Research on APT Detection Model Based on Graph Attention Network and Auto-encoder

Jiwang Sun, Hong Dai*, Baiping Sun, Xi Wei

***Abstract*—Advanced Persistent Threat (APT) detection methods often rely on labeled attack data and prior domain knowledge; however, they frequently fail to adequately capture the rich contextual information embedded within provenance graphs, thereby limiting their ability to extract complex structural patterns critical for detecting sophisticated attacks. Moreover, many existing approaches incur significant computational and memory overheads, rendering them unsuitable for large-scale or real-time deployment. To overcome these challenges, we propose GAT-AE, a novel APT detection framework that integrates a Graph Attention Network (GAT) with an auto-encoder to enhance deep feature extraction and structural representation learning from provenance graphs. By leveraging the attention mechanism, GAT-AE dynamically models the intricate associations among system entities and behaviors, significantly improving its capacity to detect stealthy and evolving APT activities. Furthermore, we introduce a dynamic masking strategy, where the masking rate is deceptively adjusted based on node centrality, enabling more effective and context-sensitive feature suppression. For the anomaly detection stage, we combine the K-Nearest Neighbors (KNN) algorithm with a K-Dimensional Tree (K-D Tree) structure to improve retrieval efficiency and detection accuracy. Extensive experiments conducted on the StreamSpot, Unicorn Wget, and DARPA E3 datasets demonstrate that GAT-AE achieves superior performance, characterized by high precision and recall rates as well as a significantly reduced false positive rate. Additionally, GAT-AE exhibits notable advantages in computational efficiency and memory utilization, highlighting its practical potential for deployment in real-world APT detection scenarios.**

***Index Terms*—Network Security, Advanced Persistent Threats, Dynamic Masking, Graph Attention Network, Auto-encoder**

I. Introduction

THE network security situation worldwide has become increasingly severe in recent years with the rapid development of network technology and the increase of information, especially APT attacks, which have become one of the significant threats affecting the security of enterprises and government organizations. However, existing APT detection methods usually rely on attack samples and a priori knowledge and are often ineffective when faced with new and unknown APT attack patterns. Therefore, designing an efficient APT detection model has important academic value and practical significance. In the field of APT attack detection, models based on the label propagation mechanism have achieved remarkable results.

For instance, the Sleuth model triggers alerts and reconstructs attack paths when behavioral violations occur, leveraging two layers of labels, one based on trustworthiness and the other on probability assigned to each node [1]. The Holmes model effectively maps potential attack chains by optimizing label aggregation and threshold computation, seamlessly integrated within the attack framework [2]. Additionally, it mitigates the impact of false dependencies through advanced noise reduction and pruning techniques. The Morse model counters the label explosion issue encountered during label propagation by implementing a label attenuation mechanism, thus preventing excessive label proliferation [3]. However, models reliant on label propagation struggle with adaptability, particularly for unknown or low-frequency attacks, and exhibit limited scalability and flexibility in complex, dynamic environments. To overcome these challenges, several studies have integrated graph features with anomaly detection methods, employing threshold-based identification of anomalous behaviors [4]. The StreamSpot [5] and Poirot [6] models address data complexity by employing graph partitioning and matching techniques. In contrast, the ATLAS model, as proposed by Alsaheel et al. [7], identifies attack behaviors through sequence-based learning, which is combined with causal graphs to derive attack paths. Meanwhile, the Log2vec model by Liu et al. [8], identifies attack behaviors through sequence-based learning, which is combined with causal graphs to derive attack paths. However, Unicorn incurs substantial computational and memory overheads, and its accuracy is highly dependent on the quality of graph summaries, which may lead to false positives or omissions [9]. Threatrace, on the other hand, precisely identifies anomalous behaviors using data provenance graphs in conjunction with a graph neural network that incorporates neighbor sampling and feature aggregation [10]. Nevertheless, Threatrace exhibits higher computational complexity, longer processing times, and a requirement for high-quality data provenance graphs.

To address these challenges, this paper presents an innovative APT detection model for deep feature extraction and structural modelling of traceability graphs in the graph representation learning phase. In particular, a dynamic

Manuscript received March, 4, 2025; revised May 10, 2025. The research work was supported by the Fundamental Research Funds for the Liaoning Universities (No. LJ212410146058) and the Graduate student science and technology innovation project of University of Science and Technology Liaoning(No. LKDYC202423).

Jiwang Sun is a postgraduate student of University of Science and Technology Liaoning, Anshan, Liaoning, CO 114051, China. (email: 1470573808@qq.com).

Hong Dai* is a professor in the School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, CO 114051, China. (corresponding author to provide phone: +086-186-4226-8599; fax: 0412-5929818; e-mail: dear_red9@163.com).

Baiping Sun is a postgraduate student of University of Science and Technology Liaoning, Anshan, Liaoning, CO 114051, China. (email: sunbaiping2023@163.com).

Xi Wei is a postgraduate student of University of Science and Technology Liaoning, Anshan, Liaoning, CO 114051, China. (email: 760861694@qq.com).

masking mechanism is designed that adjusts the masking rate through the degree of centrality of nodes to optimize feature extraction. The embedding vectors are analyzed using an outlier detection technique combining KNN and K-D Tree in the detection phase to identify potential APT attacks. Experimental results demonstrate that the model performs well on DARPA E3, StreamSpot and Unicorn Wget datasets, with significantly improved detection efficiency and significantly outperforms existing models regarding computational efficiency and memory consumption. The model not only exhibits better robustness in handling complex APT attack detection tasks but also shows excellent performance in computational efficiency, making it suitable for large-scale deployment in real applications.

II. Related Work

A. Graph Attention Network

Graph representation learning aims to map graph data into a low-dimensional vector space, where the learned node embedding effectively preserve the graph's structural and feature information [11]. Among various models, Graph Neural Networks (GNN) have emerged as a powerful approach to graph representation learning, finding widespread applications in social networks, recommendation systems, and knowledge graphs [12]. One notable GNN variant is the GAT, which introduces a self-attention mechanism to assign weights to neighboring nodes dynamically. This allows GAT to capture non-uniform relationships with greater flexibility [13]. By computing attention coefficients, the self-attention mechanism enables each node to selectively aggregate information from its neighbors based on their relative importance. The key idea is identifying highly correlated features through "non-local operations" and using these to update a node's representation. This enhances both the accuracy of local feature representations and the integration of contextual information [14]. The mapping relationship between Query and Key in the self-attention mechanism is illustrated in Fig. 1.

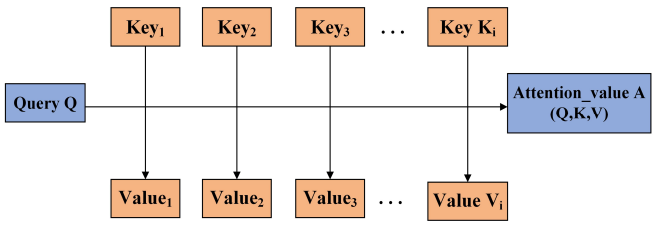

Fig. 1. Query-Key-Value mapping in self-attention

Self-attention formula, as shown in Equation (1).

$$Q = XW^{Q}, K = XW^{K}, V = XW^{v} \tag{1}$$

Where the input vector X undergoes three different sets of linear transformations to obtain the query vector (Query, Q), the key vector (Key, K) and the value vector (Value, V). These linear transformations are determined by the learnable weight matrices, W^{Q}, W^{K}, and W^{v}. The relationship between the features is measured by calculating the dot product similarity between Q and K. The results are scaled, and then the weight distribution is obtained by normalizing the similarity scores using the *Softmax* function. Finally, these weights are multiplied with V and accumulated to generate the output representation.

B. The Auto-Encoder

Auto-encoder is a typical unsupervised deep learning model, which is still essentially a multi-layer deep neural network, and the auto-encoder hopes that the output of the network can be equivalent to the input of the network, to realize the automatic learning and representation of the features of the input data samples [15]. Due to its simple structure, scalability and excellent performance, the auto-encoder is widely used in different fields such as target recognition, fault diagnosis, anomaly detection, intrusion detection and so on [16]. The core of the auto-encoder lies in minimizing the reconstruction error between input and output to learn an efficient representation of the data. The structure of the auto-encoder is shown in Fig. 2 and consists of two main parts: encoder and decoder. The encoder maps the high dimensional input to the low dimensional potential space, extracts key features, and removes redundant information. At the same time, the decoder reconstructs the data according to the low dimensional representation to make it as close as possible to the original input [17]. The whole training process minimizes the reconstruction error by optimizing the model parameters, which are commonly measured by a loss function such as the mean square error.

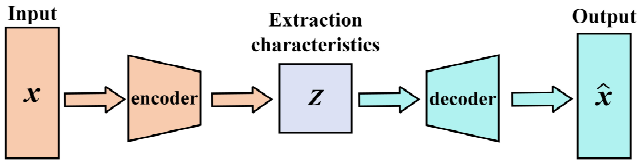

Fig. 2. Schematic diagram of the auto-encoder structure

C. Combination Mechanism of KNN and K-D Tree

Anomaly detection is widely used in several domains, such as facial expression recognition [18], network security, financial monitoring, healthcare [19], etc. The fundamental goal of anomaly detection is to identify unusual data points from a large amount of distinctive standard data that significantly deviates from the expected pattern [20]. These anomalies may represent potential system failures, network attacks, abnormal user behavior, or other unusual events. In the anomaly detection task, KNN has high computational complexity on large scale datasets because it needs to calculate the distance of each target point from all data points [21]. To improve the query efficiency of KNN, the K-D Tree is used to accelerate the distance query. K-D Tree is a binary tree structure for multidimensional data, which optimizes the nearest neighbour query of KNN by dividing the data space recursively to reduce the search range. In the K-D Tree, each node represents a region of the data space, which accelerates finding the nearest neighbour nodes by bisecting the search, thus significantly improving the query efficiency [22].

To this end, an efficient detection mechanism combining KNN and K-D Tree is proposed, which is especially suitable for real-time large-scale data analysis. In the anomaly detection task combining KNN and K-D Tree, the anomaly degree of target points can be evaluated by introducing an

anomaly scoring formula, as shown in Equation (2).

$$p = \frac{1}{K}\sum_{i=1}^{K} d(p, p_i) \tag{2}$$

Here, K is the number of predefined neighbors, and $d(p, p_i)$ is the distance between the target point p and the point p_i, which is usually calculated using the Euclidean distance.

III. Model Design

The proposed model constructs a provenance graph by extracting system entities (such as processes, files, and network connections) and their interaction relationships from audit logs. To enhance the efficiency of the graph representation, feature mapping and noise reduction techniques simplify the graph structure, minimizing redundant information while preserving critical semantic features. The resulting processed provenance graph is restructured, as illustrated in Fig. 3.

Following the graph construction, a hybrid GAT and auto-encoder framework facilitates deep-level feature extraction. GAT effectively captures global and local dependencies among nodes, while the auto-encoder refines feature representations through nonlinear transformation and reduction. A dynamic masking mechanism is also incorporated to adjust the node feature masking strategy based on data characteristics. This mechanism allows the model to intelligently emphasize critical features while mitigating the impact of noisy or less informative attributes, ultimately enhancing detection accuracy and robustness in APT attack analysis.

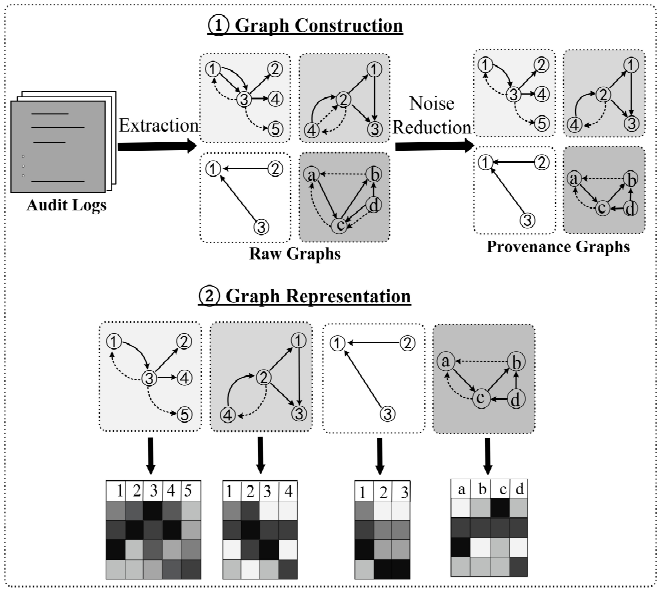

Fig. 3. Graph representation learning phases

The mechanism of combining KNN and K-D Tree is adopted for batch log level and entity level detection to accurately identify APT attacks, as shown in Fig. 4.

Meanwhile, the model introduces a highly adaptive mechanism to effectively cope with concept drift, while the innovative feedback learning and dynamic decay mechanism enable the model to seamlessly adjust to shifts in system behavior, ensuring consistently efficient detection performance. Overall, the model is not only capable of accurately identifying APT attacks but also does so with remarkable efficiency, offering excellent practicality and scalability for real-world applications.

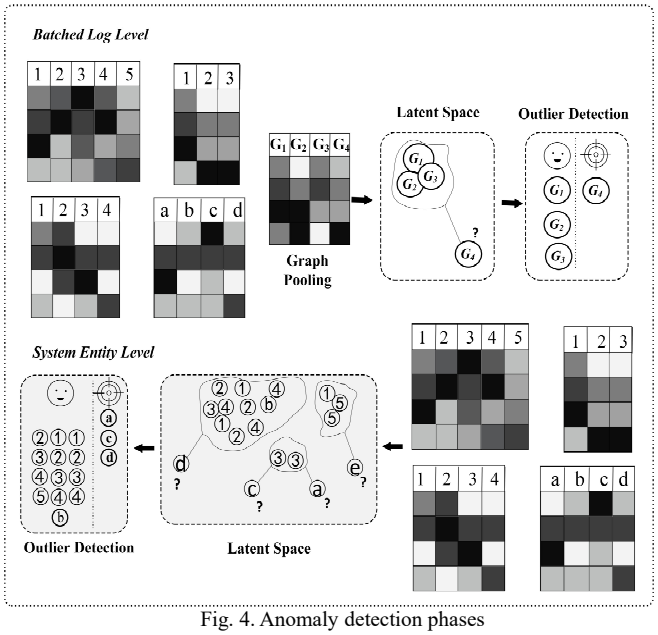

Fig. 4. Anomaly detection phases

A. Provenance Graph Construction

First, audit logs in different formats are processed using three audit log parsers, StreamSpot, CamFlow and CDM, from which system entities (processes, files, network connections) and their interactions are extracted and converted into nodes and edges to construct a provenance graph. In simple log formats, entity and interaction labels are extracted directly; in complex logs, attributes are encoded as labels using a multi-label hashing technique. Next, initial embedding is generated for the nodes and edges in the graph, and the labels are mapped to a fixed dimensional feature vector space. Nodes and edges with the same label are mapped to the same feature vector, and different labels are mapped to different feature vectors. To simplify the graph structure and reduce redundancy, redundant edges are removed, and multiple edges with the same label are merged to compute the final embedding after merging. In the construction process, practical preservation of the original semantic key information is realized, while the loss of information is almost negligible. Fig. 5 shows the provenance graph of a real-world APT attack that exploits the Pine Backdoor vulnerability. All entities and interactions unrelated to the attack have been removed from the provenance graph.

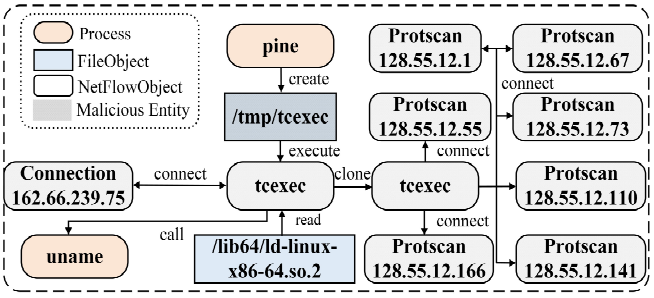

Fig. 5. A provenance graph of a real-world APT attack

B. Dynamic Masking Design

The dynamic masking mechanism forces the model to learn global and local features of the graph with incomplete information by selecting nodes for feature masking either randomly or based on graph context information. The dynamic masking mechanism designed in this paper dynamically adjusts the masking rate according to the importance of the nodes. Nodes with a high degree of centrality usually play a key role in the overall structure of the graph and information dissemination, so these nodes retain more feature information. On the contrary, nodes with lower degree centrality tend to carry more noisy information, and thus, their interference with model learning needs to be reduced by a higher mask rate. This will enable the model to focus more on the key nodes in the graph and improve the effectiveness of feature learning. The importance of a node is measured by its degree centrality C_i as shown in Equation (3).

$$C_i = \frac{\deg(i)}{|V|-1} \tag{3}$$

Here, $\deg(i)$ represents the degree of node *i, V* is the total number of nodes in the graph.

The mask rate M_i is dynamically adjusted according to the importance of the node as shown in Equation (4).

$$M_i = M_{base} - \alpha \cdot C_i \tag{4}$$

Where the base mask rate α is an adjustment factor to control the mask dynamic range. Nodes with high centrality retain more information and nodes with low centrality reduce feature noise by higher mask rate.

C. GAT-AE Model Generation

We design a model that combines GAT with an auto-encoder that contains multiple stacked layers of a graph-annotated force-meaning network. The primary role of GAT is to generate the final node embedding based on the initial features of the nodes and the features of their neighbouring nodes. Each layer of GAT receives the node embedding generated by the previous layer as input and propagates the feature information of the source node along the edges of the graph to the target through a message-passing mechanism to the target node. In this process, the delivered message not only contains the features of the source node but also carries information about the relationship between the source node and the target node. Through the self-attention mechanism, the model is able to assign different weights to different neighbouring nodes, thus better reflecting the influence of key neighbors and interactions in the node embedding.

This approach makes the generated node embedding more contextually semantic and structurally informative, providing intense expressiveness and reliability for subsequent tasks.

Calculate the attention coefficient between the message source and destination, as shown in Equation (5).

$$\alpha(src,dst) = LeakyReLU(W_{as}^{T} h_{src} + W_{am} MSG(src,dst)) \tag{5}$$

Where $\alpha(src,dst)$ denotes the attention weight from the source node to the target node, h_{src} is the feature vector of the source node for the attributes of the source node. $MSG(src,dst)$ denotes the message interaction between source and target nodes, and W_{as}^{T} and W_{am} are trainable weight matrices. The source node features and messaging information are transformed and activated using *LeakyReLU* to compute the initial attention scores. These scores are then normalized using *Softmax* to produce a distribution representing the final attention weights, as shown in Equation (6). This process enables the network to dynamically adjust the weights of the messages based on the relationships between the nodes.

$$a(src,dst) = Softmax(\alpha(src,dst)) \tag{6}$$

Then, for the target node, GAT aggregates the messages from the incoming edges by calculating the weighted sum of all incoming messages as shown in Equation (7).

$$AGG(h_{dst}, h_{\mathcal{N}}) = W_{self} h_{dst} + \sum_{i \in \mathcal{N}} a(i,dst) MSG(i,dst) \tag{7}$$

Where $AGG(h_{dst}, h_{\mathcal{N}})$ is an aggregated representation of the target node, which combines the node's own features h_{dst} and information about its neighbors $h_{\mathcal{N}}$. W_{self} is a weight matrix that adjusts the influence of the target node's own features. $\sum_{i \in \mathcal{N}} a(i,dst) MSG(i,dst)$ is an aggregation of the neighboring nodes.

Next updating its node embedding is done continuously in a multi layer network, updating the nodes layer by layer, as shown in Equation (8).

$$h_n^l = AGG^l(h_n^{l-1}, h_{\mathcal{N}_a}^{l-1}) \tag{8}$$

Where h_n^l is the hidden embedding of node *n* in GAT layer *l*, h_n^{l-1} is the hidden embedding of layer *l*-1, and N_n is the one-hop neighborhood of node *n*. By stacking multiple GAT layers, the final node embedding consists of the original node embedding and the outputs of all GAT layers in series, as shown in Equation (9).

$$h_n = emb_n \,||\, h_n^1 \,||\, \cdots \,||\, h_n^l \tag{9}$$

Where h_n denotes the final feature embedding of the target node *n*, emb_n denotes the initial embedding of node *n*, $h_n^1 \cdots h_n^l$ denotes the feature representation of node *n* in each layer of the graph neural network, and finally connects the feature vectors of the nodes in different layers in order.

From the node embedding obtained from the graph encoder, the decoder first re-masks these masked nodes and uses them as inputs for the masked feature reconstruction as shown in Equation (10).

$$h_n^* = \begin{cases} W^* h_n, n \notin \tilde{N} \\ W^* v_{remask}, n \in \tilde{N} \end{cases} \tag{10}$$

Where h_n^* is the updated features of node *n*, W^* is a weight matrix used to linearly transform the features of a node. $\tilde{N}$ is a specific set of nodes that denotes nodes that require special treatment. v_{remask} denotes the features of a node after a particular feature has been "re-masked".

Subsequently, the decoder reconstructs the initial embedding of the masked node, and the current features of the node and those of its neighbors are merged by weighting to obtain a new node representation, which is computed as shown in Equation (11).

$$x_n^* = AGG^*(h_n^*, h_{\mathcal{N}_n}^*) \tag{11}$$

Where x_n^* denotes the final embedding (or updated representation) of the target node. $AGG^*(h_n^*, h_{\mathcal{N}_n}^*)$ denotes the aggregation operation , h_n^* denotes the feature representation of a node at a particular layer, $h_{\mathcal{N}_n}^*$ denotes the feature representation of a neighboring node, which is usually the feature representation of the neighboring node in the same or previous layer.

Finally, a simple multi layer perceptron (MLP) is used to predict edge probabilities between pairs of nodes and reconstruct the loss form. In addition, instead of forcing the model to predict edge probabilities, we maximize the behavioral information contained in the embedding layers of the abstract nodes through structural reconstruction, allowing the simple MLP to convert this information into edge probabilities efficiently. Ultimately, the objective function combines the masked feature reconstruction loss and the structural reconstruction loss to help it learn the model parameters in a self-supervised manner.

D. Abnormal Detection

The core process of the detection module includes KNN , similarity calculation and outlier judgment. First, the set of neighbors of the target sample is found by the KNN algorithm, and then the average distance to the neighbors is calculated as shown in Equation (12).

$$dist_x = \frac{1}{|\mathcal{N}_x|} \sum_{x_i \in \mathcal{N}_x} \| x - x_i \| \tag{12}$$

Where $\| x - x_i \|$ denotes the distance between x and each of the neighbors of x_i , and then averaged to obtain the similarity of the sample x.

Next, the anomaly score for sample x is computed as shown in Equation (13).

$$score_x = \frac{dist_x}{\overline{dist}} \tag{13}$$

This formula is used to calculate the anomaly score of a sample. It measures the degree of abnormality of a sample by dividing the average distance $dist_x$ between x and its neighbors by the average distance $dist$ of all samples, with a higher score indicating a more likely abnormality, and determines whether or not a sample is abnormal based on the score.

Judge whether sample x is an abnormal sample based on its abnormality . If the $score_x$ exceeds the threshold θ, it is judged as abnormal (the result is 1); otherwise, it is judged as normal (the result is 0). When the target embedding is marked as an outlier, it means that the system state may deviate from the normal behavior and there is a risk of APT attack, as shown in Equation (14).

$$result_x = \begin{cases} 1, & score_x \geq \theta \\ 0, & score_x < \theta \end{cases} \tag{14}$$

In the batch log level detection, the detection module stores the benign embedding of the system state and determines whether there are outliers by calculating the average distance between the system state embedding and the stored embedding in the new provenance graph. In detecting the system entity level, the detection module stores the benign embedding of the system entity behaviors and detects the anomalies of all entity embedding in the new provenance graph. To avoid the accumulation of benign samples leading to a decrease in detection efficiency, the detection module introduces an adaptive mechanism; when the storage upper limit is exceeded, the earliest embedding is removed to make room for new samples. The method quickly adapts to the system behaviour changes and improves detection efficiency and precision.

IV. Experimentation and Evaluation

A. Datasets

The StreamSpot dataset is generated by StreamSpot using the SystemTap auditing system and contains 600 batches of audited logs covering six system call scenarios, five of which simulate normal user behavior and one of which simulates a drive-by download attack, with detailed data shown in TABLE I.

The Unicorn Wget dataset is collected by Camflow and contains 150 batches of logs, 125 batches are normal data and 25 batches contain hidden supply chain attacks. This dataset poses an important challenge in the experiments due to the large amount of data and the complex structure of the logs, and the high covert nature of the attacks. The detailed data is shown in TABLE I.

TABLE I
STREAMSPOT DATASET AND.UNICORN WGET DATASET

Dataset	Scenario	Malicious	Size(GB)
StreamSpot	CNN Download Gmail VGame YouTube		2.8
	Attack	√	
Unicorn Wget	Benign		76.6
	Attack	√	

The DARPA Engagement and Enterprise Emulation (E3) dataset originates from the DARPA Transparent Computing (TC) Program, which was designed to advance network security through transparent and comprehensive data collection. The dataset captures detailed information generated during controlled adversarial engagements conducted within enterprise-scale network environments. These simulated attack scenarios aim to mimic real-world network threats, enabling robust research into intrusion detection, threat analysis, and network defense mechanisms.

Within the broader DARPA E3 dataset, several key subsets have been developed to address different aspects of network security data. Among these, the DARPA E3 Trace,

THEIA, and CADETS datasets are particularly significant. The detailed data are shown in TABLE II.

TABLE II
DARPA E3 DATASETS

Dataset	Scenario	Malicious	Size(GB)
DARPA E3 Trace	Benign		15.40
	Extension Backdoor	√	
	Pine Backdoor	√	
	Phishing Executable	√	
DARPA E3 THEIA	Benign		17.91
	Attack	√	
DARPA E3 CADETS	Benign		18.38
	Attack	√	

B. Evaluation Metrics

To evaluate the proposed model's performance in this paper, we have chosen several key metrics, including Precision, Recall, F1-Score, and False Positive Rate (FPR). These metrics comprehensively measure the model's detection ability on different datasets from different dimensions, providing rich information for performance evaluation. These metrics are calculated based on True Positives (TP), the number of samples correctly predicted by the model to be in the positive category; True Negatives (TN), the number of samples correctly predicted by the model to be in the harmful category; False Positives (FP), the number of samples incorrectly predicted by the model to be in the positive category; and False Negatives (FN), the number of samples incorrectly predicted by the model to be in the positive category.

Precision is the proportion of positive samples among all samples predicted to be positive (malignant), reflecting the accuracy of the model in predicting the upbeat category, calculated as shown in Equation (15).

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

Recall measures the model's ability to cover positive classes, i.e., how many of all samples that are actually positive classes are correctly predicted by the model to be positive classes, and is calculated as shown in Equation (16).

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

The F1-Score combines precision and recall as a reconciled average of the two, balancing the model's predictive accuracy and coverage ability, and is particularly suited to the task of evaluating positive and negative sample imbalances, as shown in Equation (17).

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{17}$$

The False Positive Rate (FPR) is the proportion of samples that the model incorrectly predicts to be positive out of all samples that are actually in the negative category, and is calculated as shown in Equation (18).

$$FPR = \frac{FP}{FP + TN} \tag{18}$$

C. Experimental Design

In terms of experimental design, the platforms used in this paper are Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10GHz, NVIDIA GeForce RTX 4090 (24GB), and the software environments are Windows 10, Python3.8, and Cuda11.3. At the same time, we used different data partitioning strategies according to the characteristics of different datasets and used only standard samples for training. For the StreamSpot dataset, we randomly selected 400 out of 500 benign log batches for training and the remaining 100 batches for testing to ensure a balanced test set and used batch log-level detection due to the small dataset and the lack of system entities and labels for log entries. For the Unicorn Wget dataset, 100 batches of regular logs were selected for training, and the remaining portion was used for testing. The duplicate batch-log level detection was used in the experiments. For the DARPA E3 dataset, we used ground-truth labels to properly classify each network behaviour sample, such as which part is normal traffic and which part is an attack or anomalous traffic. The data is partitioned according to the chronological order of log entries, and the earliest 80% of log entries are used for training, and the remaining 20% are used as a test set, on which entity-level detection is applied. In the performance evaluation process, we used the average performance based on 100 random seeds as the final result to ensure the reliability and consistency of the experimental results.

D. Hyper parameter Setting

In our experiments, we configured the hyper parameters as follows: the scale factor (γ) in the feature reconstruction loss was set to 3, the number of nearest neighbors (k) to 10, the learning rate to 0.001, and the weight decay coefficient to 5×10^{-4}. The graph encoder, comprising three layers, was evaluated under two distinct detection scenarios: batch log-level detection and entity-level detection. For hyper parameter tuning, we explored various combinations of the embedding dimension ($d \in \{16, 32, 64, 128, 256\}$) and the number of GAT layers ($l \in \{1, 2, 3, 4\}$). The output embedding dimension (d) was selected according to the detection context: in batch log-level detection, d was set to 256 to maintain high feature expressiveness, while in entity-level detection, d was set to 64 to reduce computational overhead. The detection threshold (θ) was determined via a linear search conducted separately on each dataset. Furthermore, the dynamic masking rate was deceptively adjusted based on the centrality of each node. For batch log-level detection, θ was selected from the range [1, 10], whereas for entity-level detection, θ was determined based on dataset characteristics without the need for fine-tuning.

E. Experimental Results and Analysis

This study proposes a novel APT detection model, GAT-AE, which integrates a GAT with an auto-encoder for deep feature extraction from attack provenance graphs. By introducing a dynamic masking mechanism based on node centrality, the model enhances the focus on critical nodes, thereby improving its robustness and generalization capability against diverse APT variants. In addition, the

detection phase combines KNN and K-D Tree algorithms to optimize detection accuracy through outlier analysis. Experimental results on multiple benchmark datasets demonstrate that the GAT-AE model achieves excellent performance in maintaining high precision and low false positive rates. The detailed experimental results are presented in Table III.

TABLE III
DETECTION PERFORMANCE FOR EACH DATASET.

Dataset	Precision	F1-Score	Recall	FPR
Unicorn Wget	98.7%	97.1%	96.1%	1.6%
StreamSpot	98.6%	98.8%	99.3%	0.3%
DARPAR E3 Trace	99.5%	99.3%	99.6%	0.4%
DARPA E3 THEIA	98.8%	99.5%	99.4%	0.1%
DARPAR E3 CADETS	94.5%	97.2%	99.3%	0.1%

The proposed model integrates GAT with auto-encoder in the feature extraction phase, facilitating deep feature learning for APT attack provenance graph data. GAT captures global and local information of key nodes along the attack path. At the same time, the auto-encoder refines feature representation through nonlinear transformations, enhancing the model's ability to detect stealthy APT attacks. This innovative architecture overcomes the limitations of traditional feature engineering approaches in modeling APT attacks, enabling a more precise characterization of the complex and evolving behavioral patterns inherent in APT threats.Furthermore, a dynamic masking mechanism is designed into the model, which adjusts the masking rate based on the centrality degree of nodes within the graph. This adaptive mechanism ensures that GAT emphasizes critical nodes more during feature learning, reducing sensitivity to noisy information and ultimately improving detection performance. The dynamic masking mechanism effectively enhances precision and preserves strong generalization capabilities when confronted with diverse APT attack variants. By introducing this mechanism, the GAT-AE model maintains high detection accuracy across various APT attack scenarios, with minimal performance degradation due to variations in data distribution.

In the detection phase, the model integrates KNN with K-D Tree technology to optimize APT attack detection accuracy through outlier analysis. Traditional APT detection methods mainly rely on threshold settings or classification models. In contrast, this study adopts an unsupervised learning approach by measuring local density relationships between samples to achieve precise APT attack identification. KNN is utilized to compute local sample similarity, while K-D Tree optimizes data retrieval, enhancing computational efficiency. The model performs well on several datasets with excellent precision, recall, F1 score and false alarm rate. The precision of the Unicorn Wget and StreamSpot datasets is 98.7% and 98.6%, respectively, and the FPR for StreamSpot is only 0.3%. On the DARPA E3-series datasets, the model demonstrated high precision and low FPR on the Trace and THEIA datasets, with 99.5% precision and 0.4% FPR, and 98.8% precision and 0.1% FPR, respectively. The model maintains 94.5% precision even on the CADETS dataset, proving its stability and efficiency in a wide range of detection scenarios. To validate the effectiveness of the proposed model, this study designs a series of comparative experiments to evaluate the performance of the GAT-AE model in APT attack detection and to analyze the impact of the dynamic masking mechanism on the model's overall performance. TABLE IV presents the results of various models on the test datasets. The experimental results are systematically analyzed through comparisons with state-of-the-art models.

TABLE IV
COMPARISON BETWEEN OUR MODEL AND STATE-OF-THE-ART APT DETECTION MODELS ON DIFFERENT DATASETS

Dataset	Approach	Precision	F1-Score	Recall	FPR
Unicorn Wget	Unicorn (baseline)	86.0%	90.0%	95.0%	15.5%
	Threatrace	93.0%	95.0%	98.0%	7.4%
	Ours	98.7%	97.1%	96.1%	1.6%
Stream Spot	Unicorn (baseline)	95.0%	96.0%	93.0%	1.6%
	Threatrace	98.0%	98.0%	99.0%	0.4%
	Ours	98.6%	98.8%	99.3%	0.3%
DARPA E3 Trace	Log2vec (baseline)	54.0%	64.0%	78.0%	1.8%
	Threatrace	72.0%	83.0%	99.0%	1.1%
	Ours	99.5%	99.3%	99.6%	0.4%
DARPA E3 THEIA	Log2vec (baseline)	62.0%	64.0%	66.0%	0.3%
	Threatrace	87.0%	93.0%	99.0%	0.1%
	Ours	98.8%	99.5%	99.4%	0.1%
DARPA E3 CADETS	Log2vec (baseline)	49.0%	62.0%	85.0%	1.6%
	Threatrace	90.0%	95.0%	99.0%	0.2%
	Ours	94.5%	97.2%	99.3%	0.1%

Further experimental comparisons show that the model far outperforms the baseline model in APT detection, especially in reducing the FPR. Specific values are shown in TABLE IV. For example, in the Unicorn Wget dataset, the model's FPR is only 1.6%, much lower than Unicorn (15.5%) and Threatrace (7.4%). In the StreamSpot dataset, the model has a precision of 98.6% and an FPR of only 0.3%, significantly better than Unicorn (1.6%) and Threatrace (0.4%). In the DARPA E3 dataset, the model's FPR on the Trace and THEIA datasets is 0.4% and 0.1%, respectively, significantly outperforming Log2vec and Ttreatrace. The model exhibits high precision and low FPR on multiple datasets and good generalization ability under different APT attack patterns. Whether in high-frequency attack scenarios or hidden anomalous behavior detection tasks, the model can effectively identify potential threats, significantly reduce false alarms, and maintain stable detection performance.

In addition to improved detection precision, GAT-AE also demonstrates superior performance in computational efficiency and memory consumption. This is primarily attributed to its optimized feature extraction and detection strategies, which allow the model to operate with low

resource usage. GAT-AE exhibits strong performance in both batch log-level and entity-level detection scenarios, making it well-suited for large-scale deployment in real-world applications. Traditional APT detection methods often suffer from high computational complexity and long detection latency. However, this study optimizes the computational structure, enabling GAT-AE to improve inference speed while maintaining detection precision. As a result, the model provides excellent practical value when processing large-scale APT log data.

F. Ablative Experiments

In our comprehensive analysis, we examined the challenging Unicorn Wget dataset to investigate the impact of Feature Reconstruction (FR) and Structure Reconstruction (SR), as illustrated in Fig. 6. Through rigorous component-wise experiments, we found that FR and SR provide strong supervised signals, effectively enhancing the graph representation module.

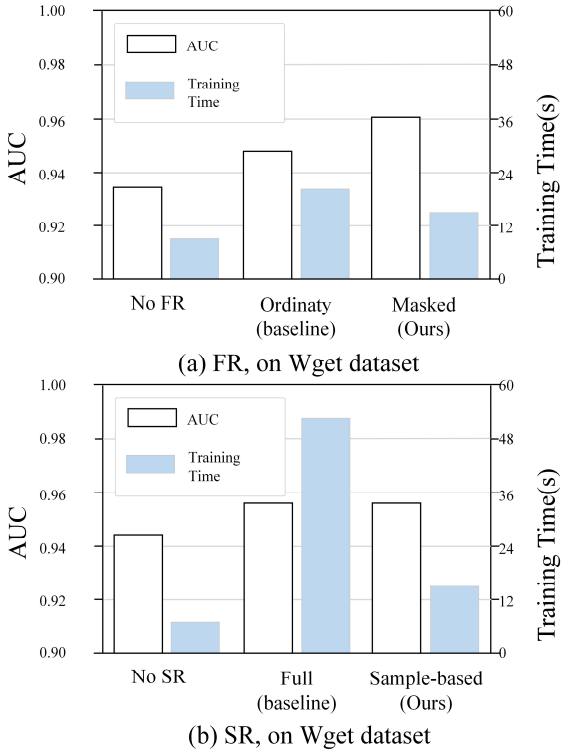

Fig. 6. Effect of reconstruction components on performance and efficiency

Notably, our findings indicate that mask-based FR yields a moderate improvement in detection performance while significantly reducing training time compared to conventional FR. Furthermore, we analyzed the differences between full SR and its sampling-based variant, demonstrating that the latter serves as an efficient complexity reduction strategy, accelerating training while maintaining performance integrity.To evaluate and benchmark the model's performance, we employed Area Under Curve (AUC) as the primary metric, where a higher AUC indicates superior discrimination between positive and negative samples. Our results confirm that the model optimizes training efficiency while simultaneously improving AUC, underscoring its advantages in both computational speed and detection accuracy.

As illustrated in Figure 7, performance steadily improves as the embedding dimension and the number of GAT layers increase. These refinements empower the model to extract more informative features, leading to superior predictive accuracy and enhanced generalization capabilities.

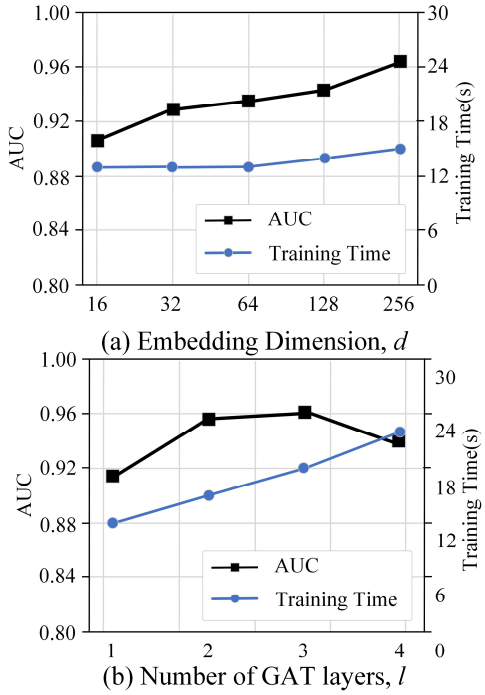

Fig. 7. Effect of different hyper parameters on performance and efficiency

In our comprehensive hyper parameter analysis, we identified two critical factors that substantially impact model performance: the embedding dimension d and the number of GAT layers *l*. Our findings indicate that increasing the embedding dimension leads to a consistent improvement in AUC, highlighting the model's enhanced capacity to capture intricate data structures. This expanded representational capability facilitates more accurate predictions and overall performance gains.

However, this improvement comes at the cost of increased computational complexity. A higher embedding dimension extends training time, necessitating a careful balance between performance enhancement and computational efficiency. Selecting an optimal embedding size is thus crucial to maximizing model effectiveness while maintaining feasible resource utilization.

Moreover, while most hyper parameters exert minimal influence on model performance, adjustments to the embedding dimension and the number of GAT layers significantly enhance the model's receptive field. Increasing the number of GAT layers enables the model to capture deeper relational dependencies across multiple graph levels, thereby improving its ability to learn meaningful structural patterns.

V. Conclusions And Future Work

In this paper, an efficient APT detection model is proposed, which combines graph attention network and auto-encoder, and introduces a dynamic masking mechanism to significantly improve the precision and efficiency of APT attack detection. In this model, GAT is

used to capture the complex relationships among nodes in graph-structured data and assign higher attention weights to important nodes, thus effectively improving the modeling ability of attack paths. Auto-encoder, on the other hand, helps extract potential attack features through unsupervised feature learning and data reconstruction, and performs conditionality reduction on the data to reduce the interference of redundant information. The introduction of the dynamic masking mechanism enables the model to automatically adjust the feature weights according to the actual distribution of the data during the training process, which further enhances the robustness to unbalanced datasets or scenarios with scarce labeled data. Overall, the experimental results validate the superior performance of the proposed model in APT detection, especially in data-scarce or complex environments, which significantly outperforms the baseline model. The excellent performance on all datasets demonstrates that the model is able to balance detection precision, resource consumption, and environmental adaptability, showing a wide range of application potential and practical value.

Future research can further enhance the applicability of the model by implementing targeted optimizations for different types of APT attack features, thereby improving its generalization ability and real-time performance across diverse scenarios. In addition to its current capability in APT attack detection, the model can be further refined to effectively address increasingly complex attack patterns, such as industry-specific targeted attacks, zero-day exploits leveraging emerging vulnerabilities, and sophisticated multi-stage attack strategies. To achieve this, future studies could explore advanced optimization techniques, including enhancing feature extraction mechanisms, integrating more refined graph-based network modeling approaches, and leveraging augmented learning to enable the model to adapt to a broader spectrum of attack characteristics.

Furthermore, as APT attack methodologies continue to evolve and real-time detection demands become increasingly stringent, future research should prioritize improving the model's responsiveness to novel attack variants. Strengthening its capacity for rapid threat identification and adaptive defense mechanisms will be essential to ensuring timely and precise threat mitigation. By continuously refining its adaptability and computational efficiency, the model can achieve superior real-time performance, making it a more robust solution in the dynamic landscape .

REFERENCES

[1] N. Hossain, M. Milajerdi, and J. Wang, "SLEUTH: Real-time attack scenario reconstruction from COTS audit data," Proceedings of the 26th USENIX Security Symposium (USENIX Security '17), pp487-504, 2017.

[2] M. Milajerdi, R. Gjomemo, and B. Eshete, "Holmes: Real-time APT Detection Through Correlation of Suspicious Information Flows," 2019 IEEE Symposium on Security and Privacy (SP), IEEE, pp1137-1152, 2019.

[3] N. Hossain, S. Sheikhi, and R. Sekar, "Combating Dependence Explosion in Forensic Analysis Using Alternative Tag Propagation Semantics," 2020 IEEE Symposium on Security and Privacy (SP), IEEE, pp1139-1155, 2020.

[4] C. Dong, M. Liu, and T. Chen, "Heterogeneous Provenance Graph Learning Model Based APT Detection," Computer Science, vol. 50, no. 4, pp359-368, 2023.

[5] E. Manzoor, S. Milajerdi, and L. Akoglu, "Fast Memory-efficient Anomaly Detection in Streaming Heterogeneous Graphs," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp1035-1044, 2016.

[6] M. Milajerdi, B. Eshete, and R. Gjomemo, "Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting," Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp1795-1812, 2019.

[7] A. Alsaheel, Y. Nan, and S. Ma, "A Sequence-based Learning Approach for Attack Investigation," Proceedings of the 30th Security Symposium, pp100-110, 2021.

[8] F. Liu, Y. Wen, and D. Zhang, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within the enterprise," Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp1777-1794, 2019.

[9] X. Han, T. Pasquier, and A. Bates, "Unicorn: Runtime provenance-based detector for advanced persistent threats," Cryptography and Security, 2020.

[10] S. Wang and Z. Wang, "THREATRACE: Detecting and Tracing Host-Based Threats in Node Level Through Provenance Graph Learning," IEEE Transactions on Information Forensics and Security, vol. 17, pp3972-3987, 2022.

[11] F. Chen, C. Wang, and B. Wang, "Graph representation learning: a survey," APSIPA Transactions on Signal and Information Processing, vol. 14, pp1143-1168, 2020.

[12] Z. Baoxin, Y. Dan, L. Yang, and Z. Yu , "Graph Contrastive Learning with Knowledge Transfer for Recommendation," Engineering Letters, vol. 32, no. 3, pp477-487, 2024.

[13] Z. Wu, S. Pan, and F. Chen, "A comprehensive survey on graph neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 1, pp4-24, 2020.

[14] X. Wang, H. Ji, and C. Shi, "Heterogeneous graph attention network," The World Wide Web Conference, pp2022-2032, 2019.

[15] H. Shao, H. Jiang, and H. Zhao, "A novel deep autoencoder feature learning method for rotating machinery fault diagnosis," Mechanical Systems and Signal Processing, vol. 95, pp187-204, 2017.

[16] F. Farahnakian, and J. Heikkonen, "A deep auto-encoder based approach for an intrusion detection system," Proceedings of the 2018 20th International Conference on Advanced Communication Technology (ICACT), IEEE, pp178-183, 2018.

[17] Y. Xiao, Y. Feng, and K. Sakurai, "An Efficient Detection Mechanism of Network Intrusions in IoT Environments Using Autoencoder and Data Partitioning," Computers, vol. 13, no. 10, pp269-274, 2024.

[18] K. Hong, "Facial Expression Recognition Based on Anomaly Detection and Multispectral Imaging," IAENG International Journal of Computer Science, vol. 51, no. 10, pp1627-1641, 2024.

[19] J. Bin, W. Feng, C. Rong, and W. Jian, "Lightweight Privacy-Preserving Anomaly Detection for Time Series Based on Federated Learning," Engineering Letters, vol. 33, no. 2, pp275-281, 2025.

[20] X. Guo, C. Dong, and H. Liang, "Research on Network Intrusion Detection Technology Based on Artificial Intelligence," China Information Journal, no. 03, pp130-132, 2024.

[21] H. Abdalla, A. Altaf, and A. Hamzah, "A Threefold-Ensemble K-Nearest Neighbor Algorithm," International Journal of Computers and Applications, vol. 47, no. 1, pp70-83, 2025.

[22] X. Song, S. Jin, and T. Han, "Research on Weighted KNN Combined with K-D Tree Algorithm in Medical Image Classification," Electronic Components and Information Technology, vol. 8, no. 08, pp122-125, 2024.