Sentient: Detecting APTs Via Capturing Indirect Dependencies and Behavioral Logic

Wenhao Yan[1,2], Ning An[1,2], Wei Qiao[1,2], Weiheng Wu[1,2], Zhigang Lu[1,2], Bo Jiang[1,2], Baoxu Liu[1,2], Junrong Liu[1,2*]

[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{yanwenhao,anning,qiaowei,wuweiheng,luzhigang,jiangbo,liubaoxu,liujunrong}@iie.ac.cn

Abstract

Advanced Persistent Threats (APTs) are difficult to detect due to their complexity and stealthiness. To mitigate such attacks, many approaches model entities and their relationship using provenance graphs to detect the stealthy and persistent characteristics of APTs. However, existing detection methods suffer from the flaws of missing indirect dependencies, noisy complex scenarios, and missing behavioral logical associations, which make it difficult to detect complex scenarios and effectively identify stealthy threats. In this paper, we propose Sentient, an APT detection method that combines pretraining and intent analysis. It employs a graph transformer to learn structural and semantic information from provenance graphs to avoid missing indirect dependencies. We mitigate scenario noise by combining global and local information. Additionally, we design an Intent Analysis Module (IAM) to associate logical relationships between behaviors. Sentient is trained solely on easily obtainable benign data to detect malicious behaviors that deviate from benign behavioral patterns. We evaluated Sentient on three widely-used datasets covering real-world attacks and simulated attacks. Notably, compared to six state-of-the-art methods, Sentient achieved an average reduction of 44% in false positive rate(FPR) for detection.

1 Introduction

Advanced Persistent Threats are notorious for their stealth and complexity. Attackers meticulously plan their strategies to gain long-term control over the target system and remain dormant, causing significant damage to modern enterprises and institutions (Alshamrani et al. 2019). For example, the SolarWinds (Office 2021) attack in 2020, which compromised multiple U.S. government agencies and major private companies, exposed sensitive information and disrupted operations on a global scale. Thus, effective detection of APTs has been a focus of both industry and academia.

The detection of APTs remains a paramount concern in cybersecurity research and practice. Leveraging audit logs to mine attack traces is currently considered the most effective approach. However, traditional Intrusion Detection Systems (IDS) struggle to cope with complex and persistent attacks, resulting in insufficient detection accuracy (Inam et al. 2023). Recent studies have analyzed valuable information in provenance graphs, effectively utilizing the rich contextual data within audit logs to provide robust support for threat detection and attribution.

Existing provenance graph-based APT detection approaches can be categorized into three classes: **Statistical-based approaches** (Wang et al. 2020; Liu et al. 2018; Hassan et al. 2019) quantify anomalies by analyzing the rarity of events in provenance graphs. However, these approaches only focus on direct event connections while ignoring the deep semantics and hidden relationships in provenance graphs. **Rule-based approaches** (Xiong et al. 2020; Hassan, Bates, and Marino 2020; Milajerdi et al. 2019a,b) rely on expert-defined rules to detect threats. However, these approaches require extensive prior domain knowledge that must be distilled by experts. **Learning-based approaches** (Han et al. 2020a; Jia et al. 2024; Yang et al. 2023; Rehman, Ahmadi, and Hassan 2024; Wu et al. 2025) employ machine learning techniques to identify anomalous behaviors and attack patterns in provenance graphs. However, attack events are often submerged in a large number of benign events, and complex dependency acquisition is limited by the receptive field of Graph Neural Networks (GNNs).

Although previous research has achieved certain effectiveness in attack detection, current methods still face several challenges in complex scenarios. We illustrate these challenges through a real attack scenario, as shown in Figure 1, where ❶ to ❻ are all labeled in the figure.

① **Missing indirect dependencies.** Two entities connected by direct interaction edges have an associated relationship, but this does not mean that non-directly connected nodes lack associations. For example, the indirect dependencies shown in ❶ and ❷ are often overlooked by models.

② **Noisy complex scenarios.** Infected entities continue to perform a large number of legitimate tasks (i.e., benign activities). Additionally, attackers may hide their attack behaviors by embedding benign substructures into their attack strategies. This causes detection models to be interfered with by weakly correlated activities, thereby producing erroneous results. For example, neighbor aggregation erroneously aggregates behaviors ❸ and ❻ that lack strong correlation, merely because they are spatially adjacent.

③ **Missing behavioral logical association.** Isolated system behaviors exhibit contextual diversity. For example,

*Corresponding author.

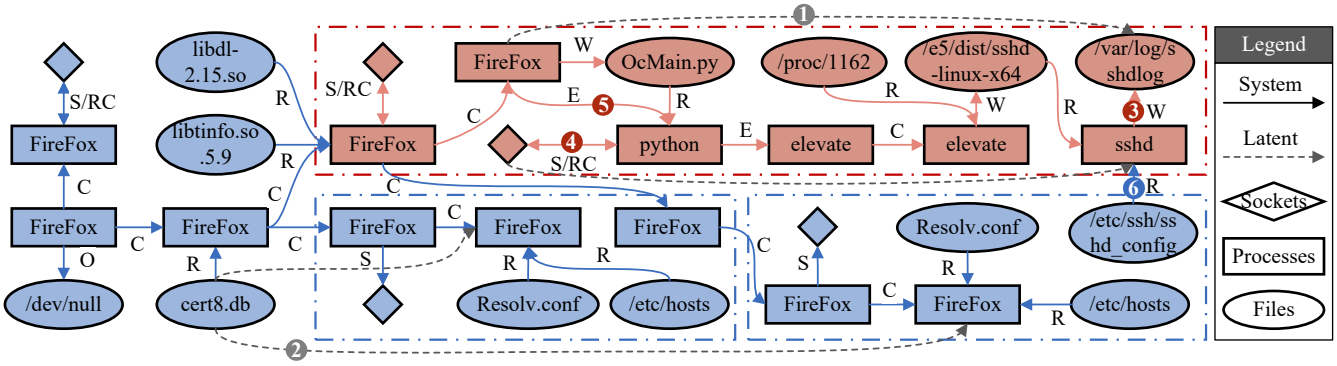


Figure 1: An example of an attack provenance graph from the DARPA E3 dataset. Red and blue denote malicious and benign, respectively. The red subgraphs highlight the core attack behavior, while the blue subgraphs represent benign DNS resolution activity. The system call type: R=Read, W=Write, O=Open, C=Clone, E=Execute, S=Send, and Rc=Receive.

when considered independently, the sshd process writing to sshdlog represented by ❹ might appear as routine logging activity under normal circumstances, while the socket interaction of the Python process represented by ❺ might be interpreted as standard network communication behavior. However, the logical association of behaviors ❹, ❺, and ❻ makes the malicious intent of the sshd and Python processes evident. However, due to the large distance between ❸ and ❹, ❺, and the influence of GNNs receptive fields, existing methods struggle to associate them.

To address the aforementioned gaps, we propose Sentient, an APT detection methodology that integrates global and local information. In contrast to existing approaches, Sentient learns established benign behavioral patterns to identify anomalous behaviors that deviate from these benign patterns, without requiring prior knowledge of attack patterns. The pre-training module leverages a graph transformer incorporating semantic and positional encodings to attend to all pertinent node information within the provenance graph, rather than merely neighboring information, thereby enabling Sentient to capture indirect dependencies. The Intent Analysis module constructs denoised scenarios for each node to mitigate information interference and captures logical associations between events through enhanced Mamba2. Additionally, Sentient simplifies the defense process by clustering behaviors with similar intent.

We implemented Sentient and compared it against six existing state-of-the-art methods on three APT attack datasets. Sentient outperformed virtually all other methods, achieving 96% accuracy and 99% recall, while reducing the average false positive rate (FPR) by 44%. We also validated Sentient's robustness against adversarial attacks.

In summary, this paper makes the following contributions.

- We propose Sentient, an accurate APT detection method for complex scenarios and stealthy attacks. It uses only easily obtainable benign data for training to learn normal behavioral intents and detect anomalous behaviors deviating from benign patterns.
- We construct a pre-training module to mine fundamental structural and semantic information in graphs, avoiding the loss of indirect dependencies.
- We combine global information to construct denoised scenarios for each node to avoid interference from noisy information.
- We design an Intent Analysis Module (IAM) to learn logical associations between behaviors.
- We conduct comprehensive evaluations using real-world datasets, and the results demonstrate the effectiveness and robustness of Sentient in detecting APTs.

2 Related Work

2.1 APT Detection

Based on the usage of audit log, current detection methods for APTs can be classified into three categories. We summarize their specific limitations in Table 1.

Model	Supervision	① Indirect Dependency	② Neighbor Denoising	③ Logical Association
Sentient	B	✓	✓	✓
Unicorn	B	✗	✗	✗
Atlas	B, A	✗	✗	✓
ProGrapher	B	✗	✗	✗
FLASH	B	✗	✗	✗
Slot	B, A	✓	✓	✗

Table 1: Comparison of APT detection approaches. Within column supervision, B indicates benign data, A refers to attack data.

Statistical-based methods, such as (Liu et al. 2018; Hassan et al. 2019; Wang et al. 2020) analyzes the distributional characteristics of system interaction behaviors and quantifies suspiciousness based on interaction frequency between system entities. These methods assume that attacks are strongly correlated with uncommon behaviors. However, rare events are not always anomalous, and this approach cannot capture deep semantic and event correlations, leading to inaccurate detection.

Rule-based methods, like (Hossain et al. 2017; Milajerdi et al. 2019b) construct attack rule databases from previous attacks and detect threats by matching audit logs against

these rules. (Xiong et al. 2020; Hossain, Sheikhi, and Sekar 2020) define rules using expert knowledge for anomaly scoring, generating alerts when scores exceed thresholds. However, these methods are limited by threat intelligence and cannot detect unknown threats, while struggling to adapt dynamically to changing network environments.

Learning-based methods model system behavioral patterns in provenance graphs to evaluate system anomalies. For example, Unicorn (Han et al. 2020a) employs graph similarity matching to detect abnormal graphs. Atlas (Alsaheel et al. 2021) applies lemmatization and word embeddings to generate sequences and uses LSTM networks to predict whether sequences are related to attacks. ProGrapher (Yang et al. 2023) uses graph embedding to construct subgraph representations and detects threats based on logical relationships between subgraphs. Threatrace (Wang et al. 2022) and Flash (Rehman, Ahmadi, and Hassan 2024) reconstruct node types to determine anomalies. Slot (Qiao et al. 2025) integrates graph reinforcement learning, dynamically aggregating neighbor information to learn correlated behaviors for threat detection. While node-level detection effectively identifies anomalies, current methods aggregate neighbor information to learn behavioral associations, which ignores indirect dependencies between non-adjacent nodes while introducing ambiguous dependencies between unrelated system behaviors.

2.2 Graph Transformer

The Transformer with attention mechanisms has achieved significant success in various fields, including Natural Language Processing (Vaswani 2017) and Computer Vision. In Graph Transformers (Chen et al. 2023), global attention is typically computed, allowing each node to attend to all other nodes, regardless of edge connections. This enables Graph Transformers to effectively capture indirect dependencies.

2.3 State Space Models

General state space models (SSMs), such as Hidden Markov Models and RNNs, process sequences by recurrently updating hidden states and generate outputs by combining hidden states with inputs. Structured State Space Models (S4) improve computational efficiency through reparameterization. Building upon S4, Mamba (Gu and Dao 2023) and Mamba2 (Dao and Gu 2024) introduce a data-dependent selection mechanism to eliminate gradient vanishing or explosion problems that occur during long sequence processing.

3 Threat Model

Building upon prior research in APT detection (Wang et al. 2022; Cheng et al. 2024; Rehman, Ahmadi, and Hassan 2024; Jia et al. 2024), our study focuses on scenarios involving stealthy and complex attack activities. During the initial intrusion phase, attackers attempt to blend malicious activities with legitimate background data to obscure their attack intentions. Similar to prior works, this model assumes the auditing system provides accurate, integrity-protected activity records. By ensuring log integrity with tamper-resistant storage, these logs serve as a reliable foundation for behavioral analysis and threat detection.

4 Methodology

As shown in Figure 2, Sentient comprises five key components: **(1) Graph Construction.** Sentient constructs provenance graphs from system logs and builds initial node representations based on semantic information and graph topological structure. **(2) Pre-training.** Sentient reconstructs node key information to comprehensively understand semantic and structural information in provenance graphs, generating ideal node embeddings that capture complex graph information. **(3) Intent Analysis Module.** Sentient combines node embeddings to construct behavioral sequences, providing a denoised environment for event association. It employs a bidirectional selection mechanism to identify logical associations between behaviors and embedded representations of behavioral intent are constructed. **(4) Threat Detection.** Anomalies are detected by verifying the similarity between reconstructed and actual edge information. **(5) Attack Investigation.** Behaviors with similar intents are clustered and attack stories are reconstructed.

4.1 Graph Construction

Sentient constructs provenance graphs from audit data collected by logging infrastructures such as Windows ETW, Linux Audit, and CamFlow (Pasquier et al. 2017). In the graph $G = (V, E)$, nodes $v \in V$ represent entities (e.g., files, processes) and edges $e \in E$ denote system interactions (e.g., write, open).

To better represent interaction scenarios and leverage rich attributes between system logs and entities, Sentient employs Word2Vec (Mikolov 2013) to project attributes (e.g., process name, file path) into low-dimensional vector representations while preserving correlations between node attributes. This semantic encoding ensures that files with similar functionalities (e.g., `/bin/vim`, `/bin/nano`) are projected into closer vector spaces, while distinct files (e.g., `/bin/vim` and `/bin/python3`) are projected into more discrete spaces.

Previous studies (Ying et al. 2021; Huang, Wang, and Li 2024) have shown that the Laplacian eigenvectors of a graph possess strong positional awareness capabilities. We apply Laplacian transformation to the adjacency matrix of the provenance graph to construct Laplacian eigenvectors, which are then used as positional representations for the nodes. The eigenvectors are derived from the graph Laplacian matrix factorization:

$$L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}, \tag{1}$$

$$Lv_i = \beta_i v_i, \quad i = 1, 2, \ldots, k+1. \tag{2}$$

where A is the adjacency matrix and D is the degree matrix. The k smallest non-trivial eigenvectors, denoted as v_i for each node i, are used as positional encodings for the nodes, with β_i representing the eigenvalue associated with v_i.

4.2 Pre-training

A complete user behaviors or attack activities typically involve multiple system calls, which manifest as multi-hop node relationships in provenance graphs. However, existing learning-based detection methods predominantly employ GNNs for neighbor message aggregation, which causes

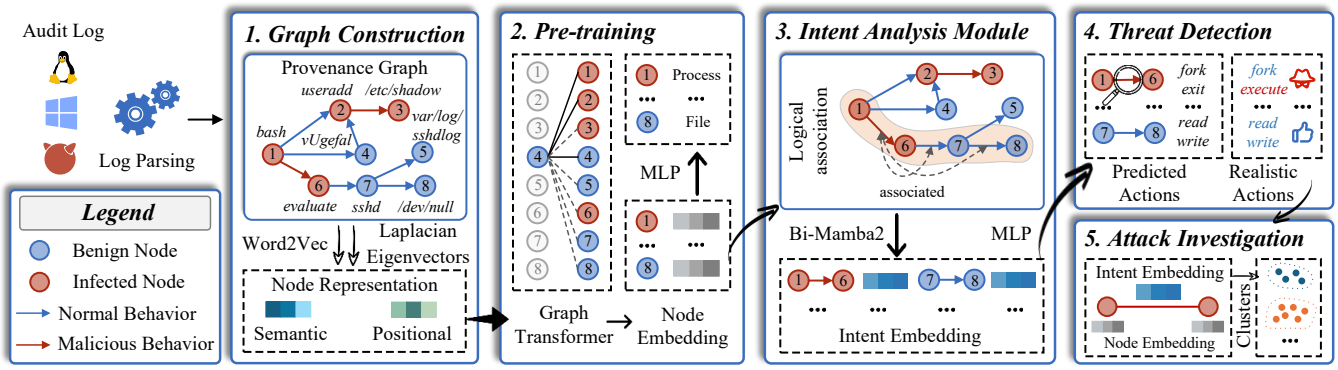


Figure 2: Overview of Sentient's Architecture.

indirect relationships between system behaviors to be overlooked. In contrast to these approaches, Sentient incorporates attention mechanisms to attend to all node information, thereby capturing indirect behavior (as shown ❶ and ❷ in Figure 1).

Specifically, Sentient combines semantic encoding α and Laplacian positional encoding β to form initial embeddings. The objective is to create a representation that incorporates both semantic information and the graph's topological structure, calculated as follows:

$$h_i^0 = \sigma((A^0\alpha + a^0) + (B^0\beta + b^0)) \,. \tag{3}$$

where A^0, a^0, B^0 and b^0 are parameters of the linear layers, and the h_i^0 is the initial representation of the node i. We use attention to mine indirect behaviors between nodes to capture complicated relationships in complete events. The node update equations for layer ℓ are as follows:

$$\hat{h}_i^{\ell+1} = O_h^\ell \Big\|_{k=1}^{H} \left(\sum_{j \in \mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^\ell \right), \tag{4}$$

$$w_{ij}^{k,\ell} = \text{softmax}_j \left(\frac{Q^{k,\ell} h_i^\ell \cdot K^{k,\ell} h_j^\ell}{\sqrt{d_k}} \right), \tag{5}$$

where $Q^{k,\ell}$, $K^{k,\ell}$, $V^{k,\ell}$ are the query, key, and value matrices for the k-th attention head at layer ℓ, H is the number of attention heads, and $\|$ denotes concatenation. The final node representation is obtained through residual connections and normalization:

$$h_i^{\ell+1} = \text{Norm}\left(\text{Norm}(h_i^\ell + \hat{h}_i^{\ell+1}) + \text{FFN}(\hat{h}_i^{\ell+1})\right), \tag{6}$$

where Norm denotes layer normalization and FFN is a feed-forward network.

Finally, the node embedding h_i captures indirect relationships between system events, which facilitates the comprehension of complete user activities. To learn user behaviors from large-scale log data and guide the generation of node embeddings that capture indirect relationships between system events, we construct a node key information reconstruction task based on node types (e.g., files, processes, sockets). Sentient employs a multi-layer perceptron (MLP) to generate probability distributions over node types: $f(h_i) = \text{MLP}(h_i)$. Where h_i represents the embedding of node i generated by Equation (7), and $f(h_i)$ denotes the probability vector for node type classification. However, node type distributions in system logs often exhibit significant imbalance. To counteract this, we employ a weighted cross-entropy loss function.

$$\begin{aligned} Loss(y, f(x)) = -\sum_{i=1}^{n} \big(& w_i y_i \log(f(x)_i) \\ & + (1 - y_i) \log(1 - f(x)_i) \big). \end{aligned} \tag{7}$$

where y represents the true node type vector, n denotes the number of classes, w_i is the weight for the i-th class, and log denotes the natural logarithm.

4.3 Intent Analysis Module

As mentioned earlier, neighbor noise interferes with the accurate capture of correlated behaviors. To address this challenge, Sentient employs a pre-trained Graph Transformer model to generate node embeddings h for provenance graphs, and utilizes random walk algorithms (Nikolentzos and Vazirgiannis 2020) to construct behavior sequences λ on the provenance graph. The i-th behavior sequence is defined as $\lambda_i = \{e_1, e_2, \ldots, e_W\}$, where W represents the maximum sequence length. Each behavior e_t in the sequence is represented as $[h_{\phi(e_t)}; h_{\psi(e_t)}]$. where $\phi(e)$ denotes the function to obtain the source node of behavior e, and $\psi(e)$ denotes the function to obtain the destination node of behavior e.

Mining logical associations between behaviors is essential for understanding behavioral intent. Sentient builds a Bidirectional Mamba2 (Bi-Mamba2) architecture to enhance bidirectional logical association capabilities on behavioral sequences. Mamba2 has shown significant effectiveness in NLP applications, particularly sentiment analysis (Qu et al. 2024), due to its ability to capture complex sequential dependencies.

Specifically, Sentient takes the behavioral sequence representation as the initial input $\lambda^{(0)}$ for Bi-Mamba2. Each layer of Bi-Mamba2 processes information in both forward and backward directions simultaneously. For the ℓ-th layer, the

computation is formulated as:

$$\lambda^{\ell+1} = \mathbf{F}(\mathbf{E}(\lambda^{\ell}) + \mathcal{R}(\mathbf{E}(\mathcal{R}(\lambda^{\ell}))), \lambda^{\ell}). \tag{8}$$

where $\mathbf{E}(\lambda^{\ell})$ is $\{y_1, \ldots, y_L\}$, $\mathcal{R}$ denotes sequence reversal operation, and $\mathbf{F}$ represents a combination function that incorporates residual connections and feed-forward neural network layers. The variable y at each layer is computed as:

$$s_t = As_{t-1} + Bx_t, \quad y_t = Cs_t + Dx_t, \tag{9}$$

where x_t denotes the input at the current time step t, s_{t-1} represents the hidden state at the previous time step. The s_t represents the derivative of the hidden state, i.e. how the state is evolving. The matrices A, B, C, and D govern state transition, input-to-state mapping, state-to-output projection, and direct feedthrough, respectively.

Sentient captures logical relationships between behaviors through the IAM and generates behavioral intent embeddings h_{e_t} for different scenarios.

4.4 Threat Detection

Subsequently, Sentient employs an MLP to reconstruct the action distribution a_t for different behaviors: $\mathbf{P}(a_t) = \text{MLP}(h_{e_t})$.

Learning Benign Patterns. Sentient mask critical behavioral information (i.e., read, write, execute operations), the model learns normal user behavioral patterns using only **benign** log. Sentient minimizes the reconstruction error (RE) between the predicted probability vector $P(a_t)$ and the observed action $L(a_t)$ from benign provenance graphs:

$$\text{RE} = \text{CrossEntropy}(\mathbf{P}(a_t), L(a_t)). \tag{10}$$

where $L(a_t)$ is a one-hot vector with probability 1 for the actual edge type of e_t and 0 elsewhere.

Anomaly Detection. In the detection phase, Sentient applies its learned understanding of benign behavioral patterns from the training phase. It uses a Graph Transformer to analyze system and lateral movement behaviors. Subsequently, IAM captures logical relationships, generating embedded representations of behaviors across various scenarios. An MLP then reconstructs these behavioral actions.

To quantify similarity, the reconstructed actions are compared to the observable actual actions by computing the RE using Equation 10. Behaviors with an RE exceeding a predefined threshold are classified as malicious, while those below the threshold are categorized as normal. The predefined threshold is based on the average RE value learned during the benign behavior learning phase. We found that using the average RE value plus 1.5 times the standard deviation yields superior detection performance.

4.5 Attack Investigation

Despite certain detection capabilities, threat detection methods inevitably produce false positives and false negatives, making security analysis heavily dependent on manual review (Alahmadi, Axon, and Martinovic 2022). The vast volume of audit logs makes it challenging for analysts to process them effectively.

Sentient constructs node embeddings capable of capturing complex scenario information through a pre-trained graph model. The IAM captures behavioral intent embeddings h_e for behavior e. Sentient concatenates these with node embeddings $h_{\phi(e)}$ and $h_{\psi(e)}$, representing the source and destination nodes of behavior e respectively:

$$\mathbf{h}_{behavior} = \text{Concat}(h_e, h_{\phi(e)}, h_{\psi(e)}). \tag{11}$$

Sentient clusters system behaviors to provide high-level behavioral representations and performs alert graph correlation. Behaviors are assigned to cluster C_k using:

$$C_k = \{e_i | \arg\min_k ||\mathbf{h}^{(i)}_{behavior} - \boldsymbol{\mu}_k||^2\}, \tag{12}$$

where $\boldsymbol{\mu}_k$ represents the centroid of cluster k. This enables merging alerts with similar behaviors, thereby alleviating the investigative burden on security analysts.

5 Evaluation

5.1 Datasets and Settings

Datasets. To comprehensively evaluate our approach, we selected three representative datasets for evaluation: **Streamspot** (E. Manzoor and Akoglu 2016): This dataset contains five benign scenarios and one malicious scenario, with 100 provenance graphs generated for each scenario. **Unicorn Wget Dataset** (Han et al. 2020b): This dataset consists of simulated attacks designed by UNICORN and includes 150 batches of logs collected by CamFlow, with 125 batches being benign and 25 batches malicious. **DARPA-E3 Dataset** (Keromytis 2018): This dataset was collected from an enterprise network during a defense exercise, where the Red Team conducted Advanced Persistent Threat attacks to exploit vulnerabilities and steal sensitive data. Since no official labels are available, we adopt the ground truth labels used by ThreaTrace and Flash for evaluation.

Baselines Methods. For comprehensive evaluation, Sentient is compared with six state-of-the-art graph-level and node-level methods. Specialized expert rules or threat reports for APT detection schemes are difficult to compare, so they are not taken into consideration.

1) Graph-level detection

Streamspot (Manzoor, Milajerdi, and Akoglu 2016): Extracts local graph features through breadth-first search and clusters snapshots to detect anomalies. **Unicorn (Han et al. 2020a)**: Analyzes system-wide data provenance graphs in real-time, employing graph sketching techniques and evolutionary models to detect abnormal behavioral patterns in APT attacks.

2) Node-level detection

Log2vec (Liu et al. 2019): Constructs heterogeneous graphs from audit logs and applies graph embedding techniques to detect abnormal activities. **Threatrace (Wang et al. 2022)**: Utilizes GraphSAGE to aggregate neighbor node information for learning structural patterns, identifying anomalies based on deviations from learned behaviors. **Flash (Rehman, Ahmadi, and Hassan 2024)**: Integrates Word2Vec semantic encoding with GraphSAGE structural

encoding to generate node embeddings for anomaly detection. **Slot (Qiao et al. 2025)**: Employs graph reinforcement learning to guide provenance graph mining with adaptive neighbor selection for anomaly detection.

Implementation. We implemented Sentient in Python 3.10 using PyTorch as the core development framework, with the torch-based DGL framework for graph learning and the Gensim library for Word2Vec. The implementation comprises approximately 3,000 lines of code. All experiments were conducted on a system running Ubuntu 22.04, equipped with an Intel(R) Core(TM) i5-12490F CPU, an NVIDIA GTX 4060Ti GPU, and 64GB of RAM. The final results shown are an average of multiple experiments.

5.2 Overall Detection Efficacy Comparsion

The Sentient is trained using a portion of the benign data from the dataset, while the remaining benign data mixed with malicious data is used for testing.

Datasets	Systems	Precision	Recall	F-score	FPR
Streamspot	Streamspot	73%	91%	81%	6.6%
	Unicorn	95%	97%	96%	1.1%
	Threatrace	*98%*	*99%*	*98%*	*0.4%*
	Sentient	**99%**	**99%**	**99%**	**0.2%**
Unicorn Wget	Unicorn	86%	95%	90%	15.5%
	Threatrace	*93%*	*98%*	*95%*	*7.4%*
	Sentient	**96%**	**99%**	**97%**	**4.1%**
DARPA E3 Cadets	Log2vec	49%	85%	62%	3.3%
	Threatrace	84%	99%	91%	0.7%
	Flash	92%	*99%*	*95%*	0.3%
	Slot	*94%*	96%	95%	*0.2%*
	Sentient	**96%**	**99%**	**97%**	**0.2%**
DARPA E3 Theia	Log2vec	62%	66%	64%	3.2%
	Threatrace	79%	99%	88%	2.1%
	Flash	91%	*99%*	95%	0.8%
	Slot	*92%*	98%	*95%*	*0.7%*
	Sentient	**95%**	**99%**	**97%**	**0.4%**
DARPA E3 Trace	Log2vec	54%	78%	64%	3.9%
	Threatrace	72%	99%	83%	2.3%
	Flash	93%	*99%*	96%	0.4%
	Slot	*94%*	98%	*96%*	*0.4%*
	Sentient	**97%**	**99%**	**98%**	**0.2%**

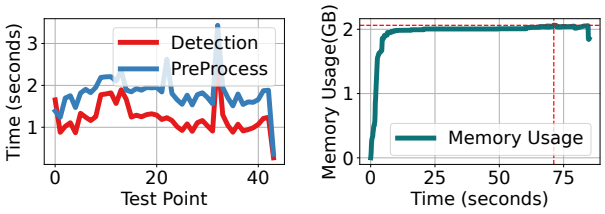
Bold indicates the best performance.
Italic underline indicates the second-best performance.

Table 2: Performance Comparison of Sentient with State-of-the-Art Methods at Graph-level and Node-level.

As shown in Table 2, we evaluated Sentient's detection efficacy at the graph-level using the Streamspot and Unicorn Wget datasets, comparing it with Streamspot, Unicorn, and Threatrace. In the Streamspot dataset, which involves relatively simple attack scenarios, Sentient easily detected anomalies and achieved near-perfect performance. On the Unicorn Wget dataset, complex scenarios compromised the detection efficacy of other methods, while Sentient maintained robust performance.

We evaluated Sentient using the DARPA E3 dataset against other node-level methods (Log2vec, Threatrace, Flash, and Slot). Sentient achieved superior results due to its capability to perceive indirect dependence in complex scenarios, reduce environmental noise, and capture logical associative relationships. In contrast, GNN-based methods (such as Threatrace, Flash, and Slot) that aggregate neighbor information fail to establish logical relationship correlations for behaviors in complex scenarios.

5.3 Performance Overhead



(a) Time Overhead (b) Memory Usage

Figure 3: Performance overhead of Sentient on the Cadets.

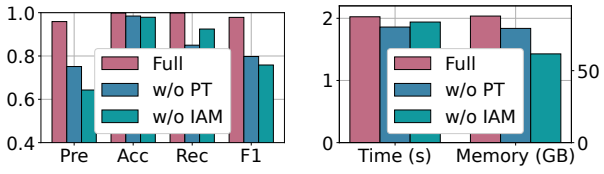
We evaluated Sentient's computational overhead using the DARPA E3 Cadet dataset, with results averaged over multiple experimental runs. Figure 3(a) demonstrates the time costs for preprocessing (raw log to provenance graph conversion) and detection phases, averaging 1.82 seconds and 1.23 seconds respectively. Figure 3(b) shows Sentient's memory consumption during detection, with peak usage of 2.01GB. Given that the E3-Cadet dataset generates approximately 2.6GB of audit logs daily over a two-week collection period, our experiments on two days of log data show that processing one day's logs (including preprocessing) requires only 63.6 seconds on average. This indicates that Sentient can meet the daily detection requirements of small organizations and enterprises with acceptable performance overhead.

5.4 Ablation Study

We conducted an ablation study of Sentient using the Cadets dataset to evaluate the importance of different components, as shown in Figure 4.

Impact of Pre-training (PT): Removing Sentient's PT component, which combines node attribute and position encoding as node embedding, caused precision to drop by 20.75%. This indicates that PT capturing complex interaction information (e.g., indirect dependencies) is crucial for node representation.

Impact of Intent Analysis Module (IAM): Removing the Bi-Mamba2 model from IAM, which directly feeds sequence representation to the MLP for action reconstruction, led to a 31.59% decrease in precision. This demonstrates that learning behavioral correlations is crucial for accurate threat detection.

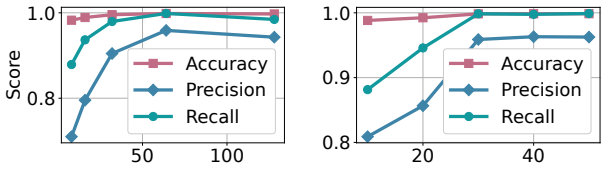


(a) Detection Performance (b) Overhead

Figure 4: Ablation Study of PT and IAM Components on CADETS Dataset: Impact on Performance and Overhead.

5.5 Hyperparameter Impact on Performance

In previous sections, we evaluated Sentient with a set of fixed hyperparameters. Here, we conduct experiments on the Cadets dataset, varying key parameters to assess their impact on Sentient's performance, as shown in Figure 5.



(a) Embedding Dimension (b) Sequence Length

Figure 5: Hyperparameter Impact on Performance.

Node Embedding Dimension: Sentient maps structural and attribute information of nodes into embeddings during the graph learning phase. The embedding dimension affects how well the embeddings represent node information. We explored different embedding dimensions, as shown in Figure 5(a). Lower dimensions reduce expressive power, negatively impacting detection, while higher dimensions lead to sparse features. We found that 64 dimensions yielded optimal performance.

Sequence Length: Sequence length determines the receptive field of events. As shown in Figure 5(b), longer sequences offer a broader receptive field but also increase performance overhead. Performance gains plateau beyond a length of 30. Considering both benefits and costs, we found 30 to be the optimal length. In practice, behavioral sequence lengths rarely exceed 30.

5.6 Robustness Against Mimicry Attacks

We evaluated Sentient's resilience against adversarial attacks using the mimicry attack method from (Goyal et al. 2023), which adds benign structures to malicious nodes to conceal malicious behaviors. Experiments on the Cadet dataset with varying numbers of added benign structures demonstrate that mimicry attacks have minimal negative impact on Sentient's performance, as shown in Table 3. This robustness stems from Sentient's ability to leverage global contextual information through pre-trained models and learn logical behavioral correlations via neighbor denoising. While mimicry attacks may deceive most methods by improving the overall impression of malicious nodes, they cannot conceal the inherent logical correlations of attack behaviors.

Adversarial Events	Precision	Recall	F-score
None	95.88%	99.85%	97.83%
1000 Events	95.39% (↓0.49%)	99.00% (↓0.85%)	97.16% (↓0.67%)
2000 Events	94.96% (↓0.92%)	98.93% (↓0.92%)	96.90% (↓0.93%)
3000 Events	95.52% (↓0.35%)	99.17% (↓0.68%)	97.31% (↓0.52%)

Table 3: Adversarial Mimicry Attack Against Our System with Varying Numbers of Added Events. Values in parentheses indicate relative changes compared to our method without adversarial data.

5.7 Alert Validation

Sentient aims to provide security analysts with effective and concise alerts to accelerate threat response. As demonstrated in previous evaluation, Sentient's superior performance in reducing false positives and false negatives significantly decreases analysts' workload. Additionally, during attack investigation, Sentient clusters behaviors with similar intents to further simplify attack chains. For instance, the port scanning trace shown in Figure 6 involves 67,374 nodes and 134,545 behaviors. After Sentient's behavioral intent clustering, analysts receive the simplified attack graph illustrated in the right of Figure 6.

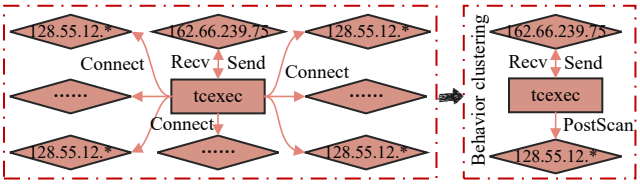


Figure 6: An example of attack activities reconstructed.

6 Conclusion

Advanced Persistent Threats pose significant cybersecurity challenges due to their stealthy and sophisticated nature. Existing detection methods fail to capture indirect dependencies and logical correlations between behaviors, resulting in high false positive rates and overlooking stealthy attacks. We propose Sentient, an APT detection method that combines graph transformers for capturing indirect dependencies with bidirectional Mamba2 for learning behavioral correlations. Sentient requires only benign data for training and detects anomalies that deviate from normal behavioral patterns. Evaluation on three widely-used datasets demonstrates superior performance, achieving a 44% average reduction in FPR compared to state-of-the-art methods.

Acknowledgments

This research is supported by National Key Research and Development Program of China (No.2023YFC2206402), the Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDA0460100), and Youth Innovation Promotion Association CAS (No.2021156). This work is also supported by the Program of Key Laboratory of Network Assessment Technology, the Chinese Academy of Sciences, Program of Beijing Key Laboratory of Network Security and Protection Technology.

References

Alahmadi, B. A.; Axon, L.; and Martinovic, I. 2022. 99% false positives: A qualitative study of {SOC} analysts' perspectives on security alarms. In *31st USENIX Security Symposium (USENIX Security 22)*, 2783–2800.

Alsaheel, A.; Nan, Y.; Ma, S.; Yu, L.; Walkup, G.; Celik, Z. B.; Zhang, X.; and Xu, D. 2021. {ATLAS}: A sequence-based learning approach for attack investigation. In *30th USENIX security symposium (USENIX security 21)*, 3005–3022.

Alshamrani, A.; Myneni, S.; Chowdhary, A.; and Huang, D. 2019. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials*, 21(2): 1851–1877.

Chen, J.; Gao, K.; Li, G.; and He, K. 2023. NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs. In *The Eleventh International Conference on Learning Representations*.

Cheng, Z.; Lv, Q.; Liang, J.; Wang, Y.; Sun, D.; Pasquier, T.; and Han, X. 2024. KAIROS: Practical intrusion detection and investigation using whole-system provenance. In *2024 IEEE Symposium on Security and Privacy (SP)*, 3533–3551. IEEE.

Dao, T.; and Gu, A. 2024. Transformers are SSMs: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, 10041–10071.

E. Manzoor, V. V., S. Momeni; and Akoglu, L. 2016. StreamSpot Code and Data. Https://sbustreamspot.github.io/.

Goyal, A.; Han, X.; Wang, G.; and Bates, A. 2023. Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems. In *30th Network and Distributed System Security Symposium*.

Gu, A.; and Dao, T. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Han, X.; Pasquier, T.; Bates, A.; Mickens, J.; and Seltzer, M. 2020a. UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats. In *Network and Distributed System Security Symposium*.

Han, X.; Pasquier, T.; Bates, A.; Mickens, J.; and Seltzer, M. 2020b. Unicorn wget Dataset. Https://dataverse.harvard.edu/dataverse/unicorn-wget.

Hassan, W. U.; Bates, A.; and Marino, D. 2020. Tactical provenance analysis for endpoint detection and response systems. In *2020 IEEE Symposium on Security and Privacy (SP)*, 1172–1189. IEEE.

Hassan, W. U.; Guo, S.; Li, D.; Chen, Z.; Jee, K.; Li, Z.; and Bates, A. 2019. Nodoze: Combatting threat alert fatigue with automated provenance triage. In *network and distributed systems security symposium*.

Hossain, M. N.; Milajerdi, S. M.; Wang, J.; Eshete, B.; Gjomemo, R.; Sekar, R.; Stoller, S.; and Venkatakrishnan, V. 2017. {SLEUTH}: Real-time attack scenario reconstruction from {COTS} audit data. In *26th USENIX Security Symposium (USENIX Security 17)*, 487–504.

Hossain, M. N.; Sheikhi, S.; and Sekar, R. 2020. Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In *2020 IEEE Symposium on Security and Privacy (SP)*, 1139–1155. IEEE.

Huang, Y.; Wang, H.; and Li, P. 2024. What Are Good Positional Encodings for Directed Graphs? *arXiv preprint arXiv:2407.20912*.

Inam, M. A.; Chen, Y.; Goyal, A.; Liu, J.; Mink, J.; Michael, N.; Gaur, S.; Bates, A.; and Hassan, W. U. 2023. Sok: History is a vast early warning system: Auditing the provenance of system intrusions. In *2023 IEEE Symposium on Security and Privacy (SP)*, 2620–2638. IEEE.

Jia, Z.; Xiong, Y.; Nan, Y.; Zhang, Y.; Zhao, J.; and Wen, M. 2024. {MAGIC}: Detecting Advanced Persistent Threats via Masked Graph Representation Learning. In *33rd USENIX Security Symposium (USENIX Security 24)*, 5197–5214.

Keromytis, A. D. 2018. Transparent Computing Engagement 3 Data Release. Https://github.com/darpa-i2o/TransparentComputing/blob/master/README-E3.md.

Liu, F.; Wen, Y.; Zhang, D.; Jiang, X.; Xing, X.; and Meng, D. 2019. Log2vec: A Heterogeneous Graph Embedding Based Approach for Detecting Cyber Threats within Enterprise. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, 1777–1794. New York, NY, USA: Association for Computing Machinery. ISBN 9781450367479.

Liu, Y.; Zhang, M.; Li, D.; Jee, K.; Li, Z.; Wu, Z.; Rhee, J.; and Mittal, P. 2018. Towards a Timely Causality Analysis for Enterprise Security. In *NDSS*.

Manzoor, E.; Milajerdi, S. M.; and Akoglu, L. 2016. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1035–1044.

Mikolov, T. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Milajerdi, S. M.; Eshete, B.; Gjomemo, R.; and Venkatakrishnan, V. 2019a. Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 1795–1812.

Milajerdi, S. M.; Gjomemo, R.; Eshete, B.; Sekar, R.; and Venkatakrishnan, V. 2019b. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE Symposium on Security and Privacy (SP)*, 1137–1152. IEEE.

Nikolentzos, G.; and Vazirgiannis, M. 2020. Random walk graph neural networks. *Advances in Neural Information Processing Systems*, 33: 16211–16222.

Office, U. G. A. 2021. SolarWinds hack. https://www.gao.gov/blog/solarwinds-cyberattack-demands-significant-federal-and-private-sector-response-infographic.

Pasquier, T.; Han, X.; Goldstein, M.; Moyer, T.; Eyers, D.; Seltzer, M.; and Bacon, J. 2017. Practical whole-system provenance capture. In *Proceedings of the 2017 Symposium on Cloud Computing*, 405–418.

Qiao, W.; Feng, Y.; Li, T.; Ma, Z.; Shen, Y.; Ma, J.; and Liu, Y. 2025. Slot: Provenance-Driven APT Detection through Graph Reinforcement Learning. In *Proceedings of the 2025 on ACM SIGSAC Conference on Computer and Communications Security*.

Qu, H.; Ning, L.; An, R.; Fan, W.; Derr, T.; Liu, H.; Xu, X.; and Li, Q. 2024. A survey of mamba. *arXiv preprint arXiv:2408.01129*.

Rehman, M. U.; Ahmadi, H.; and Hassan, W. U. 2024. FLASH: A Comprehensive Approach to Intrusion Detection via Provenance Graph Representation Learning. In *2024 IEEE Symposium on Security and Privacy (SP)*, 139–139. IEEE Computer Society.

Vaswani, A. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Wang, Q.; Hassan, W. U.; Li, D.; Jee, K.; Yu, X.; Zou, K.; Rhee, J.; Chen, Z.; Cheng, W.; Gunter, C. A.; et al. 2020. You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis. In *NDSS*.

Wang, S.; Wang, Z.; Zhou, T.; Sun, H.; Yin, X.; Han, D.; Zhang, H.; Shi, X.; and Yang, J. 2022. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Transactions on Information Forensics and Security*, 17: 3972–3987.

Wu, W.; Qiao, W.; Yan, W.; Jiang, B.; Liu, Y.; Liu, B.; Lu, Z.; and Liu, J. 2025. Brewing Vodka: Distilling Pure Knowledge for Lightweight Threat Detection in Audit Logs. In *Proceedings of the ACM on Web Conference 2025*, 2172–2182.

Xiong, C.; Zhu, T.; Dong, W.; Ruan, L.; Yang, R.; Cheng, Y.; Chen, Y.; Cheng, S.; and Chen, X. 2020. CONAN: A practical real-time APT detection system with high accuracy and efficiency. *IEEE Transactions on Dependable and Secure Computing*, 19(1): 551–565.

Yang, F.; Xu, J.; Xiong, C.; Li, Z.; and Zhang, K. 2023. {PROGRAPHER}: An Anomaly Detection System based on Provenance Graph Embedding. In *32nd USENIX Security Symposium (USENIX Security 23)*, 4355–4372.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34: 28877–28888.