks, APTs are a class of sophisticated attacks launched by resourceful adversaries using a wide spectrum of attack techniques and tools [3]. The APT perpetrators initially compromise hosts or servers in a target environment and then stealthily traverse from system to system for internal reconnaissance and data breach [4, 5].

Traditional detection systems are insufficient to defend against APT attacks. Misuse-based detectors [6, 7] that learn patterns associated with known attacks are difficult to detect APTs because of their nonrepetitive behaviors. In contrast, anomaly-based detectors [8–13] are capable of identifying unforeseen activities that do not conform to the learned normal patterns. However, they are susceptible to be circumvented by attackers because they typically treat system calls or network events as temporal sequences [8, 9, 11], which only carry the sequential relationships among log entries. As such, they cannot achieve satisfactory performance in detecting APTs [1].

Recent works suggested that the provenance graph is a better tool for threat modeling and APT detection [14]. Provenance graphs represent system executions as control flows and data flows between subjects and objects. The naïve method is to perform rule-based subgraph matching for APT detection [1, 15, 16], but it is incapable of dealing with unknown APT patterns. Moreover, provenance graphs only focus on system-level information and thus cannot effectively model network-level behaviors. Besides, the correlation graphs [3, 17] treat log entries as nodes and bridge them by rules. The limitation of log graphs is that they require too much expert knowledge to define proper correlation rules, and coarse-grained log nodes ignore semantic information of log attributes, making it difficult for them to capture system-level APT patterns.

Traditional graph embedding methods, including DeepWalk [18] and node2vec [19], rely on heuristic algorithms to aggregate graph structural information. These methods are inherently transductive and fail to encode node attributes for graph embedding. Graph Neural networks (GNNs) [20–22] are deep learning methods to perform graph representation learning with good scalability and generalization ability. A series of GNN-based anomaly detection systems [12, 23, 24] have been proposed to perform inductive graph embedding by learning a set of aggregator functions to aggregate neighbor's features. Unfortunately, these models are not well-suited for APT detection. Because it is difficult for them to encode heterogeneous graphs integrating multiple types of system entities and operations, and most of them discard meaningful edge attributes that represent interactions between nodes.

To overcome the limitations of existing approaches, we propose a hierarchical approach that is capable of effectively detecting APTs with novel attention-based GNNs. Our approach comprises three components as shown in Figure 1. (1) Graph construction: we construct the Intrahost Provenance Graph (IPG) and the Interhost Interactive Graph (IIG) to comprehensively capture the behaviors of the full APT lifecycle. (2) Graph embedding and detection algorithm: for the IPG, we propose a metapath aggregated GNN and train an autoencoder to identify anomalous hosts; for the IIG, we present an edge feature-enhanced GNN and leverage the negative sampling to detect anomalous interactions among hosts. (3) Anomaly detection and model update: after the anomalous hosts and events are detected, the enhancement mechanism dynamically updates the IIG detection model using the reported malicious hosts from the IPG detectors.

We introduce specialized designs to tackle the aforementioned problems. First, the IPG and the IIG enable modeling behaviors both at system and network levels. Second, we encode the heterogeneous IPG using the metapath aggregated GNN with the attention mechanism, which enables the full exploration of semantic information using metapaths tailored to the IPG. Third, for the IIG, the interaction edges among hosts contain meaningful information; thus, we propose the edge feature-enhanced GNN to adequately exploit the multidimensional edge features. Additionally, with the enhancement from the IPG detectors, the IIG detection model can be efficiently updated to learn new patterns, and the malicious behaviors among hosts can be further detected after the compromised hosts are reported. Finally, we introduce a compact three-stage APT model and map the anomalies at both the system and network levels to the corresponding APT stages according to the hierarchical detection framework.

Specifically, the contributions of this article are summarized as follows:

(i) We propose a novel metapath aggregated GNN that models complex semantic and structural information of the system-level provenance graph.

(ii) We present a novel edge feature-enhanced GNN that models rich interactive information of the network-level host interactive graph.

(iii) We introduce an enhancement mechanism to dynamically update the network-level IIG model using the reported malicious host from the system-level IPG detectors to learn new anomalous patterns over time.

(iv) We propose a compact three-stage APT model and a hierarchical detection framework where the system-level and network-level anomalies can be mapped to the corresponding APT stages.

(v) The proposed method is evaluated on the StreamSpot and the LANL datasets for system-level and network-level APT attack detection. Experimental results show that our method outperforms the state-of-the-art approaches.

The remainder of the article is organized as follows. Section 2 reviews the related works. In Section 3, we present the schematic architecture, the compact three-stage APT model, and the hierarchical detection framework. The graph construction is shown in Section 4. Section 5 presents the details of the anomaly detection in provenance graphs. Section 6 presents the network-level anomaly detector and the model update mechanism. Evaluation results are presented and discussed in Section 7. The limitations and future works are discussed in Section 8. The conclusion is drawn in Section 9.

2. Related Work

Our work lies in the intersection of log analysis, provenance-based threat modeling, and graph-based intrusion detection. Therefore, we discuss the existing works in the following related areas. The taxonomy and representative publications are shown in Table 1.

2.1. Sequence-Based Log Analysis. Modern systems constantly generate logs and events that describe system status at various critical points, which are ideal sources of information for attack detection and system failures debugging [32]. LogLens [11] is a real-time anomaly detection system that deployed an unsupervised learning method to analyze log sequences without the knowledge of target systems and user specifications. Advances in natural language processing have shed light on log analysis. DeepLog [8] converts system logs into natural language sequences by utilizing the Long Short-Term Memory (LSTM) network, which can automatically learn normal patterns from system behaviors and alert for anomalies. To identify rare anomalies in the constantly evolving environment, Parveen [9] presented an unsupervised ensemble learning algorithm that compresses repetitive sequences to a dictionary to detect anomalies. In addition to detecting malicious activities as they happen, attack prediction is still an open research problem. Tiresias [25] leverages the Recurrent Neural Network (RNN) to predict specific attack steps by considering previously observed events.

These log analysis approaches mostly treat logs as temporal sequences, which only hold sequential

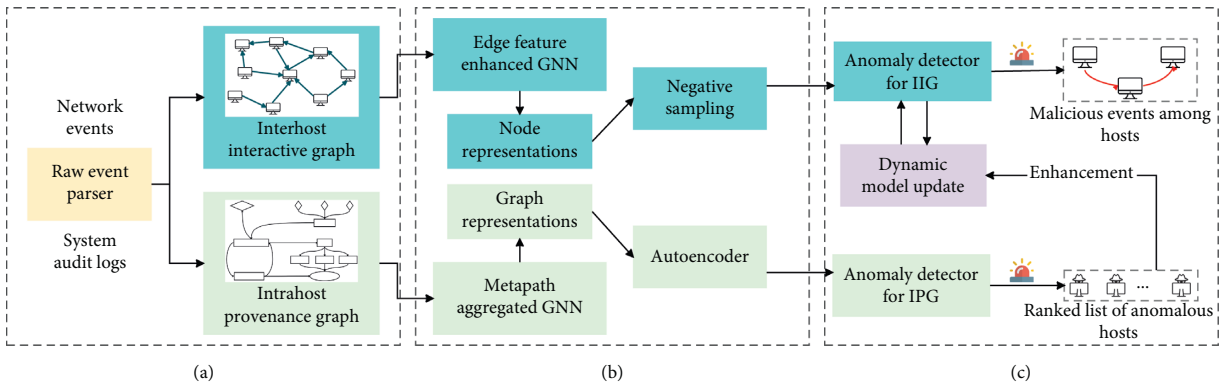

Figure 1: Schematic architecture: (a) graph construction; (b) graph embedding and detection algorithm; (c) anomaly detection and model update.

relationships among log entries. Therefore, they ignore the semantic relationships among system entities and the interactive relationships among hosts, whereas our method considers these relationships for anomaly detection.

2.2. Provenance-Based Threat Modeling. More and more research works start to focus on the provenance graphs that model the control flow and data flow between system-level entities [14]. Provenance is typically used for forensic analysis and attack attribution. BackTracker [28] leverages the provenance tracking system to identify the entry point of attacks, while PriorTracker [29] enhances it by timely forward causality tracking and automatically prioritizing the abnormal causal dependencies. To combat the challenge named "threat alert fatigue," NoDoze [30] uses contextual and historical information of alerts in the provenance dependency graph and performs attack triage by identifying anomalous subgraphs. SPADE [26] is an open-source provenance collection and management framework, which decouples the function of collection, storage, and querying. Pasquier et al. [27] designed a practical provenance capture system, called CamFlow, that can tailor the captured data to reduce the overhead.

The increasingly sophisticated APTs prompt a number of researchers to use provenance graphs for APT analysis. SLEUTH [15] constructs memory-based provenance graphs that significantly improve the speed of data processing and uses the trustworthiness and confidentiality tags for code and data to perform source identification and impact analysis. HOLMES [1] uses the semantic information of provenance to construct a customized policy framework, and the behaviors that conform to the policies are further mapped to Tactics, Techniques, and Procedures (TTPs), which are eventually mapped to high-level APT kill-chain stages. Poirot [16] uses provenance graphs to perform threat detection. It extracts Indicators of Compromise (IoCs) from Cyber Threat Intelligence (CTI) related to APTs to construct the query graphs and aligns them with provenance graphs constructed out of kernel audit logs to detect attack behaviors. However, these rule-based methods require prior expert knowledge of known APT patterns and thus cannot deal with unknown APT attacks. Our method instead is an anomaly-based system that requires no expert knowledge or labeled anomalous data.

2.3. Graph-Based Anomaly Detection. Anomaly detection with graph data has been widely researched and applied to identify outliers [12]. To analyze temporal evolution of insider threat events, Moriano et al. [31] proposed an unsupervised learning method to capture interactions between users and systems by constructing a bipartite graph. Log2vec [3] optimizes the correlated log graphs by constructing a rule-based heterogeneous graph integrating multirelationships among log entries. Then, Log2vec presents an improved graph embedding algorithm based on the random walk and word2vec and leverages the clustering threshold detector to identify malicious events. StreamSpot [13] is a memory-efficient anomaly detection system that deals with provenance graph heterogeneity and streaming challenge. To mitigate the drawbacks of StreamSpot in handling locally constrained graph features and dynamically clusters maintaining, UNICORN [2] analyzes contextualized provenance graphs to detect APTs. It is capable of modeling and summarizing the evolving system executions and report anomalous system status.

However, these approaches focus on either the system operations or the network flows for graph analysis, making them unable to fully capture the activities of APT attacks. Also, they depend on handcrafted algorithms that are difficult to generalize to different settings. GNNs have been widely adopted to learn node embeddings in an unsupervised way for anomaly detection. Ding et al. [23] studied the graph anomaly detection problem and deployed a novel model with the synergy of Graph Convolutional Network (GCN) and autoencoder. AddGraph [24] further combines GCN with Gated Recurrent Unit (GRU) to capture the long-term and short-term temporal patterns using an attention model. However, these GNN-based anomaly detection methods are not well-suited to APT detection because they either discard meaningful edge features or difficult to embed heterogeneous graphs. We introduce the metapath aggregated GNN to fully explore the semantic information embedded in the heterogeneous IPG, and we proposed the edge

Table 1: Taxonomy and representative publications of the related works.

Category		Publications
Sequence-based log analysis	Attack detection	[8, 9, 11]
	Attack prediction	[25]
Provenance-based threat modeling	Provenance capture	[26, 27]
	Forensic analysis	[28–30]
	Rule-based APT detection	[1, 15, 16]
Graph-based anomaly detection	Provenance graph	[2, 13]
	Correlation graph	[3, 31]
	GNN-based methods	[23, 24]

feature-enhanced GNN to adequately exploit the edge attributes in the IIG.

3. Overview

3.1. Architecture. As shown in Figure 1, our approach is composed of three key components:

3.1.1. Graph Construction. To capture all footprints left by APT attackers, we collect multimodal information of the system audit logs from different operating systems and network events from various protocols. The input to the graph construction component is the raw network events and system audit logs, which are normalized to uniform formats by the data parser. System audit records from different OS platforms are normalized to a common data representation of system entities and operations. Network events involving different protocols are normalized to a common data representation of interactions between hosts.

To capture the structural information and semantic dependencies, the parsed system audit logs are used to construct the IPG where all system entities are treated as nodes and operations are treated as edges, and the parsed network events are used to construct the IIG where all hosts are treated as nodes and events among them are treated as edges. By doing so, the execution status of the target environment is comprehensively captured from both system-level and network-level graphs. The IPG is capable of modeling system-level APT behaviors, such as exploitation of target systems and malicious code execution. The network-level APT events, which can be captured by the IIG, are not limited to lateral movement and DNS communication with Command and Control (C & C) servers.

3.1.2. Graph Embedding and Detection Algorithm. Graph embedding, also known as graph representation learning, is an approach that is capable of transforming graph nodes into vectors with a lower dimension while maximally preserving structural and semantic information. Specifically, we proposed a metapath aggregated GNN for the IPG and an edge feature-enhanced GNN for the IIG. For the system-level IPG with multitypes of entities and operations, we aggregate every single metapath instance, including the neighbor nodes and intermediate nodes, to a latent representation vector; then, we combine them by an attention-based mechanism to obtain the embedding of target nodes. For the network-level IIG with rich edge features, we leverage the novel edge feature-enhanced embedding algorithm based on the attention mechanism to adequately incorporate the interactions among host nodes.

After converting nodes into latent vectors, we design two detection algorithms for anomaly detection at both the system and network levels. Note that we have no labeled anomaly data to training the detection model; thus, the detection models are trained in an unsupervised manner. For the IPG, we use graph embeddings to train an autoencoder model by optimizing the reconstruction errors that are used to flag anomalies. Moreover, for the IIG, we compute the anomaly scores for edges based on the corresponding node embeddings and leverage the negative sampling to generate anomalous edges for training the detection model.

3.1.3. Anomaly Detection and Model Update. At last, the learned models are implemented to identify anomalous network events and system operations. The IPG detectors report a ranked list of suspicious hosts that perform system-level malicious actions inside, while the IIG detector reports a set of malicious events among hosts. However, the training data recording normal executions are usually incomplete and noisy, which may lead to an excessive number of false alarms during anomaly detection. To enhance the detection model, we use the reported suspicious hosts from the IPG detectors to dynamically update the IIG detection model. Specifically, after the IPG detectors report a list of suspicious hosts, malicious interactions appear among these hosts with high probability. Thus, we treat the interactions among these hosts as malicious samples to dynamically update the IIG detection model to learn new anomalous patterns over time.

3.2. Three-Stage APT Model. In contrast to other conventional attacks, APTs are characterized by persistence, diverse attack vectors, and low-and-slow attack patterns, which give rise to multiple attack stages of the APT lifecycle. MITRE ATT & CK framework presents a 14-stage APT knowledge base to describe adversary tactics and techniques of APT attacks. The Lockheed Martin cyber kill-chain [33] is a 7-phase framework to describe the sophisticated APT attack process. Nevertheless, some stages, such as the Initial Reconnaissance, are difficult to be detected given that the actions of the phase are conducted without any interaction with the target network. As such, it is impracticable to design

a comprehensive approach to detect all the stages of the APT lifecycle. Moreover, an overly redundant multistage model can only be used to better for understanding the evolution of APTs, but not for detecting them.

After analyzing various APT cases, a compact three-stage APT model that is at the heart of the APT lifecycle is proposed. The invariant parts of APTs include three stages:

(i) Infiltration and malicious code execution: at first, the attacker must deploy the malware and execute the malicious code in the victim host to exploit an entry point of the target network or establish a foothold for the next move.

(ii) Internal reconnaissance and lateral movement: the goal of the APT perpetrator is to steal confidential data or to damage critical network components. To this end, the attacker would need to further compromise more vulnerable hosts, traverse from system to system within the target environment, and escalate privileges at any time if needed.

(iii) C & C communication and data exfiltration: the infected host would try to communicate with the C & C server to receive remote attack instructions or any other relevant tasks. Besides, actions that involve exfiltrating data to the C & C server and undermining critical components fall under this stage.

The compact three-stage APT lifecycle is presented to model APT attacks performed in complex networks and is applicable and necessary for most APT scenarios.

3.3. The Hierarchical APT Detection Framework. As shown in Figure 2, the hierarchical APT detection framework maps anomalies of system-level hosts and network-level events to high-level APT stages. APT attackers not only deploy malicious code inside specific hosts to manipulate processes and files but also move laterally for internal reconnaissance and information exfiltration. The IPG detectors over the Intrahost Provenance Graphs report malicious hosts that are suffering attacks. Further, these hosts detected at the system layer can be used as seeds to analyze which internal hosts they try to communicate with at the network layer. Moreover, the IIG detection model can also be dynamically updated by learning detection results from the IPG detectors. At last, the infected host nodes and attack paths among them are mapped to a compact three-stage APT model described above. Bringing together the information of system audit logs and network events into graph data allows us to comprehensively model the status of the target environment, and the synergy between the IPG and IIG detection model enables our method to capture footprints of APT attacks by considering the target environment as a whole.

4. Graph Construction

4.1. Intrahost Provenance Graph. The system-level provenance graph enables a strong semantic expression of system execution and is increasingly attractive for researchers in the APT analysis. The provenance collection system captures the record information at the system level from different operating systems; thus, causality dependencies can be properly captured. As shown in Figure 3, the provenance graph treats all subjects (e.g., processes, threads) and objects (e.g., files, sockets) as nodes and all operations from subjects to objects (such as a process reading a file, interprocess communication, and a process sending data to an Internet socket) as directed edges. The built-in auditing systems, such as Windows ETW (Event Tracing for Windows) and Linux Auditd, provide coarse-grained provenance capture, while the extra provenance infrastructure, such as Hi-Fi [34], LPM [35], and CamFlow [27], are used to collect more fine-grained provenance.

Following the idea of the provenance graph, we collect system operations from all hosts of the target network and model these event data as a platform-neutral heterogeneous graph for each host. Formally, a heterogeneous IPG is denoted by IPG = (V, E), in which V is a set of nodes involving subjects and objects and E denotes a set of events. The types of subject and object in IPG include process, thread, file, and socket. It is noteworthy mentioning that the subject and object are relative, meaning that a subject of an event can be the object of another operation. The event in IPG can be represented by a 4-tuple <subject, object, operation, *timestamp*> where the operation types include the fork, clone, open, write, read, etc. The IPG is used to detect the infiltration and malicious code execution of the APT model.

4.2. Interhost Interaction Graph. To exfiltrate confidential data or sabotage critical components, the APT perpetrators would need to laterally move across multiple hosts of the target Intranet to escalate privilege and search for the components and data. Besides, the infected hosts would try to communicate with the C & C server via the DNS requests to receive the instructions and exfiltrate data. As such, the analysis of interactive relationships among multiple hosts is essential for the detection and mitigation of APT activities. To this end, the Interhost Interaction Graph is proposed to model the communication and connectivity of the entire target network.

As shown in Figure 4, in the IIG, each node represents a host or server while the interactions between them are denoted by edges. In an enterprise, hosts own different users who tend to conduct diverse operations with internal and external nodes. For instance, administrators often log into multiple hosts for policy management, parameter configuration, and crash recovery. Moreover, file transmission and domain name lookup are frequently performed, which could generate a volume of data flow and DNS traffic. To capture these interhost behaviors, the IIG takes network flow events, authentication events, and DNS lookup events into consideration. Formally, the Interhost Interaction Graph is denoted by IIG = (V, E), in which V is the set of host nodes and E is the set of events conducted by the source host to the destination host. The IIG is mainly used to detect the last two stages of the proposed APT model.

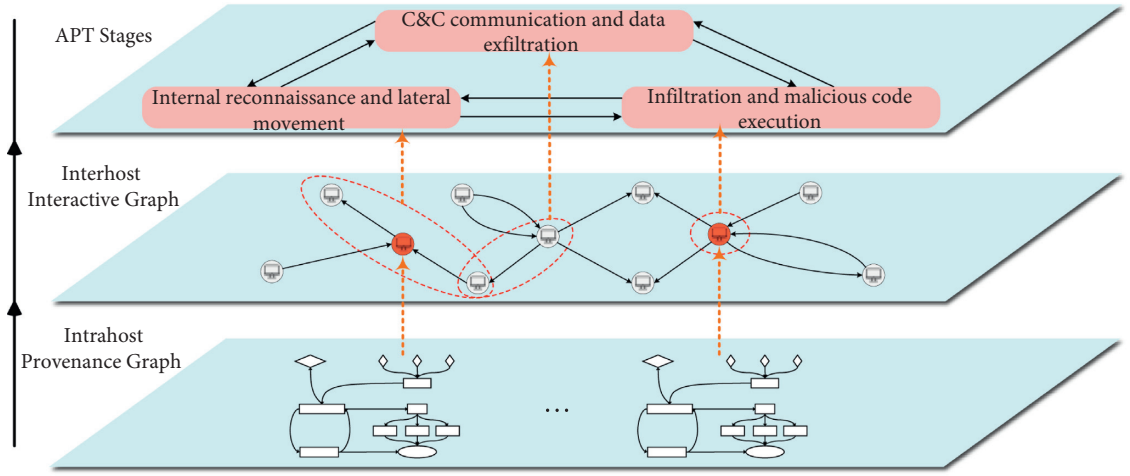

Figure 2: The hierarchical APT detection framework.

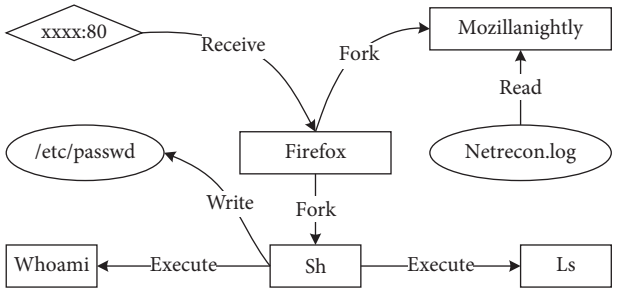

Figure 3: The intrahost provenance graph.

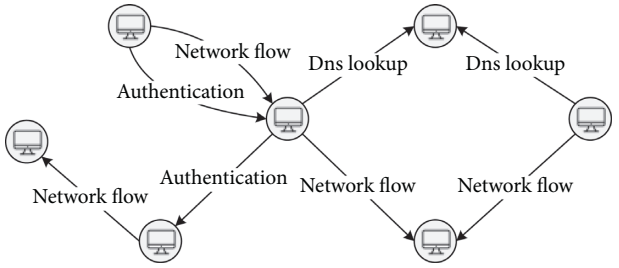

Figure 4: The interhost interaction graph.

5. Anomaly Dete ction on the IPG

5.1. Metapath Aggregated GNN. The constructed IPGs are heterogeneous with multiple different types of system entities and causal dependencies. Thus, we propose a novel metapath aggregated GNN to embed nodes of the heterogeneous IPG into low-dimensional node representations in a semantically meaningful way. The definitions of metapath and metapath instance are as follows:

Metapath: Ametapath P is an ordered sequence in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$ defined on the graph network schema $T_{\mathcal{G}} = (\mathcal{A}, \mathcal{R})$, which describes a composite relation $R = R_1 \cdot R_2 \cdot \cdots \cdot R_l$ between node types A_l and A_{l+1}, where l denotes the length of P and $\cdot$ is the composite operator on relations. For simplicity, we use an abbreviation form $P = (A_1, A_2, \ldots, A_{l+1})$ to denote a metapath when there is only one event type between a pair of entities.

Metapath instance: Given that metapath $P = (A_1, A_2, \ldots, A_{l+1})$, if $\forall i, \phi(v_i) = A_i$ and $e_i = <v_i, v_{i+1}>$ belongs to the relationship $R_i \in P$, then metapath instance p is presented in the form of $p = (v_1, v_2, \ldots, v_{l+1})$ following the schema defined by metapath P.

As shown in Figure 5, we propose a metapath encoder for the IPG embedding. Given metapath P, the metapath encoder combines the information embedded in the metapath context by aggregating the metapath-based neighbors of the target node v and the intermediate nodes for each metapath instance of P. Specifically, let $p(v,u)$ denote a metapath instance of P correlating the target node v and its metapath-based neighbor $u \in \mathcal{N}_v^P$, and $I_{p(v,u)} = p(v,u) \setminus \{v,u\}$ denotes the set of intermediate nodes within $p(v,u)$. The metapath encoder first employs a node aggregator to embed all node embeddings along with a metapath instance into a single vector:

$$\mathbf{h}_{p(v,u)} = f\left(\mathbf{h}'_v, \mathbf{h}'_u, \left\{\mathbf{h}'_t, \forall t \in I_{p(v,u)}\right\}\right), \tag{1}$$

where $\mathbf{h}'_v$, $\mathbf{h}'_u$, and $\mathbf{h}'_t$ are the input node representations, $f(\cdot)$ is a node aggregation function; $\mathbf{h}_{p(v,u)}$ is the aggregated output of the metapath instance $p(v,u)$.

After aggregating every single metapath instance into a latent representation vector, the metapath encoder combines different instances of metapath P regarding target node v through the graph attention layer. It is reasonable to use the weighted sum of metapath instances because different instances would have different degrees of contribution to the target node's embedding. Thus, all metapath instances can be weighted summed by learning a normalized attention coefficient:

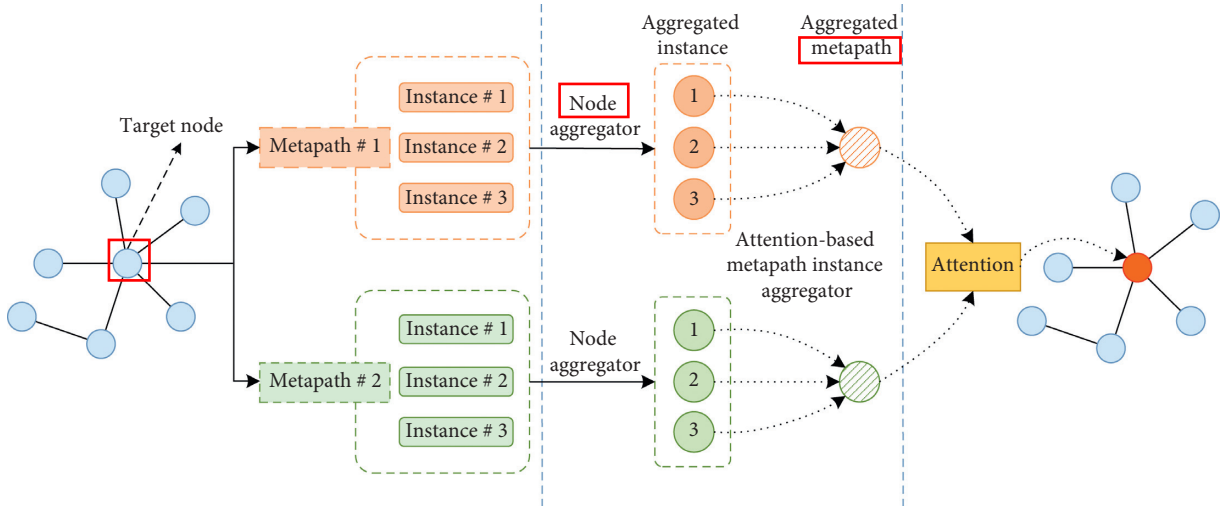

Figure 5: The metapath encoder for Intrahost Provenance Graphs.

$$e_{vu}^{p} = a\left(\mathbf{W}\mathbf{h}_{v}', \mathbf{W}\mathbf{h}_{p(v,u)}\right),$$

$$\alpha_{vu}^{p} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}_{P}^{\top}\left[\mathbf{W}\mathbf{h}_{v}' \| \mathbf{W}\mathbf{h}_{p(v,u)}\right]\right)\right)}{\sum_{s \in \mathcal{N}_{v}^{P}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}_{P}^{\top}\left[\mathbf{W}\mathbf{h}_{v}' \| \mathbf{W}\mathbf{h}_{p(v,s)}\right]\right)\right)}, \quad (2)$$

$$\mathbf{h}_{v}^{P} = \sigma\left(\sum_{u \in \mathcal{N}_{v}^{P}} \alpha_{vu}^{p} \cdot \mathbf{h}_{p(v,u)}\right),$$

Here, $\mathbf{a}_P$ is a weight attention vector for metapath P. We concatenate the representations of target node v and metapath instance embedding $p(v,u)$ parametrized by $\mathbf{a}_P^{\top}$ and apply the LeakyReLU nonlinear activation function to compute the attention coefficient e_{vu}^{p}, which indicates the importance of the representation of the metapath instance $p(v,u)$ to the node v. The coefficients are normalized to make them easily comparable across different instances using the softmax function, and then they are used as the weights to compute the combination of all representations of different metapath instances about node v. The output finally goes through an activation function $\sigma(\cdot)$ to obtain the representation of target node v regarding metapath P.

After aggregating the information contained in each metapath, we need to further aggregate the structural and semantic information among all different metapaths. For a given target node v and the set of metapaths $\mathcal{P} = \{P_1, P_2, \ldots, P_K\}$, we now have K representations for each metapath P_i denoted as $\{\mathbf{h}_{v}^{P_1}, \mathbf{h}_{v}^{P_2}, \ldots, \mathbf{h}_{v}^{P_K}\}$. The metapath encoder further employs an attention mechanism to combine the metapath-specific representations by assigning different weights to different metapaths. First, we average the transformed representations of metapath $P_i \in \mathcal{P}$ for all nodes as

$$\mathbf{e}_{P_i} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \tanh\left(\mathbf{W} \cdot \mathbf{h}_{v}^{P_i} + \mathbf{b}\right), \quad (3)$$

where $\mathbf{W}$ and $\mathbf{b}$ are the weight and bias parameters. Then, the attention-based mechanism is used to aggregate the metapath-specific representations of the target node v as follows:

$$\alpha_{P_i} = \frac{\exp\left(\mathbf{a}^{\top} \cdot \mathbf{e}_{P_i}\right)}{\sum_{P_j \in \mathcal{P}} \exp\left(\mathbf{a}^{\top} \cdot \mathbf{e}_{P_j}\right)},$$

$$\mathbf{h}_{v} = \sum_{P_i \in \mathcal{P}} \alpha_{P_i} \cdot \mathbf{h}_{v}^{P_i}, \quad (4)$$

where $\mathbf{a}$ is a weight attention vector and α_{P_i} indicates the relative importance of metapath P_i to the target node v. Once $\mathbf{e}_{P_i}$ is obtained for each metapath $P_i \in \mathcal{P}$, we use the normalized importance α_{P_i} to sum up all metapath-specific representations of node v in a weighted way to obtain the final node representation $\mathbf{h}_v$. Finally, to obtain the graph level representation $\mathbf{h}_g$, the mean readout operation is performed by averaging all the representations of the target nodes.

5.2. Autoencoder-Based Anomaly Detection. To detect anomalies launched by APT attackers, an anomaly detection mechanism based on the deep autoencoder is designed to reconstruct the learned graph embeddings. The deep autoencoder is used to learn data encodings in an unsupervised fashion, typically for data denoising and dimensionality reduction. Here, we leverage it to perform anomaly detection by computing the reconstruction errors that are used to flag anomalies. The intuition behind this is that the instances with large reconstruction errors are more likely to be anomalies because their behavior patterns significantly deviate from the normal patterns and thus cannot be effectively reconstructed from the observed data. The deep autoencoder is capable of identifying those hosts that do not conform to the expected normal patterns.

For system-level anomaly detection, the autoencoder takes the representations of IPG snapshots as its inputs. Thus, the anomaly detection in the provenance graph scenario can be simply regarded as a graph classification problem. Specifically, given graph embedding $\mathbf{h}_g$, the encoder is first implemented to compress the input into a latent feature space with a lower dimension. After that, the decoder attempts to reconstruct the original data based on the latent vector. We can optimize the model parameters by minimizing the following MSE loss function:

$$\mathscr{L}_{\mathrm{AE}} = \frac{1}{n}\sum_{i=1}^{n}\left\|\mathrm{Dec}\left(\mathrm{Enc}\left(\mathbf{h}_g\right)\right) - \mathbf{h}_g\right\|^2. \tag{5}$$

The loss function takes ℓ_2-norm distance as the measurement of the reconstruction errors. After several training iterations, the anomaly score of each IPG snapshot can be computed as follows:

$$\mathrm{S}\left(\mathbf{h}_{\mathrm{g}}\right) = \left\|\mathrm{Dec}\left(\mathrm{Enc}\left(\mathbf{h}_g\right)\right) - \mathbf{h}_g\right\|^2. \tag{6}$$

It indicates that the snapshots with large anomaly scores are more likely to be under attack. Thus, we can rank the top-k suspicious hosts and set threshold λ to trigger alerts.

6. Anomaly Detection on the IIG

6.1. Edge Feature-Enhanced GNN. The Interhost Interaction Graph is a combination of host nodes and network-level information flows over the whole target network. For the IIG, the host nodes are relatively uniform, whereas the interactions between them vary in types, including authentication, network flow, and DNS lookup. These events contain important information about the network activities. Consequently, multidimensional edge features of IIG events should be further exploited to capture the interactive information. We propose a novel edge feature-enhanced GNN that leverages the attention mechanism to fully incorporate edge features for the IIG representations.

Given an Interhost Interaction Graph, let $\mathbf{E}_{vu}$ be a tensor of edge features between nodes v and u, and $\mathbf{E}_{vu}^{p}$ is the p^{th} channel of the edge feature in P-dimensional feature vector $\mathbf{E}_{vu}$. In contrast to the existing attention mechanism described in GAT [22], the proposed mechanism allows us to implement attention operations guided by the edge features. For the IIG with various activities, we consider the multidimensional edge features as multichannel signals, and each channel of signals can guide an independent attention operation, respectively. For a specific channel for the p^{th} edge feature, the normalized attention coefficient is computed as follows:

$$\begin{aligned} e_{vu}^{p} &= f\left(\mathbf{h}_v', \mathbf{h}_{\mathrm{u}}'\right)\mathbf{E}_{vu}^{p}, \\ a_{vu}^{p} &= \frac{\exp\left(\mathrm{LeakyReLU}\left(e_{vu}^{p}\right)\right)}{\sum_{s\in\mathscr{N}_v}\exp\left(\mathrm{LeakyReLU}\left(e_{vs}^{p}\right)\right)}, \end{aligned} \tag{7}$$

where $\mathbf{E}_{vu}^{p}$ is the p^{th} channel feature of the edge connecting nodes v and u, and $f(\cdot)$ is an arbitrary attention function that produces a scalar importance value from two input embedding vectors. Here, we adopt a linear function to perform the following:

$$f\left(\mathbf{h}_v', \mathbf{h}_u'\right) = \mathbf{a}^{\top}\left(\mathbf{W}\mathbf{h}_{\mathbf{v}}' \| \mathbf{W}\mathbf{h}_{\mathbf{u}}'\right). \tag{8}$$

At last, we aggregate the neighbor embeddings and the corresponding edge features based on the attention mechanism to generate the new embedding for the target node. The aggregation operation can be formulated as follows:

$$\mathbf{h}_v = \|_{p=1}^{P}\ \sigma\left(\sum_{u\in\mathscr{N}_v}\alpha_{vu}^{p}\mathbf{W}\left(\mathbf{h}_u', \mathbf{E}_{vu}^{p}\right)\right), \tag{9}$$

where $\mathbf{h}_v$ is the new embedding of the target node. We use weight α_{vu}^{p} to sum up neighbor embeddings and the edge feature of a specific channel. The results produced by all different channels of the edge features are combined by the concatenation operation. By doing so, the multidimensional edge features can be seamlessly aggregated into the node embeddings, which helps capture the structural and semantic information across the whole target network.

6.2. Negative Sampling. Due to the characteristics of APT attacks, it is difficult to collect sufficient labeled data that describe APT attack patterns comprehensively. So, it results in poor performance if we train a detection model in a supervised fashion that the labeled data cover only a small part of the possible anomalous operations. Besides, the goal of anomaly detection in IIG is to discriminate the anomalous edges when new events occur in the target network. Thus, we compute the anomaly scores for edges based on the corresponding node embeddings and leverage the negative sampling to train the detection model. Given an IIG snapshot G_t at timestamp t, we produce the vector representations of all host nodes. For each incoming edge (v, u) of G_t, the anomaly score can be computed as follows:

$$\mathrm{S}(v,u) = \omega\cdot\sigma\left(\beta\cdot\left(\left\|\mathbf{a}\odot\mathbf{h}_v + \mathbf{b}\odot\mathbf{h}_u\right\|_2^2 - \mu\right)\right), \tag{10}$$

表征边的异常得分

where $\mathbf{h}_v$ and $\mathbf{h}_u$ denote the vector representations of node v and node u, respectively, and $\sigma(\cdot)$ is the sigmoid function. The node embeddings are parametrized by weight vectors $\mathbf{a}$ and $\mathbf{b}$ that are optimized in the output layer. β and μ represent hyperparameters in the anomaly score function.

To overcome the challenge of insufficient anomaly data, we leverage the negative sampling to train a detection model that learns the normal behaviors of the target network. Specifically, we generate a negative sample for each normal edge as an anomalous edge. The negative sampling means replacing those nodes at the end of normal edges with other nodes in the graph. For example, given normal edge (v, u), the set of selected nodes should be $\mathrm{V}\setminus\mathscr{N}_v$ if node u is replaced. Thus, the edges are assigned to the hosts that originally had no interactions. We define a Bernoulli distribution for the negative sampling: for normal edge (v, u), we corrupt the event by replacing the head host with a probability of $d_v/(d_v + d_u)$ or replacing the tail host with a probability of $d_u/(d_v + d_u)$, where d_v and d_u denote the degree of node v and node u, respectively.

It is worth noting that the generated negative samples may still be normal; thus, the strict loss functions, such as the cross-entropy, cannot be used to discriminate the original edges and the generated ones. Thus, we adopt the margin-based pairwise loss function in training the detection model:

$$\mathscr{L}^t = \min \sum_{(v,u)\in\mathscr{E}^t} \sum_{(v',u')\notin\mathscr{E}^t} \max\{0, \gamma + S(v,u) - S(v',u')\}, \quad (11)$$

samller bigger

where $s(v,u)$ and $s(v',u')$ are the anomaly scores of the existing edges and the generated edges, respectively. $\gamma \in (0,1)$ is the margin between the anomaly scores of the normal edge and the anomalous one. The minimization of the loss function $\mathscr{L}^t$ drives $s(v,u)$ to be smaller while $s(v',u')$ to be larger, which is consistent with our goal.

6.3. Dynamic Model Update. The detection models proposed in this article do not require any anomalous data for training, so they are capable of detecting unforeseen anomaly types such as zero-day attacks exploited in APT campaigns. However, the training data are usually noisy and incomplete, so false positives and false negatives may be generated when we use the trained model for anomaly detection. Note that the IPG describes normal patterns from the perspective of system events and the IIG models normal behaviors at the network level. The promising idea is to combine the IPG and IIG to build an anomaly detection mechanism that fully integrates both sources of information. To this end, we propose an enhancement mechanism that dynamically updates the IIG detection model using the reported malicious hosts by the IPG detectors.

Recall that the IPG anomaly detectors will report a set of hosts that do not conform to the learned normal patterns. Administrators can diagnose the systems by referring to the list of suspicious hosts. Beyond that, APT actors may further move laterally to infiltrate across multiple hosts after compromising a certain target. To detect the subsequent actions of APT attacks, we use the detected suspicious hosts to enhance the IIG detector by dynamically updating the detection model. Note that the false-positive samples may appear in the reported list of suspicious hosts; thus, before updating the IIG model with newly reported hosts from the IPG detectors, the security analysts should manually remove the false-positive samples to ensure that the hosts used for the update are all compromised.

We can update the IIG detection model by defining the loss function $\mathscr{L}^t_{\text{update}}$ as follows:

$$\mathscr{L}^t_{\text{update}} = \min \sum_{(v,u)\notin\mathscr{E}^t} \sum_{(v',u')\in\mathscr{E}^t} \max\{0, \gamma - S(v,u) + S(v',u')\}, \quad (12)$$

where v and u are suspicious hosts reported by the IPG detectors, and (v,u) are edges that may appear between them with a high probability of being malicious. For each potentially malicious edge (v,u), we choose a sample (v',u') as a normal edge that exists in $\mathscr{E}^t$ and is not reported as an anomaly. With this view, the minimization of the update loss function $\mathscr{L}^t_{\text{update}}$ pushes $S(v,u)$ to become larger and $S(v',u')$ to become smaller. With the enhancement from the IPG detectors, the IIG detection model can be efficiently updated to learn new attack patterns. The combination of the IPG and IIG can collectively detect APT activities from a comprehensive perspective.

7. Experimental Evaluation

The efficacy of the proposed method is explored in this section. We conducted all our experimental evaluations on Ubuntu 18.04 LTS, which possesses an Intel Xeon W-2133 3.60 GHz CPU, an NVIDIA GeForce RTX 2080 Ti GPU, and 64 GB RAM. The proposed graph embedding algorithms are implemented using Python's Deep Graph Library (DGL) that leverages the message passing mechanism to build GNNs. We evaluate our approach with different attack scenarios. For the Intrahost Provenance Graph and the corresponding IPG anomaly detector, we use the StreamSpot dataset to test the detection performance of our method at the system level. The Los Alamos National Lab (LANL) dataset is used to evaluate the IIG anomaly detector over the Interhost Interactive Graph from a network perspective.

7.1. Datasets

7.1.1. StreamSpot Dataset. The StreamSpot dataset is composed of 600 provenance graphs derived from 5 benign and 1 attack scenarios. The benign scenarios involve normal behaviors of playing video games, checking emails in Google Mail, browsing cnn.com, downloading files, and watching YouTube videos. The attack scenario involves malicious behaviors of a drive-by download attack triggered by browsing a malicious URL which exploits a Flash vulnerability and further gains root access to the victim host. For each scenario, 100 tasks were executed automatically on a Linux machine. The Linux SystemTap logging system is used to record system call traces from the start of a task until its termination on the machine. All system calls running on the machine (including the ones not from the task) are collected and used to construct provenance graphs for anomaly detection. The statistics of the dataset are summarized in Table 2.

7.1.2. LANL Dataset. The LANL dataset comprises 58 consecutive days of event data collected from the internal computer network of Los Alamos National Laboratory. It involves five data sources, including the network flow data collected from several key routers, DNS lookup events collected from internal DNS servers, process start and stop events collected from individual Windows computers and servers, authentication events collected from Windows-based individual computers and Active Directory servers, and a set of red team events that presents malicious behaviors deviating from normal computer activities. In total, the dataset comprises 1,648,275,307 events for 17,684 computers, involves 749 malicious events with 305

compromised computers which presents a typical APT campaign. We utilize three data sources to evaluate the IIG detector at the network level. Authentication events and internal network flow events are taken into account for detection of the second APT stage as attackers often attempt to compromise more hosts. To detect anomalous hosts that engage in communication with the C & C server and data exfiltration in the last phase of APT, attention is focused on the network flow and DNS lookup events from internal to external hosts.

7.2. Evaluation on the StreamSpot Dataset. We compare the IPG detector with three anomaly detection methods: StreamSpot [13], UNICORN [2], and Graph Attention Network (GAT) [22]. StreamSpot and UNICORN are state-of-the-art approaches for provenance-based APT anomaly detection using heuristic algorithms. GAT is a GNN algorithm that leverages self-attention layers to assign different weights to different neighbor nodes. We use 60% of the benign data to train the detection model and 40% of it along with all attack data for testing. The number of the metapath aggregated GNN layers is 3. The learning rate is 0.003. The weight decay for regularization is $5e-6$. As shown in Table 3, we employ precision, accuracy, recall, and F-score to evaluate different methods. The IPG detector performs best in all evaluation metrics compared with all baselines. In particular, the IPG detector has gained more than 25% improvement compared with the StreamSpot and more than 10% improvement compared with the UNICORN and GAT.

As shown in Table 4, we split the benign graphs into three datasets to fully explore the performance of our method. The Receiver Operator Characteristic (ROC) curves for all datasets are shown in Figure 6; note that the IPG detector is capable of ranking attack graphs correctly with AUC = 0.98 (area under the ROC curve). More specifically, on the ALL dataset, the IPG detector reaches an ideal performance with TPR = 0.98 and FPR = 0.1 when the threshold is set to 0.18 (which means that the graphs with reconstruction errors greater than 0.18 are classified as anomalies). Finally, we evaluate the influence of the training ratio on AUC and Average Precision (AP). Figure 7 shows the variations of AP and AUC with the training ratio varied from 10% to 90%. We note that our method achieves good detection performance when sufficient benign training graphs are employed.

7.3. Evaluation on the LANL Dataset. The LANL dataset is used to evaluate the performance of the IIG detector and the model update mechanism. The event types used for evaluation regarding authentication, network flow, and DNS lookup, which describe the network-level behaviors. The dataset properties are shown in Table 5, with all the three types of events are benign. We randomly select 600 malicious events from the red team file, accounting for 80% of the total. The 600 malicious operations involve 280 suspicious hosts and 254 suspicious users. The target environment involves 4,110 host nodes. We use 4,000 benign events (i.e., 2,000 authentication events, 1,000 flow events, and 1,000 DNS events) to examine FPRs, and the rest of benign events to train the IIG model. All attack data are used to examine the IIG detector's ability to identify malicious events.

Table 2: StreamSpot dataset summary.

Dataset	Scenario	Avg. $\|E\|$	Avg. $\|V\|$	# of graphs
StreamSpot	GMail	37,382	6,827	100
	Download	310,814	8,831	100
	CNN	294,903	8,990	100
	VGame	112,958	8,637	100
	YouTube	113,229	8,292	100
	Attack	28,423	8,891	100

To evaluate the performance of the IIG detector in different scenarios, we split the benign dataset into three subdatasets: authentication and DNS lookup events (AD), authentication and network flow events (AF), and all three types of benign events (AFD). Figure 8 presents the detection results in terms of ROC curves on the three datasets. Note that the IIG detector performs best with AUC = 0.90 when we use all benign events to train the detection model. The AUC of the model trained by the AD and AF datasets are 0.83 and 0.85, respectively. When we use the AFD for training and set the threshold to 0.27, the TPR of the IIG detector is 0.83, and the FPR does not exceed 0.05. It is evident from the ROC curve that increasing the number of event types for model training contributes to modeling network behaviors. As such, anomaly detection models incorporating multiple benign patterns are more likely to identify outliers.

We compare the IIG detector with four baseline methods (Tiresias [25], ensemble method [36], log2vec [3], and GAT [22]) to evaluate the effectiveness of our method. Tiresias is an advanced log-entry-level approach that leverages RNN to predict future events on a host. The ensemble method uses Principal Component Analysis, k-means clustering, and Median Absolute Deviation–based outlier detection to identify anomalies. Log2vec is a heterogeneous graph embedding-based approach that leverages random walk and word2vec to identify malicious events. We implement the IIG detector with a 3-layer edge feature-enhanced GNN. The learning rate is 0.002, and the weight decay for regularization is 5e-6.

As shown in Table 6, we employ precision, recall, *F*-score, and AUC to compare the performances of different methods. The LANL dataset is extremely unbalanced; thus, we do not use the accuracy as a metric, which cannot indicate performance of the detectors when anomalous data are scarce. The ROC curves of the IIG detector and the baselines are shown in Figure 9. Tiresias simply treats events with less than ten occurrences as anomalies. However, the rare events involve numerous benign patterns, which leads to poor performance with the precision and *F*-score less than 0.60. The ensemble method analyzes network events by identifying characteristic patterns as statistical features to detect APT behaviors, but the performance is not ideal. The performances of log2vec

Table 3: Detection results of different approaches on the StreamSpot dataset.

Experiment	Precision	Accuracy	Recall	F-score
StreamSpot	0.77	0.84	0.74	0.75
UNICORN	0.87	0.90	0.84	0.86
GAT	0.86	0.91	0.88	0.87
IPG detector	0.98	0.98	0.98	0.98

Table 4: StreamSpot subdatasets summary.

Dataset	Scenarios	Avg. $\|E\|$	Avg. $\|V\|$	# of graphs
ALL	GMail, download, CNN, VGame, YouTube	173,857	8,315	500
GVC	GMail, VGame, CNN	148,414	8,151	300
YDC	YouTube, download, CNN	239,648	8,705	300

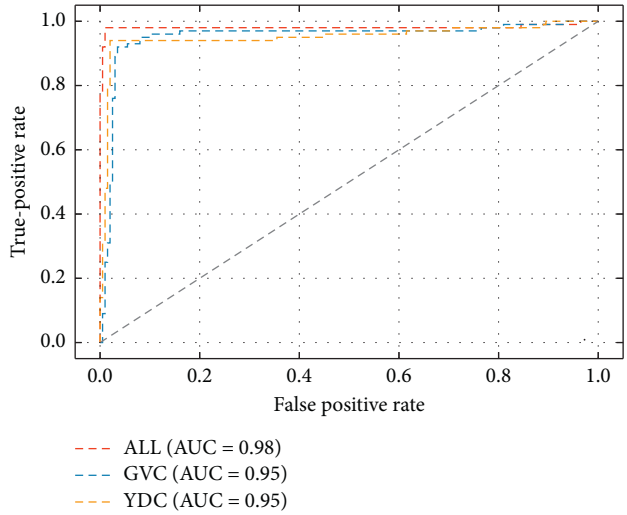

Figure 6: ROC curves of the IPG detector on the StreamSpot subdatasets.

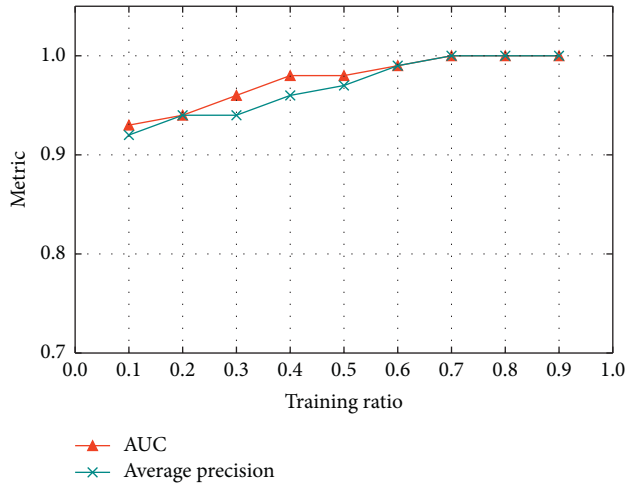

Figure 7: Performance of the IPG detector with different training ratios.

(AUC = 0.85) and GAT (AUC = 0.83) are similar. However, log2vec relies on a heuristic model that uses too many user-defined parameters and only uses statistical information from a fixed hop of neighbors, which ignores the global information of APT activities. GAT is unable to encode network-level events involving multidimensional edge features. The IIG detector incorporates edge features to perform graph embedding, thus producing significant improvement in all evaluation metrics (AUC = 0.9 and all other metrics are 0.83).

After executing malicious code and securing the entry point of the target environment, ATP attackers further search for critical assets and move laterally to exfiltrate information and undermine components. The malicious hosts reported from the system-level anomaly detectors can be further used to enhance the network-level anomaly detector. To evaluate the enhancement mechanism, we randomly label k malicious hosts assuming they are detected by the IIG detectors to update the trained IIG model. By increasing the number of reported hosts from 0 to 10, we investigate the impact of k in the IIG detector. The results are shown in Figure 10, where we employ AUC, AP, and TPR to verify the performance improvement. Compared with the IIG detector without enhancement, we can achieve improved performance as the number of reported malicious hosts increases. Specifically, when we label 10 malicious hosts, the IIG detector performs best with AUC = 0.95, TPR = 0.92, and AP = 0.91.

Table 5: LANL dataset summary.

Dataset	Event types	# of hosts	# of events	# of test events
LANL	Authentication	4,110	2,317,309	2,000
	Network flow	3,740	1,645,377	1,000
	DNS lookup	2,102	974,533	1,000
	Attack	280	600	600

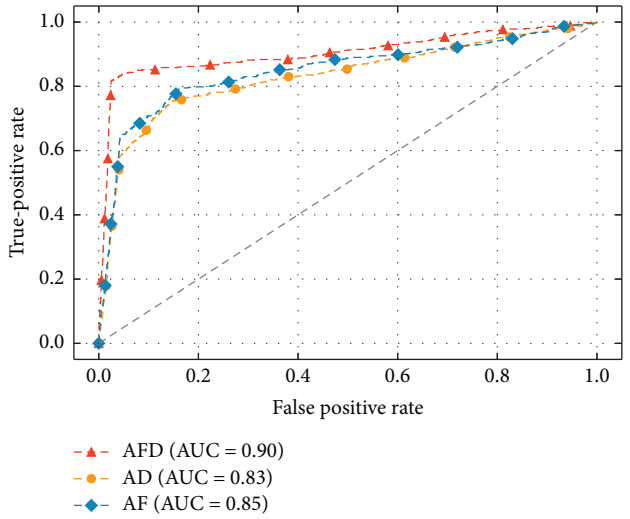

Figure 8: ROC curves of the IIG detector on the LANL subdatasets.

Table 6: Detection results of different methods on the LANL dataset.

Method	Precision	Recall	*F*-score	AUC
Tiresias	0.56	0.63	0.59	0.76
Ensemble method	0.62	0.68	0.65	0.80
Log2vec	0.72	0.78	0.74	0.85
GAT	0.69	0.73	0.71	0.83
IIG detector	0.83	0.83	0.83	0.90

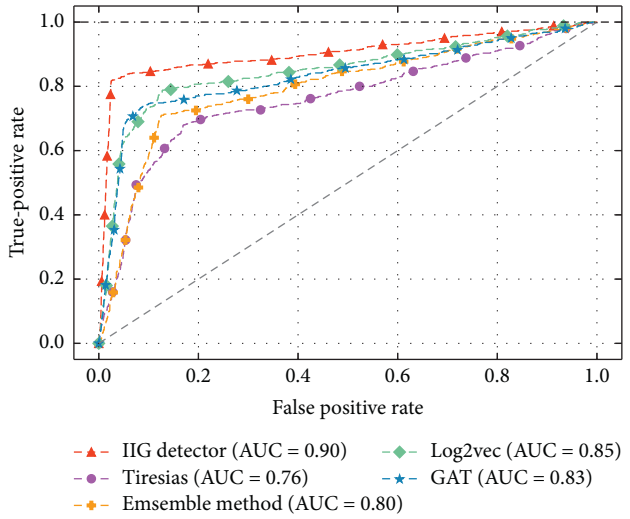

Figure 9: ROC curves of different methods on the LANL dataset.

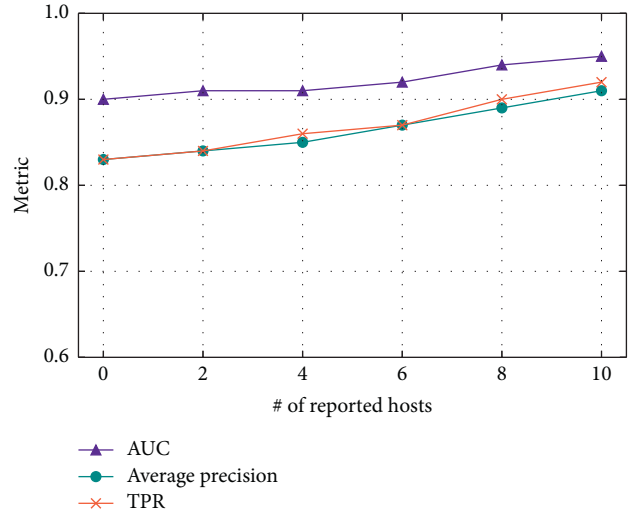

Figure 10: Performance of the IIG detector with different # of reported hosts.

8. Discussion

In this study, we propose a hierarchical framework with two detection models, IPG and IIG, to detect APT attacks. Currently, we evaluate the IPG detector on the StreamSpot dataset and the IIG detector on the LANL dataset separately. However, the two models need to be combined for a more adequate analysis. Besides, the identified anomalies should be further analyzed and mapped to the three-stage APT model. Thus, a complete dataset that records behaviors of a target environment at both the network and system levels is needed. In the future, we could implement a virtual environment to execute normal operations and simulate APT attacks. We could thus evaluate our approach adequately in such an environment.

9. Conclusion

To overcome the limitations of graph-based attack detection in APT studies, we propose a hierarchical approach for defending against APTs with novel attention-based GNNs. To comprehensively capture the behaviors of the full APT lifecycle, we propose a metapath aggregated GNN and an edge feature-enhanced GNN to identify anomalies at both the system and network levels. Besides, we present an enhancement mechanism to dynamically update the IIG model with reported hosts from the IPG detectors in the hierarchical detection framework. The evaluations show that our methods outperform the state-of-the-art baselines.

Data Availability

The StreamSpot data and LANL data used to support the results of this work are publically available at https://github.com/sbustreamspot/sbustreamspot-data/ and https://csr.lanl.gov/data/cyber1/.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China, under Grant 2019YFB2102002, and in part by the Key Research and Development Program of Jiangsu Province, under Grant BE2019012.

References

[1] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan, "Holmes: real-time APT detection through correlation of suspicious information flows," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1137–1152, IEEE, San Francisco, CA, USA, May 2019.

[2] X. Y. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "UNICORN: runtime provenance-based detector for advanced persistent threats," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2020.

[3] F. Liu, Y. Wen, D. X. Zhang, X. H. Jiang, X. Y. Xing, and D. Meng, "Log2vec: a heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1777–1794, ACM, London, UK, November 2019.

[4] A. Zimba, H. Chen, Z. Wang, and M. Chishimba, "Modeling and detection of the multi-stages of advanced persistent threats attacks based on semi-supervised learning and complex networks characteristics," *Future Generation Computer Systems*, vol. 106, pp. 501–517, 2020.

[5] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: techniques, solutions, challenges, and research opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019.

[6] X. Yan and J. Zhang, "Early detection of cyber security threats using structured behavior modeling," *Transactions on Information and System Security*, vol. 5, pp. 1–19, 2013.

[7] A. S. K. Pathan, *The State of the Art in Intrusion Prevention and Detection*, Auerbach Publications, Boca Raton, FL, USA, 2014.

[8] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, ACM, Dallas, TX, USA, November 2017.

[9] P. Parveen, N. McDaniel, V. S. Hariharan, B. Thuraisingham, and L. Khan, "Unsupervised ensemble based learning for insider threat detection," in *Proceedings of the 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pp. 718–727, Amsterdam, Netherlands, September 2012.

[10] M. Du, Z. Chen, C. Liu, R. Oak, and D. Song, "Lifelong anomaly detection through unlearning," in *Proceedings of the SIGSAC Conference on Computer and Communications Security*, pp. 1283–1297, ACM, London, UK, November 2019.

[11] B. Debnath, M. Solaimani, M. A. G. Gulzar, N. Arora, C. Lumezanu et al., "LogLens: a real-time log analysis system,," in *Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1052–1062, IEEE, Vienna, Austria, July 2018.

[12] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 626–688, 2015.

[13] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1035–1044, San Francisco, CA, USA, August 2016.

[14] Z. Li, Q. A. Chen, R. Yang, and Y. Chen, "Threat detection and investigation with system-level provenance graphs: a survey," 2020, https://arxiv.org/abs/2006.01722.

[15] M. N. Hossain, S. M. Milajerdi, J. Wang et al., "Real-time attack scenario reconstruction from COTS audit data," in *Proceedings of the 26th USENIX Conference on Security Symposium*, pp. 487–504, Vancouver, Canada, August 2017.

[16] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. Venkatakrishnan, "Poirot: aligning attack behavior with kernel audit records for cyber threat hunting," in *Proceedings of the SIGSAC Conference on Computer and Communications Security*, pp. 1813–1830, ACM, London, UK, November 2019.

[17] K. Pei, Z. S. Gu, B. Saltaformaggio et al., "Hercule: attack story reconstruction via community discovery on correlated log graph," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pp. 583–595, ACM, Los Angeles, CA, USA, December 2016.

[18] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*, pp. 701–710, New York, NY, USA, August 2014.

[19] A. Grover and J. Leskovec, "Node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*, pp. 855–864, San Francisco, CA, USA, August 2016.

[20] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graph," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, Curran Associates Inc, Long Beach, CA, USA, December 2017.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, Toulon, France, April 2017.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the International Conference on Learning Representations*, Vancouver, Canada, April 2018.

[23] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 594–602, SIAM, Calgary, Canada, May 2019.

[24] L. Zheng, Z. P. Li, J. Li, Z. Li, and J. Gao, "AddGraph: anomaly detection in dynamic graph using attention-based temporal GCN," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence Main Track*, pp. 4419–4425, Macao, China, August 2019.

[25] Y. Shen, E. Mariconti, P. A. Vervier, and G. Stringhini, "Tiresias: predicting security events through deep learning,"

in *Proceedings of the SIGSAC Conference on Computer and Communications Security*, pp. 592–605, ACM, Toronto Canada, October 2018.

[26] A. Gehani and D. Tariq, "SPADE: support for provenance auditing in distributed environments," in *Proceedings of the 13th International Middleware Conference*, pp. 101–120, Montreal, Canada, December 2012.

[27] T. Pasquier, X. Han, M. Goldstein et al., "Practical whole-system provenance capture," in *Proceedings of the 2017 Symposium on Cloud Computing*, pp. 405–418, ACM, Santa Clara, CA, USA, September 2017.

[28] S. T. King and P. M. Chen, "Backtracking intrusions," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 223–236, 2003.

[29] Y. Liu, M. Zhang, D. Li et al., "Towards a timely causality analysis for enterprise security," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2018.

[30] W. U. Hassan, S. Guo, D. Li et al., "Combatting threat alert fatigue with automated provenance triage," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2019.

[31] P. Moriano, J. Pendleton, S. Rich, and L. J. Camp, "Insider threat event detection in user-system interactions," in *Proceedings of the 2017 International Workshop on Managing Insider Security Threats*, pp. 1–12, ACM, Dallas, TX, USA, October 2017.

[32] Y. Han, A. P. Li, and R. Jiang, "Needle in a haystack: attack detection from large-scale system audit," in *Proceedings of the 2019 IEEE 19th International Conference on Communication Technology (ICCT)*, pp. 1418–1426, Xi'an, China, October 2019.

[33] E. Hutchins, M. Cloppert, and R. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," in *Proceedings of the 6th International Conference on Information Warfare and Security*, Washington, DC, USA, March 2011.

[34] D. J. Pohly, S. McLaughlin, P. McDaniel, and K. Butler, "Hi-fi: collecting high-fidelity whole-system provenance," in *Proceedings of the Computer Security Applications Conference*, pp. 259–268, Orlando, FL, USA, December 2012.

[35] A. M. Bates, D. Tian, K. R. Butler, and T. Moyer, "Trustworthy whole-system provenance for the Linux kernel," in *Proceedings of the 24th USENIX Security Symposium*, pp. 319–334, USENIX, Washington, DC, USA, August 2015.

[36] A. Bohara, M. A. Noureddine, A. Fawaz, and W. H. Sanders, "An unsupervised multi-detector approach for identifying malicious lateral movement," in *Proceedings of the 2017 IEEE 36th Symposium on Reliable Distributed Systems*, pp. 224–233, IEEE, Hong Kong, China, September 2017.