electronics

MDPI

Article

Hybrid Time–Position Embedding for Provenance-Based Intrusion Detection

Seonghyeon Gong [1], Jake Cho [2,*] and Kyuwon Ken Choi [1]

[1] Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616, USA; sgong3@illinoistech.edu (S.G.); kchoi12@illinoistech.edu (K.K.C.)
[2] Department of Computer Science, Lewis University, Romeoville, IL 60446, USA
* Correspondence: jcho1@lewisu.edu

Abstract

Provenance-based Intrusion Detection Systems (IDSs) model the causal relationships between security events through a provenance graph and learn contextual information to detect Advanced Persistent Threats (APTs) effectively. However, existing provenance graph representation methods fail to fully reflect the characteristics of security domain data and the semantic information embedded in system logs, resulting in limited learning efficiency and detection accuracy. This paper proposes a provenance representation method that effectively captures security context from system log data. The proposed method improves the performance of provenance-based IDSs by combining (1) a provenance graph construction technique that transforms meaningful string attributes—such as command lines, process names, and file paths—into vector representations to extract semantic information in the security context, (2) a hybrid time–position embedding technique for capturing causal relationships between events, and (3) an iterative refinement learning strategy tailored to the characteristics of system log data. Experimental results using the DARPA Transparent Computing Engagement 3 (E3) benchmark dataset for APT detection demonstrate that our method achieves improved accuracy compared to existing approaches while significantly accelerating convergence during iterative training. These results suggest that the proposed embedding technique can more effectively capture abnormal temporal patterns, such as the long dwell times characteristic of APT attacks.

Keywords: data provenance; intrusion detection; embedding; semantic information

1. Introduction

As information technology advances and networks become increasingly complex, the scale and impact of cyberattacks continue to grow. As demonstrated by high-profile incidents such as Stuxnet [1], the Sony Pictures hack [2], NotPetya [3], and the SolarWinds breach [4], APTs can result in catastrophic consequences for nations, societies, and enterprises [5]. IDSs are essential for responding to such incidents; however, they face significant limitations in detecting and responding to sophisticated attacks like APTs [6]. Traditional rule-based and signature-based IDSs effectively identify known, isolated attack patterns. However, these systems fail to detect novel or stealthy APT attacks that exploit zero-day vulnerabilities or use evasion techniques [7]. Moreover, their lack of behavioral analysis makes detecting complex, multi-stage attack sequences difficult [8]. To overcome the limitations of traditional IDS approaches, many studies have explored machine learning-based IDS (ML-IDS), which has shown some success in detecting APTs [9].

Check for updates

Academic Editor: Yu-an Tan

Received: 3 November 2025
Revised: 21 February 2026
Accepted: 23 February 2026
Published: 28 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Nevertheless, ML-IDSs also have limitations: they struggle to capture causal relationships among security events, which is crucial for detecting the staged behaviors of APTs that often involve long, stealthy periods [10]. Furthermore, their performance heavily depends on the quality of the training dataset and the domain expertise applied during feature engineering. These challenges hinder the development of generalizable detection models and can lead to high false-positive rates and overfitting [11].

To address the challenges posed by highly structured and stealthy APT attacks, data provenance-based approaches have recently gained significant attention. Recent comprehensive surveys [12] confirm that provenance graphs remain a foundational technology for state-of-the-art intrusion detection, offering unparalleled visibility into complex attack behaviors. Data provenance represents the contextual and causal relationships between system events using a directed acyclic graph (DAG) structure [13]. This representation enables the modeling of complex system behaviors and supports learning attack patterns, making it suitable for detecting APTs. As a result, a growing body of research has explored provenance-based IDS to detect and classify sophisticated attacks.

However, provenance-based IDSs still face several limitations: (1) Neglect of Semantic Information: Security event logs often include not only universally unique identifiers (UUIDs) for processes and network sockets but also rich semantic attributes such as executable commands, timestamps, and file paths [14,15]. These attributes may contain important contextual clues depending on the attack scenario [16]. Nevertheless, many IDS approaches disregard such information due to challenges in generalization and representation. (2) Difficulty in Inferring Temporal Causality: Provenance graphs typically establish edges based on the order of event occurrence. However, this approach overlooks the actual time intervals between events, limiting the system's ability to infer precise causal relationships [17]. For instance, APT attacks often exploit long dwell times and mimic normal behavior between lateral movements. Detection strategies that rely solely on event order may fail to capture such attack patterns [18].

In this research, we propose a novel provenance graph generation method that addresses the limitation of semantic information neglect in provenance-based IDS by leveraging a hybrid time–position embedding mechanism. Our method parses system logs to extract subject–type–object triples and accordingly constructs provenance nodes and edges. We enrich node attributes by aggregating contextual information derived from each subject's behavior.

Furthermore, we incorporate time information into the embedding process to support accurate causal inference. Rather than encoding only the procedural order of events, our method also embeds the actual time intervals between them. This enables the model to reason about temporal relationships and detect abnormalities in timing patterns, which is especially effective for capturing APT behaviors characterized by long periods of inactivity.

Our approach employs a graph neural network (GNN) [19] to learn from the constructed provenance graph. To improve learning efficiency, we further refine the training data through an iterative mechanism that considers the unique characteristics of security datasets.

The main contributions of this study are as follows:

- We propose a novel provenance representation method using a hybrid time–position embedding mechanism. This method generates provenance graphs that better express system behavior by embedding procedural order and time intervals from system logs. It can be broadly applied to the embedding process of raw security data and contributes to improving dataset quality, particularly during preprocessing and dataset construction.

- We demonstrate that models trained on provenance graphs generated using our log embedding technique can more effectively capture the semantic information in system logs. Through experiments on the DARPA E3 benchmark dataset, we show that the GNN model trained on our proposed representation converges more quickly than baseline methods, indicating the effectiveness of our embedding in expressing node attributes in the provenance graph.

The rest of the paper is organized as follows: Section 2 provides an overview of existing ML-based and provenance-based IDS methods for detecting APTs. Section 3 details the mechanism and algorithm of the proposed time-aware provenance-based IDSs. In Section 4, we present the experimental results on the DARPA E3 benchmark dataset to demonstrate the effectiveness of our approach. Finally, Section 5 concludes the paper.

2. Related Work

This section reviews existing approaches to provenance-based intrusion detection systems (IDSs) and the efforts made to address their limitations. We mainly focus on provenance graph embedding techniques aimed at encoding sufficient semantic information from the perspective of IDSs and recent studies that attempt to embed semantic attributes into provenance representations.

2.1. Provenance- and Causality-Based Anomaly Detection Approaches

ProvDetector [20] leverages Doc2Vec [21] representations to model process-level operating system logs and generate provenance graphs for detecting stealthy malware. TRACE [22] constructs provenance and causal graphs by tracing execution units induced by loop structures in program behavior, targeting real-time APT detection. ATLAS [14] identifies abstract attack strategies commonly shared among APT campaigns and builds a sequence-based model using audit logs and causality graphs to propose a generalized detection strategy. ThreaTrace [23] presents a node-level anomaly detection method based on host system logs. Unlike graph-level approaches, it focuses on fine-grained detection by classifying anomalies at the individual node level. The framework uses a multi-model architecture to enhance classification performance and applies GraphSAGE [24] to analyze the causal dependencies between nodes. However, it does not incorporate semantic information within the embedding process.

PalanTír [25] addresses the dependency explosion problem in traditional log-based detection by analyzing causality at the instruction level within system call traces. Although this approach enables detailed tracking of attack stages at the command level, it still lacks explicit consideration of semantic information. Paradise [26] proposes a scalable, distributed provenance-based IDS capable of real-time analysis of large-scale logs. It integrates a Kafka-based infrastructure [16] to support high-throughput log ingestion and real-time intrusion detection. DEPCOMM [27] introduces a method for graph summarization by identifying process-centric communities from causal dependency graphs and pruning redundant edges caused by repetitive benign behaviors. This method reduces subgraph size while improving detection performance. Angus [28] proposes an active learning framework for provenance-based IDS to enhance learning efficiency, even in scenarios with extremely limited labeled data. The study demonstrates that high detection performance can still be achieved under label-scarce conditions.

While these studies have improved provenance-based IDS performance in terms of scalability, learning efficiency, and detection accuracy, they largely overlook the unique characteristics of security domain data. In particular, few approaches focus on mechanisms to extract and encode rich semantic and contextual information embedded within raw system logs into the provenance representation.

2.2. Semantic Representation for IDS

Attack2Vec [29] proposed a temporal word embedding technique to capture semantic information across different stages of sophisticated attacks. This approach treats each event as a word and sequences of events as sentences, enabling the modeling of vector representations along the temporal axis. While this work provides a foundation for handling sequences of events, it does not capture causal relationships between attack events. Watson [30] introduced a semantic inference and behavior abstraction mechanism to automate the extraction of high-level activity information. By clustering inferred behaviors, the system identifies a small number of representative audit logs from large-scale security data, thereby facilitating more efficient analysis. DEEPCASE [31] presented a semi-supervised approach for learning contextual representations and identifying correlations among security events. This method does not rely heavily on labeled data and achieves high detection accuracy through contextual embeddings based on surrounding events. FLASH [32] proposed a provenance graph representation and learning framework for APT detection by combining Word2Vec-based semantic encoding with GNN-based structural encoding. To overcome the limitations of Word2Vec in ignoring sequential information, FLASH incorporates Transformer-style positional embedding to enhance APT detection performance. Most existing efforts to extract semantic information from security data focus on log embedding techniques. However, due to the nature of APTs—such as long dwell times and anomalous execution delays—more sophisticated causal reasoning over security data nodes is necessary. Recently, self-supervised learning paradigms have been introduced to enhance semantic representation. MAGIC [33] proposes a masked graph representation learning framework that reconstructs masked subgraphs to learn robust feature representations. By identifying outliers at both system-entity and batched-log levels, it effectively captures fine-grained semantic patterns within provenance graphs without relying on extensive labeled datasets.

2.3. Temporal and Positional Encoding for Provenance Graphs

Incorporating temporal and positional information into semantic and contextual representations is critical for accurately modeling APT characteristics in the security domain. UNICORN [34] models long-term behavior through graph sketching techniques to detect slow-acting APT attacks. While this approach effectively summarizes entire event flows and improves detection, it still leaves room for improvement in accurately modeling event causality based on timestamp data. PSSID [15] addresses evasive behavior in provenance-based IDSs by using the longest common subsequence (LCS) metric to measure similarity between provenance sequences and rule-based reference sequences. Although this method is robust against reordered execution and noise injection, it lacks semantic and temporal encoding capabilities. ProGrapher [35] combines time-series-based snapshot generation with whole-graph embedding to detect APTs. By incorporating event ordering, it provides temporal awareness and generates key indicators to assist analysts, thereby enhancing the practical utility of the system. PG-AID [36] generates provenance graphs using meta-path Doc2Vec [21] embeddings and Transformer-based attention mechanisms. It leverages attention vectors to introduce positional encoding into provenance representations. Recent advancements like ORTHRUS [37] emphasize the importance of spatiotemporal dynamics by utilizing dynamic graph structures with explicit edge timestamps. By learning the temporal and structural patterns of benign behavior in a self-supervised manner, it effectively identifies anomalies that deviate from typical temporal sequences.

While many prior works incorporate positional embeddings into security context modeling, relying solely on node order in provenance graphs is insufficient for fully capturing the semantic characteristics of APTs.

3. Method

In this section, we present our hybrid time–position embedding method designed to improve the learning efficiency of semantic information in provenance-based intrusion detection systems and enhance their detection performance against APTs. As illustrated in Figure 1, our approach consists of a five-stage pipeline. The Provenance Graph Parsing module takes system log data as input and generates a provenance graph that incorporates a wide range of semantic information relevant to the security domain. The Node Attribute Embedding and Hybrid Time–Position Embedding stages perform vectorization of this semantic information, embedding contextual and temporal characteristics into node representations. The resulting embedded graph data is then used to train a GNN, which learns both structural and semantic patterns within the graph. Finally, in the Attack Detection phase, the trained GNN performs node-level anomaly detection to identify potential attack behaviors.

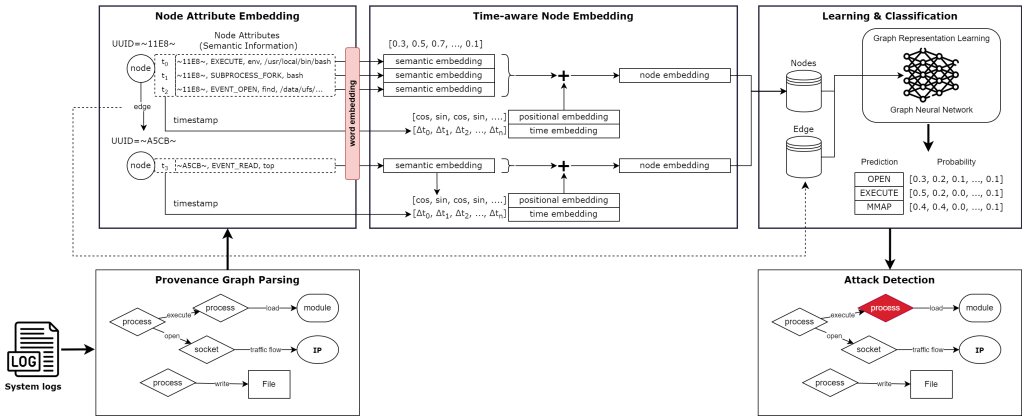


Figure 1. Overall architecture of the proposed provenance-based intrusion detection system.

3.1. Provenance Graph Parsing

Our method constructs a provenance graph $G = (V, E)$ from system log data, where V represents system entities and E represents causal dependencies. The graph generation process is structured as follows:

Nodes: We utilize Universally Unique Identifiers (UUIDs) present in the logs to uniquely identify system entities such as processes, files, and network sockets. Each unique UUID maps to a single node $v \in V$ in the provenance graph.

Edges: Edges represent the causal relationships defined by system events (e.g., fork, read, write, connect). An interaction between a subject entity (e.g., a process) and an object entity (e.g., a file) creates a directed edge $e \in E$ from the subject node to the object node.

Node Attributes: To capture the semantic context of each entity, we aggregate all log entries associated with a specific node into a sequential document. For each event involving a node, we extract critical string attributes—specifically the process command line, event and action types, and file and resource paths.

These attributes are appended in chronological order based on event timestamps to form a semantic sequence $S_v = \{w_1, w_2, \ldots, w_n\}$ for node v. This sequence effectively acts as a textual history of the node's behavior, which serves as the input for the subsequent embedding phase. Edges between nodes are determined by event types, while each node is enriched with semantic information extracted from the log entry, including event types, executed commands, process names, and file paths—attributes highly relevant to the security domain. At this stage, each node in the provenance graph is associated with a text sequence composed of meaningful tokens from the original system log, effectively forming a sentence representation for that node's attribute.

3.2. Semantic Information and Hybrid Time–Position Embedding

Identifiers such as UUIDs, process names, and file paths serve as critical semantic tokens in security contexts. To encode these attributes, we employ the Word2Vec model [38]. This choice is driven by the unique characteristics of system logs, which consist of repetitive operational tokens and identifiers lacking the linguistic polysemy found in natural language. In this domain, capturing local co-occurrence patterns is more efficient and effective than deploying heavy contextual models like BERT. This rationale aligns with prior log embedding approaches such as Log2Vec [39] and FLASH [32], which have successfully demonstrated the efficacy of Word2Vec in embedding security event attributes into dense vector spaces.

In our method, if multiple events are associated with a single node (e.g., multiple edges connected to the same entity), we aggregate the corresponding log entries into a single sequence to represent the node's attribute. As a result, each node's attribute is represented by a variable-length set of log tokens, which is then embedded into a fixed-length vector using Word2Vec. While this approach effectively captures semantic attributes, it discards the sequential order and timing information of the log entries—information that is critical for detecting APTs.

To mitigate the loss of temporal context, we propose a hybrid time–position embedding technique. While several existing provenance embedding approaches use positional encoding to address the loss of sequential information [32,36], we argue that such methods are insufficient for capturing the behavior of stealthy attacks, particularly APTs. APTs are often characterized by seemingly benign actions and prolonged dwell times, which cannot be captured by sequence order alone.

To address this, we extend the standard positional embedding mechanism used in self-attention models [40] by incorporating a temporal embedding that reflects the actual time intervals between events. Specifically, for each node, we normalize the time gaps between individual log entries and the starting timestamp of the node. This temporal information is embedded as a supplementary vector and added to the semantic embedding output to form the final node representation.

This approach enables the model to distinguish patterns such as abnormal delays between process invocations or prolonged inactivity—both of which are typical indicators of APT behavior. By embedding such time-aware features, our method enhances the graph's capacity to model temporal causality and improves its effectiveness in detecting stealthy and persistent threats.

Algorithm 1 and Equation (1) illustrate the embedding process of our proposed method. Given a sequence of tokens associated with a node attribute, denoted as $X = [x_1, x_2, \ldots, x_n]$, we first generate the initial word embeddings $x_i \in \mathbb{R}^d$ using a pre-trained Word2Vec model. Let τ_i denote the timestamp corresponding to token x_i. Our hybrid time–position embedding for each token is defined as:

$$E_i = x_i + p_i + t_i, \tag{1}$$

p_i is the standard sinusoidal positional embedding, defined as:

$$p_i^{(2k)} = \sin\left(\frac{i}{10{,}000^{2k/d}}\right), \quad p_i^{(2k+1)} = \cos\left(\frac{i}{10{,}000^{2k/d}}\right), \tag{2}$$

and t_i is the temporal embedding derived from the continuous time interval. To effectively capture temporal patterns across varying scales (from milliseconds to hours), we employ a log-scaled sinusoidal encoding:

$$t_i^{(2k)} = \sin\left(\frac{\Delta\tau_i'}{10{,}000^{2k/d}}\right), \quad t_i^{(2k+1)} = \cos\left(\frac{\Delta\tau_i'}{10{,}000^{2k/d}}\right), \tag{3}$$

where $\Delta\tau_i' = \log(\Delta\tau_i + 1)$ is the log-normalized relative time difference from the first event timestamp ($\Delta\tau_i = \tau_i - \tau_1$). This formulation avoids the explosion of values for long-duration attacks while preserving fine-grained temporal resolution for short bursts.

The complete time–position embedded sequence is represented as:

$$H = [E_1, E_2, \ldots, E_n] \in \mathbb{R}^{n\times d}. \tag{4}$$

We apply mean pooling over the sequence to generate a fixed-size node embedding vector z_{node}:

$$z_{node} = \frac{1}{n}\sum_{i=1}^{n} E_i. \tag{5}$$

Algorithm 1 Hybrid Time–Position Embedding

Require: Word list $W = [w_1, w_2, \ldots, w_n]$, timestamp list $\mathcal{T} = [\tau_1, \tau_2, \ldots, \tau_n]$, word2vec model $\mathcal{M}$, embedding dimension d
Ensure: Hybrid embedding vector $z_{node} \in \mathbb{R}^d$

$P \leftarrow \mathbb{R}^{n\times d}$ ▷ Initialize positional embedding matrix P
$\Delta\mathcal{T} \leftarrow [\tau_1 - \tau_1, \tau_2 - \tau_1, \ldots, \tau_n - \tau_1]$ ▷ Normalize timestamps
$X \leftarrow [\mathcal{M}(w_1), \ldots, \mathcal{M}(w_n)]$ ▷ Convert words to embeddings
$T \leftarrow \text{Sinusoidal}(\Delta\mathcal{T})$ ▷ Compute temporal embeddings
$E_i \leftarrow X_i + P_i + T_i \quad \forall i \in [1, n]$ ▷ Compute hybrid embeddings
$z_{node} \leftarrow \frac{1}{n}\sum_{i=1}^{n} E_i$ ▷ Aggregate by mean
z_{node}

3.3. Graph Representation Learning

To detect abnormal behaviors and malicious nodes in the provenance graph, it is essential to learn structural representations that incorporate the contextual information of neighboring nodes. To this end, we adopt a graph embedding approach based on Graph Neural Networks (GNNs).

GNNs enable each node to update its embedding by aggregating information from its neighbors, effectively capturing the underlying structure of the graph. In our framework, we construct a 2-layer GNN model based on GraphSAGE [24], using the node feature vectors obtained from the hybrid time–position embedding as input. The node representation at layer k is computed as follows:

$$\mathbf{h}_v^{(k)} = \sigma\left(W^{(k)} \cdot \text{AGG}^{(k)}\left(\left\{\mathbf{h}_v^{(k-1)}\right\} \cup \left\{\mathbf{h}_u^{(k-1)}, \forall u \in \mathcal{N}(v)\right\}\right)\right) \tag{6}$$

where $\mathbf{h}_v^{(0)} = \mathbf{X}_v$ is the initial node embedding, σ is the ReLU activation function, $W^{(k)}$ is the trainable weight matrix, and $\text{AGG}^{(k)}$ is a mean-based neighborhood aggregation function. This architecture effectively reflects structural patterns in the provenance graph.

Attack detection using provenance graphs is computationally expensive due to the large-scale graph structure and the sparse nature of attack behaviors. Most system logs originate from repetitive benign activities, resulting in redundant and semantically neutral patterns in the provenance graph. Consequently, a naïve node-level classification approach using GNNs is not only inefficient in terms of computational cost but also ineffective in focusing on critical anomalous behaviors.

To address this issue, we adopt a confidence-based iterative refinement strategy inspired by FLASH [32]. After performing predictions on all nodes using the trained GNN, we compute the confidence score of each node based on the softmax probabilities of the top two classes:

$$\text{Confidence}(v) = \frac{\hat{y}_1 - \hat{y}_2}{\hat{y}_1} \tag{7}$$

where $\hat{y}_1$ and $\hat{y}_2$ are the highest and second-highest softmax scores, respectively. Nodes that are correctly predicted with high confidence are removed from the training pool, while the model continues training only on the remaining uncertain (potentially suspicious) nodes.

This iterative refinement strategy enables the model to focus on more challenging samples, enhancing detection precision while saving computational resources. Ultimately, this process allows for more accurate identification of suspicious nodes, which can be used to reconstruct the attack flow and support post hoc analysis.

3.4. Attack Detection Phase

We utilize a model trained on the provenance graph to detect abnormal behavior at the node level. One of the node attributes, 'type', serves as the classification label for the classifier. This approach is based on the assumption that behaviors not observed in the general category are considered abnormal. In the final detection phase, we employ an ensemble of 22 GNN snapshots saved during the iterative refinement process. This configuration aligns with the baseline method [32] to ensure a fair comparison. To minimize false negatives, we adopt a conservative voting strategy: a node is classified as benign only if all ensemble models predict it as benign. If any single model classifies the node as anomalous (i.e., misclassifies its type), it is flagged as a potential threat. This rigorous criterion ensures high sensitivity to stealthy attack behaviors, thereby reducing the risk of missed detections.

4. Experiment and Evaluation

In this section, we present the experimental setup and results to evaluate the effectiveness of the proposed Hybrid Time–Position Embedding. We designed our experiments to address the following three key research questions (RQs). All experiments were conducted on a machine with 48 Intel vCPUs, 256 GB of RAM, and Debian 12 OS.

- RQ1. Does Hybrid Time–Position Embedding improve node classification performance over traditional embedding methods?
- RQ2. Does Hybrid Time–Position Embedding achieve better convergence during iterative refinement than positional-only embedding?
- RQ3. Is the proposed method effective in detecting specific APT phases such as Initial Access, Persistence, and Exfiltration across the attack lifecycle?

4.1. Dataset

We utilize the E3 dataset released by DARPA as part of its Transparent Computing (TC) program. This dataset was developed during Engagement 3 (E3) to support research on real-time and forensic-based APT detection and traceability techniques. The E3 dataset contains fine-grained system event logs, which can be structured into provenance graphs representing interactions between system entities.

Crucially, the dataset includes both normal system operations and sophisticated APT scenarios executed by a red team, making it highly suitable for intrusion detection research. To ensure experimental reproducibility and fair comparison, we strictly adopted the data evaluation protocol established by the baseline methodology [32]. Crucially, we employed a time-sequential splitting rather than random shuffling to reflect real-world deployment

scenarios. Specifically, for each dataset, system events were chronologically ordered based on their timestamps. The initial segments of the records, representing baseline system behavior, were designated as the training set, while the subsequent segments containing Advanced Persistent Threat (APT) scenarios were reserved for testing. In practice, for both the Cadets and Trace datasets, we selectively designated one of the earlier JSON log files as the training set and a subsequent JSON file as the test set. This chronological partitioning ensures that the model is trained on past events and evaluated on unseen future activities, thereby preventing data leakage and strictly validating the system's ability to detect novel attack patterns in a temporal context.

Table 1 summarizes the statistical characteristics of the datasets used in our evaluation. Each log entry (event) consists of eight primary features: (1) actorID and objectID (unique identifiers for system entities), (2) actor_type and object (entity categories such as SUBJECT_PROCESS or FILE_OBJECT_UNIX_SOCKET), (3) action (type of system call, e.g., EVENT_CLOSE, EVENT_FORK), (4) timestamp (nanosecond-precision time), (5) exec (executable command string), and (6) path (file or socket path string).

As shown in Table 1, the Cadets dataset exhibits the highest structural and semantic complexity, containing 85 unique executables and over 96,000 distinct paths. This diversity in system behavior makes it significantly more challenging for detection models to establish a stable baseline of normal activity compared to datasets with more uniform patterns (e.g., Trace), thus directly impacting the convergence speed during iterative refinement.

Table 1. Statistical summary of the DARPA TC E3 datasets. The complexity is indicated by the variety of executables (Execs) and file paths (Paths).

Dataset	Events	Nodes	Execs	Paths	Malicious Nodes
Cadets	2,059,154	362,645	85	96,019	12,858
Trace	2,472,290	1,271,939	1	52,989	68,173

4.2. Baselines

To evaluate the effectiveness of the proposed Hybrid Time–Position Embedding, we compare our method against two representative provenance-based intrusion detection systems: ThreaTrace [23] and FLASH [32].

- **ThreaTrace**: ThreaTrace identifies abnormal behavior by modeling the distributional patterns of system calls (syscall distributions). It characterizes process behavior through multi-model frameworks to maximize detection performance.
- **FLASH**: FLASH leverages Word2Vec-based semantic embeddings combined with positional encoding to construct meaningful node representations. It integrates a GNN with a lightweight classifier and applies iterative refinement to boost detection accuracy. FLASH also demonstrated superior scalability through its efficient architecture.

4.3. Implementation Details

To ensure a fair comparison with the baseline methods [23,32], we aligned our hyperparameter settings with theirs where applicable. Log sequences contained in node attributes are embedded into 30-dimensional vectors using Word2Vec (window size of 5, minimum count of 1, and 300 epochs using the CBOW architecture). The proposed Hybrid Time–Position Embedding augments these vectors by incorporating sinusoidal positional encoding and a log-scaled temporal embedding derived from timestamp intervals. We utilize a 2-layer GraphSAGE model [24] with 32 hidden units per layer, trained using the Adam optimizer with a learning rate of 0.01 and a weight decay of 5×10^{-4}. The training objective is based on cross-entropy loss, with class imbalance mitigated via class-weighted loss. Iterative refinement is performed over a fixed number of 22 epochs; after each round,

nodes that are correctly predicted with a confidence score of 0.9 or higher are masked out from the training data loader, without re-partitioning the underlying dataset.

4.4. Evaluation Metrics

To evaluate the effectiveness of the proposed method, we utilize the following standard performance metrics:

- Precision, Recall, and F1-score: Precision measures the accuracy of malicious alerts, while Recall (Detection Rate) quantify the ability to capture all actual attack nodes. The F1-score provides the harmonic mean of these two, representing the overall balanced performance.
- False-Alarm Rate (FAR): Also known as the false positive rate, FAR is a crucial metric in the IDS domain. It represents the probability that benign system activities are incorrectly flagged as malicious (FP/(FP + TN)).
- PR-AUC: This represents the area under the Precision–Recall curve. In our ensemble-based detection, the PR-AUC is calculated by varying the consensus threshold (the number of models flagging an anomaly), offering a comprehensive view of the model's performance beyond a single operational point.

4.5. RQ1: Classification Performance by Embedding Strategy

Table 2 presents the classification results comparing different embedding strategies. All models were trained on the same dataset and evaluated using precision, recall, and F1-score. Our proposed embedding method outperforms existing approaches across all metrics. The improvement is particularly notable for nodes rich in semantic information (e.g., processes and network flows), indicating that incorporating timestamp intervals allows the model to capture both sequential and contextual aspects of the logs better.

Table 2. Comparison of detection performance across different methods on the DARPA E3 dataset (Cadets and Trace).

Method	Metric	Cadets	Trace
ThreaTrace [23]	Precision	0.90	0.72
	Recall	0.99	0.99
	F1-score	0.95	0.83
Flash [32]	Precision	0.952	0.951
	Recall	0.999	0.986
	F1-score	0.962	0.969
	FAR	0.002964	0.003124
	PR-AUC	0.9102	0.9838
Proposed	Precision	0.967	0.953
	Recall	0.999	0.988
	F1-score	0.983	0.969
	FAR	0.00132	0.003077
	PR-AUC	0.9257	0.9869

4.6. RQ2: Convergence Properties in Iterative Refinement

This experiment investigates whether the proposed Hybrid Embedding enables better convergence behavior compared to positional-only embedding. Each model starts from the same initial GNN configuration, and the number of misclassified samples is tracked after each refinement round. This metric reflects how efficiently the model stabilizes during training.

As shown in Figures 2 and 3, the proposed Hybrid Time–Position Embedding demonstrates faster and more stable convergence compared to the baseline FLASH (which uses

positional-only embeddings). Our method significantly reduces the number of misclassified nodes in the early rounds (especially within rounds 0 to 7), while the baseline shows a slower, more gradual decrease. Moreover, our model reaches convergence earlier and maintains a lower misclassification rate in later rounds. These results confirm that incorporating timestamp information improves model decision-making during iterative refinement and yields more favorable convergence behavior than simple positional encoding alone.

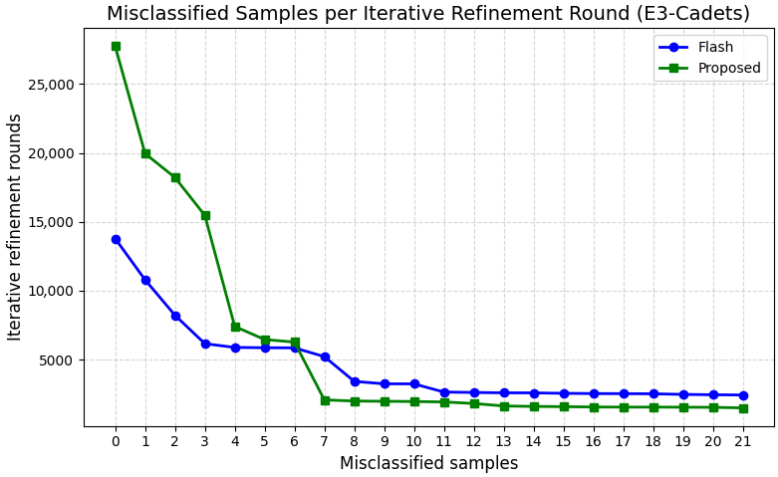


Figure 2. Convergence comparison of iterative refinement between baseline (FLASH) and the proposed method on the Cadets dataset (addresses RQ2).

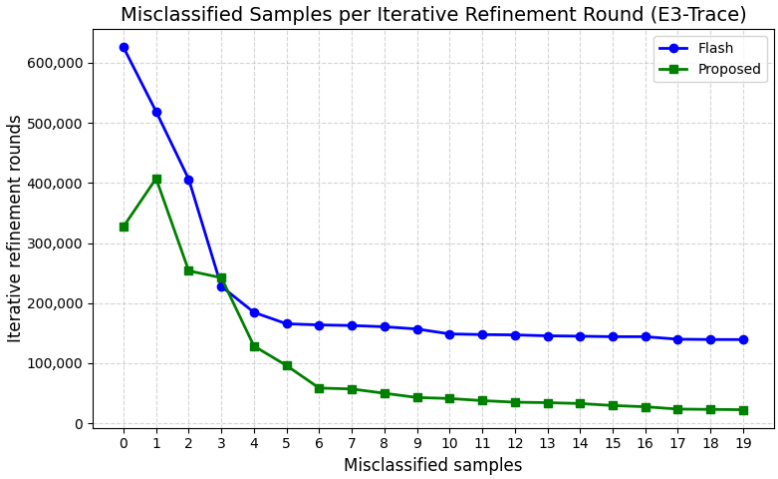


Figure 3. Convergence comparison of iterative refinement between baseline (FLASH) and the proposed method on the Trace dataset (addresses RQ2).

4.7. RQ3: Detection Performance by Attack Stage

This experiment evaluates whether the proposed solution can effectively identify malicious activities across different stages of an APT lifecycle. Rather than relying solely on node-level metrics, we analyzed the volume of malicious events (log entries) associated with True Positives (TP), False Positives (FP), and False Negatives (FN). This granular breakdown allows us to identify specific attack patterns where the model excels or faces challenges (detailed mapping rules for each attack stage are provided in Appendix A).

As shown in Table 3, the proposed method successfully identifies the vast majority of malicious operations across both datasets. The system demonstrates robust detection performance in the Discovery and Collection phases, effectively capturing the bulk of events associated with early-stage reconnaissance and data gathering.

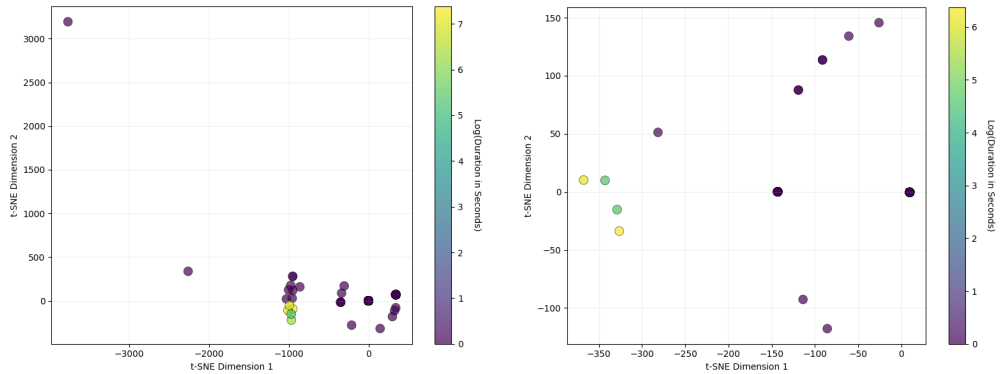
However, the analysis also reveals specific challenges. The concentration of False Positives in the Execution and Persistence phases suggests that the model faces difficulties in differentiating benign administrative tasks—such as legitimate background services or scheduled maintenance—from malicious persistence mechanisms when they exhibit similar system call patterns. Conversely, the False Negative rate remains extremely low across all stages, confirming that the system maintains high sensitivity and successfully minimizes the risk of missing actual attack behaviors.

Table 3. Volume of events associated with True Positives (TP), False Positives (FP), and False Negatives (FN), categorized by attack stage across Cadets and Trace datasets. The high FP count in Execution and Persistence indicates areas where benign behavior closely mimics attack patterns.

Attack Stage	**Cadets**			**Trace**		
	TP	**FP**	**FN**	**TP**	**FP**	**FN**
Discovery	8	8	70	17	500	0
Collection	0	2	0	0	0	0
Execution	12,831	40	0	8	0	0
Privilege Escalation	0	1	0	0	0	0
Defense Evasion	0	8	0	1	1026	0
Initial Access	10	217	0	0	33	0
Persistence	2	155	0	12	1529	0
Exfiltration	0	3	1	67,338	387	0
Unknown Patterns	0	11	6	0	0	790
Total Impact	12,851	445	77	67,376	3475	790

This result confirms that while the Hybrid Time–Position Embedding is highly effective at capturing the temporal context of complex attacks (especially Exfiltration and Collection), further refinement is needed to reduce false alarms triggered by ambiguous system maintenance activities in the Execution and Persistence phases.

To further investigate the model's robustness against time-variant attack strategies, we visualized the learned temporal embeddings of the detected malicious nodes using t-SNE for both Cadets and Trace datasets, as shown in Figure 4. The color gradient represents the log-scaled duration of the node's activity.



(**a**) Cadets (**b**) Trace

Figure 4. t-SNE projection of the learned temporal embeddings for detected malicious nodes in Cadets and Trace datasets (addresses RQ3). The color spectrum (from purple to yellow/green) indicates the log-scaled duration of activity. The clear separation between short-burst activities (purple) and long-duration patterns (yellow/green) demonstrates that the model effectively disentangles temporal behaviors, enabling robust detection of both immediate threats and "low-and-slow" APT attacks. In the Cadets dataset (**a**), long-duration attack patterns are distinctly clustered at the bottom of the projection, whereas in the Trace dataset (**b**), they are concentrated on the left side.

Notably, long-duration activities (yellow/green points), which correspond to "low-and-slow" APT behaviors, form distinct clusters separate from short-duration burst activities (purple points) in both datasets. This structural separation in the embedding space—appearing at the bottom for Cadets and on the left for Trace—serves as strong evidence that our proposed method successfully encodes temporal duration as a discriminative feature, preventing the model from being evaded by simple time-delay tactics.

4.8. Discussion

While the proposed Hybrid Time–Position Embedding method demonstrated slightly higher detection accuracy than the existing baseline approach, the precision, recall, and F1-score performance gain was marginal. In both datasets, the difference between the baseline and our method remained within 1–2%.

However, a more noteworthy outcome was observed during the model training process where our method significantly outperformed the baseline regarding convergence speed. For instance, in the Trace dataset, our method reduced the number of misclassified samples from 327,842 to 58,211 within only 6 refinement rounds (an 82.2% reduction), whereas the baseline remained at a high error level of approximately 140,422 even after 20 rounds. Similarly, in the Cadets dataset, our method reached a tighter stability floor of 1550 misclassified samples, which is 36.7% lower than the baseline's floor (2450). As shown in Figures 2 and 3, our model achieved convergence in fewer epochs and maintained a stable, lower misclassification rate. This quantitative evidence indicates that the proposed embedding scheme provides more informative representations, allowing the GNN to stabilize its decision boundary more efficiently.

This improvement in convergence behavior is particularly valuable in practical deployment scenarios. In provenance-based intrusion detection systems, reducing training time and computational overhead can directly contribute to better scalability and faster response. Since GNN training over large-scale provenance graphs can be computationally expensive, enhancing convergence efficiency without sacrificing detection accuracy presents a meaningful contribution.

5. Conclusions

In this paper, we proposed a novel Hybrid Time–Position Embedding technique to improve semantic representation in provenance-based intrusion detection systems. By integrating both positional encoding and relative timestamp intervals, our approach aims to capture the sequential order of events and their temporal gaps, which are critical in understanding stealthy behaviors like those found in APTs.

While our empirical results show only a slight improvement in detection accuracy compared to existing methods, we observe a significant enhancement in convergence speed during GNN training. This implies that our representation method enables the model to learn more effectively from limited iterations, reducing computational cost without degrading performance.

Our findings suggest that time-aware semantic embedding can be a practical design choice for provenance-based IDSs, particularly in environments requiring rapid model updates and efficient resource usage. Future work will explore additional temporal modeling strategies, such as incorporating edge-level temporal dynamics and attention-based time series embeddings. Furthermore, verifying the model's generalization capability against unseen or zero-day attacks remains a challenge due to the limited diversity of prepared attack scenarios in public datasets. We plan to address this by evaluating the system's robustness against novel attack patterns and exploring adversarial training techniques to further enhance detection performance.

Author Contributions: Conceptualization, S.G. and J.C.; methodology, S.G.; software, S.G.; validation, S.G.; formal analysis, S.G.; investigation, S.G.; resources, S.G. and J.C.; data curation, S.G.; writing—original draft preparation, S.G.; writing—review and editing, J.C.; visualization, S.G.; supervision, J.C. and K.K.C.; project administration, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00407295).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this study are publicly available as part of the DARPA Transparent Computing (TC) program. Specifically, the Engagement 3 (E3) dataset used for experiments, including the Cadets and Trace subsets, can be accessed from the official DARPA Transparent Computing repository at https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md (accessed on 1 March 2025).

Acknowledgments: During the preparation of this manuscript/study, the authors used ChatGPT (OpenAI, GPT-5, 2025 version) for the purposes of English language editing. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IDS	Intrusion Detection System
APT	Advanced Persistent Threat
ML	Machine Learning
DAG	Directed Acyclic Graph
UUID	Universally Unique Identifier
GNN	Graph Neural Network
TC	Transparent Computing
E3	Engagement 3 (DARPA Transparent Computing Dataset)
ReLU	Rectified Linear Unit
OS	Operating System
LCS	Longest Common Subsequence
IP	Internet Protocol
CPU	Central Processing Unit
vCPU	Virtual Central Processing Unit
RAM	Random Access Memory

Appendix A. Mapping Rules for Attack Phase Identification

To ensure the reproducibility of our experiments and clarify the definition of attack phases, we provide the detailed mapping rules used to categorize system events into MITRE ATT&CK phases. These rules are based on specific process names and system call actions observed in the dataset.

Table A1. Mapping rules for identifying attack phases based on process names and system call actions. This mapping is used to label the ground truth for phase-based analysis.

Attack Phase	Associated Processes (Exec)	System Call Actions
Initial Access	nginx, smtpd, imapd, inetd, sshd, proxymap, anvil, master, pickup, qmgr, ipop3d	EVENT_ACCEPT, EVENT_BIND, EVENT_RECVFROM, EVENT_RECVMSG
Execution	bash, sh, csh, python2.7, php-fpm, atrun, cron, expr, jot, minions, main, test, resizewin, fortune	EVENT_EXECUTE, EVENT_FORK, EVENT_MODIFY_PROCESS, EVENT_SIGNAL, EVENT_EXIT
Persistence	cron, atrun, sshd, inetd, screen, mkdir, cp, mount, nohup, tmux-1002, rc	EVENT_CREATE_OBJECT, EVENT_WRITE, EVENT_LINK, EVENT_MODIFY_FILE_ATTRIBUTES, EVENT_CLOSE
Privilege Escalation	sudo, su, pkg, doas	EVENT_CHANGE_PRINCIPAL
Defense Evasion	rm, unlink, mv, newsyslog, sleep, syslogd, chgrp, lockf, dd, chmod, chown, cleanup, history, ipfw, pfctl	EVENT_UNLINK, EVENT_RENAME, EVENT_TRUNCATE
Discovery	find, netstat, ifconfig, ps, lsof, df, ls, tail, head, cat, top, vmstat, sysctl, dmesg, uptime, hostname, route, env, kenv, less, mount, wc, cmp, date, stat, whoami, id, groups, uname, tty, limits, pwait, kldstat	EVENT_OPEN, EVENT_READ, EVENT_LSEEK, EVENT_FLOWS_TO, EVENT_FCNTL, EVENT_MMAP
Collection	cat, grep, egrep, awk, sed, sort, bzip2, xz, bzcat, dd, cp, tee, tar, zip, gzip, uniq, tr, cut, nawk, mktemp, md5	EVENT_READ, EVENT_OPEN
Exfiltration	wget, mail, sendmail, mailwrapper, alpine, msgs, sshd, curl, ftp, nc, scp, ssh	EVENT_SENDTO, EVENT_SENDMSG, EVENT_CONNECT

References

1. Kushner, D. The real story of stuxnet. *IEEE Spectr.* **2013**, *50*, 48–53. [CrossRef]
2. Haggard, S.; Lindsay, J.R. *North Korea and the Sony Hack: Exporting Instability Through Cyberspace*; JSTOR: New York, NY, USA, 2015.
3. Krasznay, C. Case Study: The Notpetya Campaign. In *Információés Kiberbiztonság*; Ludovika Egyetemi Kiadó: Budapest, Hungary, 2020; pp. 485–499.
4. Alkhadra, R.; Abuzaid, J.; AlShammari, M.; Mohammad, N. Solar winds hack: In-depth analysis and countermeasures. In *Proceedings of the 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*; IEEE: Piscataway, NJ, USA, 2021; pp. 1–7.
5. Alshamrani, A.; Myneni, S.; Chowdhary, A.; Huang, D. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 1851–1877. [CrossRef]
6. Talib, M.A.; Nasir, Q.; Nassif, A.B.; Mokhamed, T.; Ahmed, N.; Mahfood, B. APT beaconing detection: A systematic review. *Comput. Secur.* **2022**, *122*, 102875. [CrossRef]
7. Li, Z.; Cheng, X.; Sun, L.; Zhang, J.; Chen, B. A hierarchical approach for advanced persistent threat detection with attention-based graph neural networks. *Secur. Commun. Netw.* **2021**, *2021*, 9961342. [CrossRef]
8. Ahmad, R.; Alsmadi, I.; Alhamdani, W.; Tawalbeh, L. Zero-day attack detection: A systematic literature review. *Artif. Intell. Rev.* **2023**, *56*, 10733–10811. [CrossRef]
9. Ali, S.; Rehman, S.U.; Imran, A.; Adeem, G.; Iqbal, Z.; Kim, K.I. Comparative evaluation of ai-based techniques for zero-day attacks detection. *Electronics* **2022**, *11*, 3934. [CrossRef]
10. Santhosh Kumar, S.; Selvi, M.; Kannan, A. A comprehensive survey on machine learning-based intrusion detection systems for secure communication in internet of things. *Comput. Intell. Neurosci.* **2023**, *2023*, 8981988. [CrossRef]

11. Thakkar, A.; Lohiya, R. A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. *Arch. Comput. Methods Eng.* **2021**, *28*, 3211–3243. [CrossRef]
12. Bilot, T.; Jiang, B.; Li, Z.; El Madhoun, N.; Al Agha, K.; Zouaoui, A.; Pasquier, T. Sometimes Simpler is Better: A Comprehensive Analysis of State-of-the-Art Provenance-Based Intrusion Detection Systems. In *Proceedings of the 34th USENIX Security Symposium (USENIX Security 25)*; USENIX: Berkeley, CA, USA, 2025; pp. 7193–7212.
13. Zipperle, M.; Gottwalt, F.; Chang, E.; Dillon, T. Provenance-based intrusion detection systems: A survey. *ACM Comput. Surv.* **2022**, *55*, 1–36. [CrossRef]
14. Alsaheel, A.; Nan, Y.; Ma, S.; Yu, L.; Walkup, G.; Celik, Z.B.; Zhang, X.; Xu, D. {ATLAS}: A sequence-based learning approach for attack investigation. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*; USENIX: Berkeley, CA, USA, 2021; pp. 3005–3022.
15. Wu, L.; Xie, Y.; Wu, Y.; Liang, J.; Li, X. Provenance Based Intrusion Detection via Measuring Provenance Sequence Similarity. In *Proceedings of the 2022 International Conference on Blockchain Technology and Information Security (ICBCTIS)*; IEEE: Piscataway, NJ, USA, 2022; pp. 198–201.
16. Liu, M.; Xue, Z.; Xu, X.; Zhong, C.; Chen, J. Host-based intrusion detection system with system calls: Review and future trends. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–36. [CrossRef]
17. Goyal, A.; Han, X.; Wang, G.; Bates, A. Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems. In Proceedings of the 30th Network and Distributed System Security Symposium, San Diego, CA, USA, 27 February–3 March 2023.
18. Inam, M.A.; Chen, Y.; Goyal, A.; Liu, J.; Mink, J.; Michael, N.; Gaur, S.; Bates, A.; Hassan, W.U. Sok: History is a vast early warning system: Auditing the provenance of system intrusions. In *Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP)*; IEEE: Piscataway, NJ, USA, 2023; pp. 2620–2638.
19. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80. [CrossRef] [PubMed]
20. Wang, Q.; Hassan, W.U.; Li, D.; Jee, K.; Yu, X.; Zou, K.; Rhee, J.; Chen, Z.; Cheng, W.; Gunter, C.A.; et al. You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis. In Proceedings of the NDSS, San Diego, CA, USA, 23–26 February 2020.
21. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning. PMLR*; ML Research Press: Cambridge, MA, USA, 2014; pp. 1188–1196.
22. Irshad, H.; Ciocarlie, G.; Gehani, A.; Yegneswaran, V.; Lee, K.H.; Patel, J.; Jha, S.; Kwon, Y.; Xu, D.; Zhang, X. Trace: Enterprise-wide provenance tracking for real-time apt detection. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 4363–4376. [CrossRef]
23. Wang, S.; Wang, Z.; Zhou, T.; Sun, H.; Yin, X.; Han, D.; Zhang, H.; Shi, X.; Yang, J. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 3972–3987. [CrossRef]
24. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*; NIPS: Grenada, Spain, 2017; Volume 30.
25. Zeng, J.; Zhang, C.; Liang, Z. Palantír: Optimizing attack provenance with hardware-enhanced system observability. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*; ACM: New York, NY, USA, 2022; pp. 3135–3149.
26. Wu, Y.; Xie, Y.; Liao, X.; Zhou, P.; Feng, D.; Wu, L.; Li, X.; Wildani, A.; Long, D. Paradise: real-time, generalized, and distributed provenance-based intrusion detection. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 1624–1640. [CrossRef]
27. Xu, Z.; Fang, P.; Liu, C.; Xiao, X.; Wen, Y.; Meng, D. Depcomm: Graph summarization on system audit logs for attack investigation. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP)*; IEEE: Piscataway, NJ, USA, 2022; pp. 540–557.
28. Wu, L.; Xie, Y.; Li, J.; Feng, D.; Liang, J.; Wu, Y. Angus: efficient active learning strategies for provenance based intrusion detection. *Cybersecurity* **2025**, *8*, 6. [CrossRef]
29. Shen, Y.; Stringhini, G. {ATTACK2VEC}: Leveraging temporal word embeddings to understand the evolution of cyberattacks. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security 19)*; USENIX: Berkeley, CA, USA, 2019; pp. 905–921.
30. Zeng, J.; Chua, Z.L.; Chen, Y.; Ji, K.; Liang, Z.; Mao, J. WATSON: Abstracting Behaviors from Audit Logs via Aggregation of Contextual Semantics. In Proceedings of the NDSS, Online, 21–25 February 2021.
31. Van Ede, T.; Aghakhani, H.; Spahn, N.; Bortolameotti, R.; Cova, M.; Continella, A.; Van Steen, M.; Peter, A.; Kruegel, C.; Vigna, G. Deepcase: Semi-supervised contextual analysis of security events. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP)*; IEEE: Piscataway, NJ, USA, 2022; pp. 522–539.
32. Rehman, M.U.; Ahmadi, H.; Hassan, W.U. Flash: A comprehensive approach to intrusion detection via provenance graph representation learning. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*; IEEE: Piscataway, NJ, USA, 2024; pp. 3552–3570.
33. Jia, Z.; Xiong, Y.; Nan, Y.; Zhang, Y.; Zhao, J.; Wen, M. {MAGIC}: Detecting advanced persistent threats via masked graph representation learning. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24)*; USENIX: Berkeley, CA, USA, 2024; pp. 5197–5214.

34. Han, X.; Pasquier, T.; Bates, A.; Mickens, J.; Seltzer, M. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv* **2020**, arXiv:2001.01525. [CrossRef]
35. Yang, F.; Xu, J.; Xiong, C.; Li, Z.; Zhang, K. {PROGRAPHER}: An anomaly detection system based on provenance graph embedding. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23)*; USENIX: Berkeley, CA, USA, 2023; pp. 4355–4372.
36. Meng, L.; Xi, R.; Li, Z.; Zhu, H. PG-AID: An Anomaly-based Intrusion Detection Method Using Provenance Graph. In *Proceedings of the 2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*; IEEE: Piscataway, NJ, USA, 2024; pp. 2522–2527.
37. Jiang, B.; Bilot, T.; El Madhoun, N.; Al Agha, K.; Zouaoui, A.; Iqbal, S.; Han, X.; Pasquier, T. ORTHRUS: Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems. In *Proceedings of the Security Symposium (USENIX Sec'25)*; USENIX: Berkeley, CA, USA, 2025.
38. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781. [CrossRef]
39. Liu, F.; Wen, Y.; Zhang, D.; Jiang, X.; Xing, X.; Meng, D. 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*; ACM: New York, NY, USA, 2019; pp. 1777–1794.
40. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; NIPS: Grenada, Spain, 2017; Volume 30.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.