MPKAN: APT Attack Detection on Audit Logs via Graph Semantic Enhancement

Zehui Wang[†‡], Dan Du[†‡], Yinhao Qi[†‡], Wenhao Yan[†‡], Xiaobo Yang*[†‡], Bo Jiang[†‡], Zhigang Lu[†‡]
[†]Institute of Information Engineering,Chinese Academy of Sciences, Beijing, China
[‡]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{wangzehui, dudan, qiyinhao, yanwenhao, yangxiaobo, jiangbo, luzhigang}@iie.ac.cn

2025 28th International Conference on Computer Supported Cooperative Work in Design (CSCWD) | 979-8-3315-1305-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/CSCWD64889.2025.11033314

***Abstract*—As cloud computing and mobile work blur traditional network boundaries, security measures like static firewalls and signature-based systems are becoming inadequate. Audit logs contain fine-grained OS-level information, but due to their vast volume and the complex relationships between entities, processing and analyzing them remains a significant challenge.**

In this paper, we introduce MPKAN, a new method for detecting APT attacks, which enhances the information input of nodes and edges, integrates node-level and edge-level information, and improves graph-level semantics. It uses meta-path random walks to enhance semantic connections between nodes, merges multiple edges in the provenance graph into a single edge while retaining the original edge information and operation sequence relationships, and by associating heterogeneous graph neighbors, utilizing the message passing mechanism to iteratively update states based on neighbor node information, and using a knowledge association network to integrate node-level and edge-level information, we can effectively capture local and global structural information in the graph. MPKAN's evaluations on the ATLAS and Darpa datasets demonstrate its excellent performance in complex attack scenarios, achieving an average accuracy of 0.9899 and an F1 score of 0.9853, confirming its effectiveness and efficiency.

***Index Terms*—APT attack detection, Audit logs, meta-paths.**

I. Introduction

As network attack and defense technologies continue to evolve, Advanced Persistent Threats (APTs) have become a significant threat in the modern cyberspace [1] [2]. APT attacks are characterized by their stealth and persistence, as attackers conduct thorough reconnaissance and intelligence gathering on their targets in the early stages. They employ sophisticated technical strategies and social engineering tactics, including but not limited to zero-day [3] exploitations, custom malware, and advanced stealth techniques, to penetrate the cybersecurity defenses of target organizations.

Traditional network perimeter defense methods are no longer effective. Conventional network perimeter defense techniques, such as static firewall rules, signature-based intrusion detection systems, and perimeter defense devices, have gradually shown their limitations [4]. Additionally, with the rise of cloud computing and mobile workforces, network boundaries have become increasingly blurred, rendering traditional network boundary-based security strategies inadequate for adapting to this dynamic environment. Therefore, conducting behavioral analysis at a deeper data level enables the identification of more potential threats.

Using audit logs for fine-grained behavioral analysis can uncover more potential attacks [5] [6] [7]. Compared to other types of logs, audit logs record information at the operating system level and typically include key details such as timestamps, user identities, actions performed, and the outcomes of those actions. This provides a solid foundation for post-incident investigations and forensics. Therefore, using audit logs in attack investigations allows for more effective detection of abnormal behavior, understanding of attackers' methods, and reconstruction of the complete attack chain.

Current research methods overlook the complex behaviors between entities. To explore the relationships between entities in audit logs, including processes, files, and networks—most researchers choose to construct provenance graphs (PG) to represent these connections. Since audit logs contain numerous basic operations, such as open and read, researchers often simplify the graph construction by retaining only the first occurrence of an operation event as the edge to reduce the complexity of the edges. While this approach simplifies the graph construction, it results in a significant loss of semantics.

In this paper, to address the issues mentioned above, we propose MPKAN, a detection method specifically designed for APT attacks. Our method strengthens semantic connections when learning node features by using meta-path random walks as the node traversal method, deepening the association of nodes of the same type in the heterogeneous graph. It enhances the semantic information of edges in the provenance graph by retaining multiple operation events between two entities and organizing them into an operation sequence. This enriches the attributes of the edges without increasing the number of edges. And MPKAN enhances the capture of complex behaviors in graph structures by passing messages among neighbor nodes to iteratively update node states, and by integrating knowledge to effectively associate node and edge information from multiple dimensions, thereby further improving detection accuracy. The main contributions of this paper are as follows:

- We propose a new APT attack detection method that focuses on the operation sequence between entities in the provenance graph. This method models by integrating node and edge information from heterogeneous graphs.
- This method uses message passing mechanisms and knowledge fusion, It can captures local features in the graph effectively.
- We evaluate MPKAN on five datasets, demonstrating high

accuracy and F1 scores. The ATLAS dataset includes four attack datasets, with an average accuracy of 1.000 and F1 score of 0.9840. The DARPA dataset includes multiple attack scenarios, with an average accuracy of 0.9899 and F1 score of 0.9853.

II. Background & Motivation

A. Related Work

Some existing research converts audit logs into graph form for analysis to effectively discover hidden patterns and detect security threats [8] [9] [5]. In these methods, information from audit logs, such as user activities, system events, and network interactions, is represented as nodes and edges in a graph [10] [11]. Nodes represent entities like users, files, and processes, while edges represent interactions or events between these entities [5] [8]. For example, Watson [12] utilizes knowledge graphs, extracting sequences of audit events into triplets that form a sequence. These sequences serve as behavior examples, using graph embedding techniques for behavior summarization and semantic aggregation. Researchers can apply advanced graph theory algorithms and machine learning techniques to more efficiently identify anomalies and potential threats.

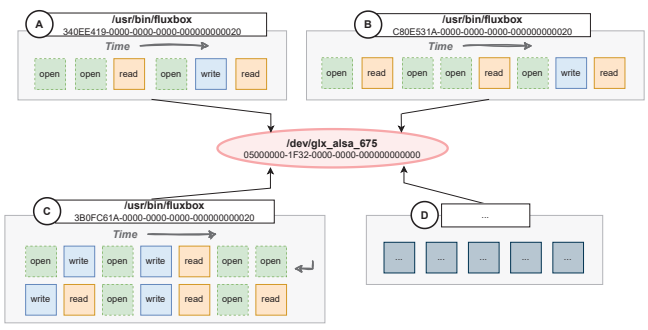
Some methods use sequence learning to capture complex relationships in audit logs, aiming to identify behavior patterns in the system and potential security threats. These methods often employ advanced sequence analysis techniques, such as deep learning. For instance, Airtag [13] treats log texts as sequence data and utilizes unsupervised learning techniques to directly train deep learning models on these texts. This approach avoids the time-consuming and error-prone process of causal graph generation and manual data labeling found in traditional methods. Sequence learning improves detection efficiency and enhances the ability to predict attack behaviors by extracting key temporal patterns from complex data.

B. Motivation

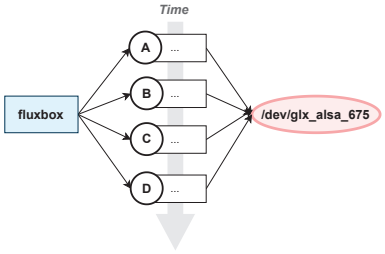
The provenance graph is not a fully connected graph; it consists of multiple subgraphs. The entities within the provenance graph involve numerous basic operations, which bring rich semantics, but the vast amount of data also presents challenges for research and analysis.

Different processes may have similar functions. For example, in the association between processes and files, as illustrated in Fig. 1(a), the file object $/dev/glx\_alsa\_675$ is accessed by multiple processes *A*, *B*, *C*, and *D* in sequence. These processes are all generated by *fluxbox* and represent *fluxbox* process entities in different states. It is evident that these processes are strongly related, indicating that similar operations were performed at different times.

The edges carry rich semantics. Each of these operations, as shown in Fig. 1(b), such as open, read, and write, constitutes an edge in the provenance graph. These sequential basic operations, such as open (a necessary prerequisite for read and write), read, and write, collectively form several specific operations.



(a)



(b)

Fig. 1: Motivation Example

III. METHODOLOGY

As shown in Fig. 2, MPKAN consists of four main modules: graph construction, node feature extraction, edge feature extraction, and the MPKAN model.

A. Graph Construction

TABLE I: Element of Graph Construction

Src	Dst	Connection Relationship
Process	Process	Clone/Fork
	File	Read, Open, Execute, Modify_file_attributes
	Netflow	Connect, Unlink, Send(Sendto, Sendmsg), Recv(Recvfrom, Recvmsg)

We define the entitie and edge types as shown in Table I. In the DARPA dataset, we unify the edge types of *sendto* and *sendmsg* as *send*, and *recvfrom* and *recvmsg* as *recv*. For structural consistency, we unify the *sock_send*, *connected_session*, *connect*, and *connected_remote_ip* in the ATLAS dataset as *connect*, and *execute*, and *executed* as *exec*.

B. Node Feature Extraction

In heterogeneous graphs, nodes include processes, networks, files, and edges represent operations between these entities. Due to the diversity of node types, it is necessary to guide the random walk process with specific path patterns to effectively capture semantic relationships between nodes. MPKAN defines meta-path-guided random walks to obtain node sequences and uses Word2Vec [14] to extract node features.

Meta-Path Extraction. To make the relationships between edges of different categories clearer, we use meta-path random walks as the method for path extraction [15]. By guiding

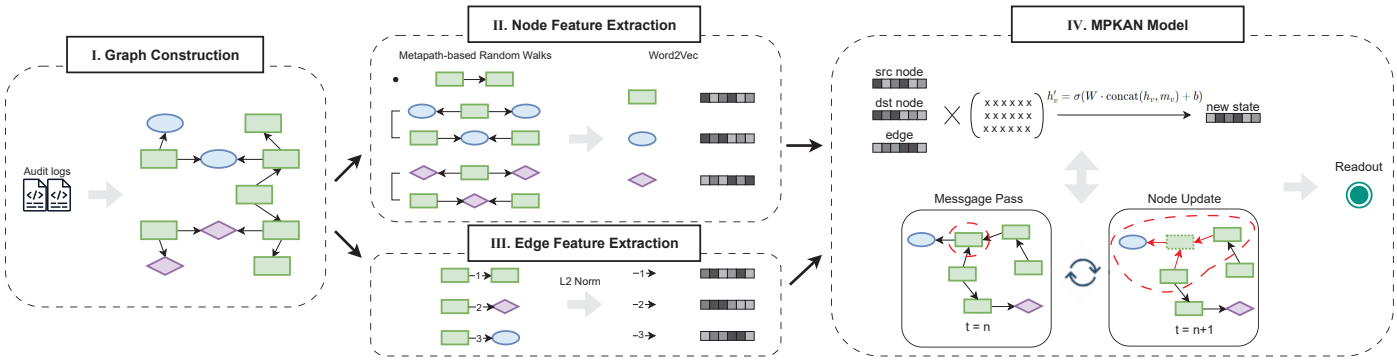


Fig. 2: Overview of MPKAN

The equations $h'_v = \sigma(W \cdot \text{concat}(h_v, m_v) + b)$ in Section IV expresses the procedure of updating a node in MPKAN by integrating information from its neighbors, where σ is the activation function, W is the weight matrix, and b is the bias, h'_v is updated to new embeddings that combine its own features with those of its neighbors from h_v, m_v represents the message received by node v from other nodes.

TABLE II: Meta-Path Definition

Metapath	Description
process→process	A process calls another process
process→file←process	Two processes operate one file
file←process→file	Two files are operated by one process
process→network←process	Two processes connect to one network
network←process→network	Two networks are connected by one process

the walks according to predefined meta-paths, we can focus particularly on important, semantically related nodes, thereby helping the model understand the relationships between behaviors and entities. This approach also better preserves the type characteristics of nodes and the topological structure of the network, making it more suitable for subsequent feature representation. We define five categories of meta-paths, as shown in Table II.

We establish a bidirectional graph to ensure connectivity between nodes. We filter out the neighbor directions that match the current relationship type for traversal and remove the paths that have already been traversed to ensure the validity of the traversal sequence.

Word2Vec. We choose Word2Vec to generate node embeddings. After extracting random sequences by using meta-paths, we use Skip-gram as the algorithm for generating node embeddings. The objective of the Skip-gram model is to predict the context from the target word. For a given sequence of words, the model takes each word as input and predicts its surrounding context words. The Skip-gram model can be represented by the following probabilistic model, where the goal is to maximize the following function:

$$L(\theta) = \prod_{w \in C} \prod_{w_i \in w} \prod_{-m \le j \le m, j \ne 0} P(w_{i+j} | w_i; \theta) \qquad (1)$$

where, C is the collection of all documents in the training corpus, w is one document in C, w_i is the target word in document w, w_{i+j} is the context word of the target word, m is the size of the context window, and θ is the model parameter.

Each conditional probability $P(w_o|w_i;\theta)$ can be calculated using the softmax function as follows:

$$P(w_o|w_i;\theta) = \frac{\exp({v'_{w_o}}^{\top} v_{w_i})}{\sum_{w=1}^{W} \exp({v'_w}^{\top} v_{w_i})} \qquad (2)$$

where v_w and v'_w are the input and output vector representations of word w, and W is the size of the vocabulary.

By extracting sequences of defined types through meta-path random walks and using Word2Vec to learn embeddings of associated nodes, the representation of nodes in the contextual semantics can be enhanced, while also deepening the features of similar nodes.

C. Edge Feature Extraction

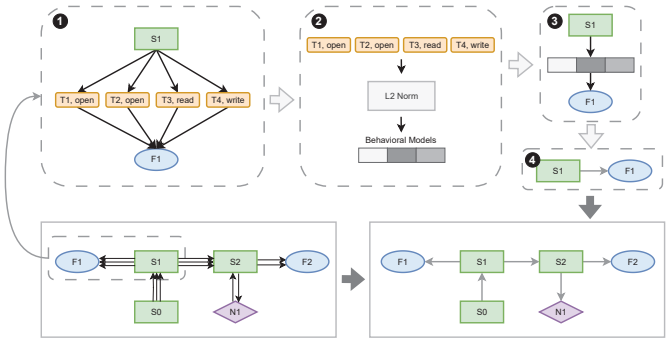


Fig. 3: Workflow of Merging Multiple Edges

In the provenance graph, each operation represents an edge, as shown in Fig. 1(b), and there are multiple fine-grained operations between different entities. These sequential fine-grained operations carry rich semantics but can also lead to an explosion in the graph's complexity. To reduce the overall complexity of the graph, we need to trim and compress the edges. In this paper, we sort the operations between two entities by time, as illustrated in Fig. 3. By merging the

information from multiple edges between entities in the initial provenance graph, a new compressed edge is generated, which can better represent the relationship between the two entities.

We capture the first 32 operations between two entities and arrange them in event order. If there are fewer than 32 operations, we pad the sequence with zeros at the end. We apply L2 normalization to the resulting operation sequence. L2 normalization scales the vector by dividing it by its L2 norm (i.e., its Euclidean length), making the vector's length equal to 1. The formula for L2 normalization is as follows:

For the vector $x = [x_1, x_2, \ldots, x_n]$, its L2-normalized vector x' is calculated as follows:

$$x' = \frac{x}{\|x\|_2} \tag{3}$$

where $\|x\|_2$ is the L2 norm of x.

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{4}$$

Through the above operations, most of the interaction information between entities can be preserved. Considering the repetition of many operations between entities, only the first 32 operations are retained. By applying L2 normalization, the generated edge feature sequences can interact more effectively with node features.

D. MPKAN

To capture complex dependencies between nodes and generate high-quality node embeddings, we draw inspiration from the message passing mechanism in MPNN [16]. MPKAN iterates multiple rounds of message passing to gradually propagate and aggregate information from neighboring nodes. Through the process of node feature extraction described earlier, different message passing rules can be used to handle different types of nodes and edges, thereby better capturing the complex relationships in heterogeneous graphs. Considering that edge-level information is also crucial in the graph, MPKAN uses both node and edge information as input to build a more comprehensive and multi-dimensional node representation, leveraging the associative relationships provided by edges to enhance the graph's representation capabilities. Algorithm 1 illustrates the entire computation workflow of MPKAN.

where the specific computation process for the src node , dst node, and the edge in the update phase are

$$\begin{cases} h'_u = \sigma(W_{\text{src}} \cdot h_u^{(t)} + b_{\text{src}}) \\ h'_v = \sigma(W_{\text{dst}} \cdot h_v^{(t)} + b_{\text{dst}}) \\ z'_{uv} = \sigma(W_{\text{edge}} \cdot z_{uv} + b_{\text{edge}}) \end{cases} \tag{5}$$

The computation process of feature combination is

$$h_{\text{combined}} = \text{Concatenate}(h'_u, h'_v, z'_{uv}) \tag{6}$$

and the computation process of node embedding update is

$$h_v^{(t+1)} = \sigma(W_{\text{comb}} \cdot h_{\text{combined}} + b_{\text{comb}}) \tag{7}$$

where $W_{src}, W_{dst}, W_{edge}$ are the transformation matrices for the source node, destination node, and edge. And $b_{src}, b_{dst}, b_{edge}$ are the bias terms for the respective transformations. And σ is activation function, here is ReLU. W_{comb} is the transformation matrix for combining features, and b_{comb} is the bias term for the combination.

Algorithm 1 MPKAN Model

Require: $G(V, E)$
Ensure: Y_e
function MPKAN(V, E)
  Initialize $H^{(0)}$ from V
  Initialize Z from E
for each epoch **do**
  for each edge(u, v) in E **do**
    $src\_embeddings \leftarrow H^{(0)}[u]$
    $dst\_embeddings \leftarrow H^{(0)}[v]$
    $edge\_embeddings \leftarrow Z[(u, v)]$
    $transformed \leftarrow KAN(embeddings)$
    $concatenated \leftarrow Concatenate(src, dst, edge)$
    $H^{(1)}[v] \leftarrow MessagePassing(concatenated^{(0)})$
  end for
  $H^{(1)} \leftarrow dropout(H^{(1)})$
end for
Readout
  $Y_e \leftarrow classifier(H^{(n)})$

Message Passing Mechanism. The message passing mechanism typically involves three main steps: message generation, message aggregation, and node state update. These steps are repeated in each training iteration, with each round based on the current state of all nodes in the graph. MPKAN learns local context information through message passing mechanisms and understands local structures and global patterns in the graph by the information passing process between neighboring nodes.

Knowledge Association. Knowledge Association [17] is capable of flexibly handling complex graph structures, overcoming some of the limitations of traditional MPNN. In MPNN, both the Message Function and Update Function phases utilize MLP (Multilayer Perceptron). MLP is a feedforward neural network that consists of multiple fully connected layers. It can process various types of data, but its method of processing information is often straightforward, with each layer's output depending solely on the output of the previous layer. Therefore, MLP may struggle to effectively capture the complex topological structures and relationships within a graph, as it lacks an inherent mechanism for handling graph-structured data.

MPKAN is optimized for the characteristics of graph data. It independently processes the information from each component (sou node, dst node, and edge) and combines them before the final output, allowing the network to capture complex information from multiple aspects.

IV. EVALUATION

To evaluate the effectiveness of MPKAN, we seek answers to the following questions:

Q1: How does MPKAN perform in detecting various scenarios and datasets? (§IV. *B*)

Q2: What advantages does MPKAN have compared to other methods? (§IV. *C*)

Q3: What are the effects of different levels of edge information retention on the results? (§IV. *D*)

Q4: How much improvement does MPKAN offer in detection compared to MPNN? (§IV. *D*)

All experiments are conducted on a single host with 12th Gen Intel(R) Core(TM) i7-12700K 3.60 GHz, 96 GB physical memory. The OS is Ubuntu 20.04.6 LTS.

A. Dataset

TABLE III: Characteristics of Five Datasets

Dataset	APT Campaign	Raw data		Attack data	
		#Entity	#Event	#Entity	#Event
S-1	Watering Hole	6569	20036	14	3637
S-2	Malvertising dominate	13847	70011	14	4248
S-3	Spam campaign	7867	27055	27	2729
S-4	Pony campaign	11752	28504	21	8406
Darpa	Backdoor, Port Scan	25021	3227015	11317	2486136

We evaluate MPKAN on four single-host datasets from Atlas [18] and the Darpa THEIA dataset [19]. Table III summarizes these five datasets. This table outlines the key attack characteristics of various APT strategies. These attack scenarios encompass most mainstream APT attack methods, thus which can reflect detection capabilities effectively.

B. Effectiveness Analysis

TABLE IV: Effectiveness Analysis

	Accuracy	Precision	Recall	F-Score	FPR	FNR
S1	0.9931	1.0000	0.9677	0.9836	0.0000	0.0323
S2	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000
S3	0.9947	1.0000	0.9474	0.9730	0.0000	0.0526
S4	0.9903	1.0000	0.9592	0.9792	0.0000	0.0408
Ave. S	0.9945	1.0000	0.9686	0.9840	0.0000	0.0314
Darpa	0.9899	0.9983	0.9727	0.9853	0.0009	0.0273

Experimental Setup. We use events as the detection granularity and employ supervised learning methods. We split the data into 80% for the training set and 20% for the test set, and we train the model with 50 epochs. We test the five datasets individually, calculating key metrics such as accuracy, Precision, Recall, F-Score, FPR, and FNR to evaluate the model's performance. We compute the average performance across the four datasets within the Atlas dataset, using this as a basis for comparison with other methods.

Results Analysis. From the experimental metrics in the table IV, it can be observed that MPKAN excels in detecting a variety of APT attacks, demonstrating robust performance across different scenarios. When applied to four real-world attack scenarios from the ATLAS dataset, the model's average accuracy consistently reaches 0.9945, and its F1 score stands at 0.9840, indicating a high balance between precision and recall. Such results underscore the reliability of MPKAN in identifying true threats while minimizing false positives. Additionally, in the mixed attack dataset from the Darpa dataset, the performance remains consistently high, with the accuracy reaching 0.9899 and the F1 score achieving 0.9853. These metrics reflect the model's ability to generalize well across various attack types and datasets.

C. Comparative Analysis

TABLE V: Comparative Analysis on ATLAS

Method	Precision	Recall	F-score
Graph-traversal	0.1782	1.0000	0.3026
Non-optimized causal graph	0.8758	0.4155	0.5636
Oversampling-only model	0.9785	0.7964	0.8781
One-hot encoding	0.9960	0.8075	0.8919
Support Vector Machine (SVM)	0.8712	0.9042	0.8874
ATLAS	0.9988	**0.9989**	**0.9988**
MPKAN	**1.0000**	0.9686	0.9840

TABLE VI: Comparative Analysis on DARPA

Method	Precision	Recall	F-score
Deeplog	0.1600	0.1400	0.1499
LogRobust	0.4240	0.3500	0.3849
LogGAN	0.3560	0.3300	0.0040
Log2Vec	0.6200	0.6600	0.6400
MPKAN	**0.9983**	**0.9727**	**0.9853**

Experimental Setup. We compare different methods on two datasets with different input formats. On the ATLAS dataset, we use several baseline methods to evaluate the performance of the MPKAN model, including graph traversal methods which identify connection patterns in graphs through traversal techniques, non-optimized causal graph methods which construct causal graphs to explain causal relationships between events, oversampling-only models which address class imbalance by increasing the number of samples in minority classes through oversampling techniques, one-hot encoding methods which transform features into one-hot encoded vectors, and support vector machine (SVM), a classic machine learning algorithm that classifies data points by finding the optimal hyperplane. On the Darpa dataset, we compare several log detect methods.

Results Analysis. From Table V, it can be seen that MPKAN outperforms most baseline models. Its precision value is higher than that of Atlas, but its recall value is lower, indicating that the MPKAN model is more conservative, leading to a slightly lower F1 score compared to Atlas.

From Table VI, it can be seen that MPKAN performs better than other APT detection solutions. MPKAN enhances semantic analysis of nodes and edges, applying meta-path random walks and an improved MPNN algorithm, significantly improving the ability to capture complex attack patterns. This makes MPKAN a robust and effective approach for detecting advanced threats in modern security environments.

D. Modular Analysis

Experimental Setup. To investigate the impact of different edge information on overall performance, we design the following three comparative experiments: not using edge information, using only one type of edge information, and using the fused edge information mentioned in this paper. Additionally,

to validate the advantages of the MPKAN method compared to the MPNN, we conduct tests across these three types of edge information. Since the edge information in the Darpa dataset is more extensive, we perform experiments on the Darpa dataset. We use accuracy and F1 scores as comprehensive evaluation metrics, as shown in Fig. 4 and Fig. 5, respectively.

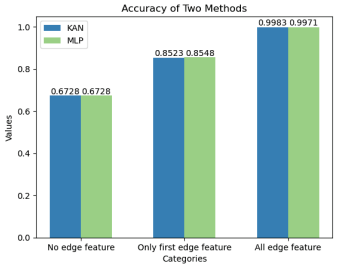


Fig. 4: Acc of Diff Edge Info

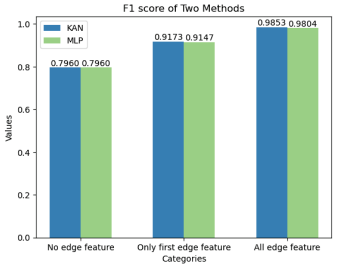


Fig. 5: F1 of Diff Edge Info

Results Analysis. It can be observed that without edge information, neither method can learn edge features, so only node features are used. Therefore, when edge information is absent, both methods have equal accuracy and F1 scores. When using only one type of edge, because only the relationship of the edge is retained and most of the interaction information between entities is removed, the accuracy and F1 scores of both methods are similar, and they cannot effectively distinguish between events. However, when using the fused semantics of multiple edges as proposed in this paper, MPKAN achieves higher accuracy and F1 scores compared to the traditional MPNN. This is because MPKAN can flexibly learn both node and edge features, allowing it to effectively distinguish between benign and malicious events based on the learned model.

V. Conclusion

In this paper, MPKAN addresses the problem of semantic loss in traditional provenance graphs used for audit log analysis by introducing a method designed to enhance the detection of APT attacks. MPKAN enhances these graphs by focusing on sequences of operations between entities and strengthens both node and edge features. The main contributions of this approach include utilizing meta-path random walks to deepen semantic connections among nodes, preserving multiple operational events to enrich edge attributes, and integrating the Knowledge Association Network with the MPNN algorithm to improve node embeddings. This enhances the model's ability to detect complex patterns. Evaluation on the ATLAS and Darpa datasets demonstrates MPKAN's effectiveness, achieving an average accuracy of 0.9899 and an F1 score of 0.9853, confirming its robustness in complex attack scenarios.

Acknowledgment

This research is supported by National Key Research and Development Program of China (No.2023YFC2206402), 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

[1] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019.

[2] "Advanced Persistent Threats: a Symantec Perspective."

[3] "APT Attacks: Exploring Advanced Persistent Threats," May 2023.

[4] "Big Data Analytics for Sophisticated Attack Detection."

[5] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "HOLMES: Real-Time APT Detection through Correlation of Suspicious Information Flows," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1137–1152, May 2019. ISSN: 2375-1207.

[6] Y. Kwon, F. Wang, W. Wang, K. H. Lee, W.-C. Lee, S. Ma, X. Zhang, D. Xu, S. Jha, G. Ciocarlie, A. Gehani, and V. Yegneswaran, "MCI : Modeling-based Causality Inference in Audit Logging for Attack Investigation," in *Proceedings 2018 Network and Distributed System Security Symposium*, (San Diego, CA), Internet Society, 2018.

[7] X. Shu, F. Araujo, D. L. Schales, M. P. Stoecklin, J. Jang, H. Huang, and J. R. Rao, "Threat Intelligence Computing," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, (New York, NY, USA), pp. 1883–1898, Association for Computing Machinery, Oct. 2018.

[8] N. Hossain, S. M. Milajerdi, J. Wang, B. Eshete, R. Gjomemo, R. Sekar, S. D. Stoller, and V. N. Venkatakrishnan, "SLEUTH: Real-time Attack Scenario Reconstruction from COTS Audit Data,"

[9] Z. Xu, P. Fang, C. Liu, X. Xiao, Y. Wen, and D. Meng, "DEPCOMM: Graph Summarization on System Audit Logs for Attack Investigation," in *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 540–557, May 2022. ISSN: 2375-1207.

[10] J. Zengy, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua, "SHADEWATCHER: Recommendation-guided Cyber Threat Analysis using System Audit Records," in *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 489–506, May 2022. ISSN: 2375-1207.

[11] W. U. Hassan, A. Bates, and D. Marino, "Tactical Provenance Analysis for Endpoint Detection and Response Systems," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1172–1189, May 2020. ISSN: 2375-1207.

[12] J. Zeng, Z. L. Chua, Y. Chen, K. Ji, Z. Liang, and J. Mao, "WATSON: Abstracting Behaviors from Audit Logs via Aggregation of Contextual Semantics," in *Proceedings 2021 Network and Distributed System Security Symposium*, (Virtual), Internet Society, 2021.

[13] H. Ding, J. Zhai, Y. Nan, and S. Ma, "{AIRTAG}: Towards Automated Attack Investigation by Unsupervised Learning with Log Texts," pp. 373–390, 2023.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Sept. 2013. arXiv:1301.3781 [cs].

[15] X. Fu, J. Zhang, Z. Meng, and I. King, "MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding," in *Proceedings of The Web Conference 2020*, pp. 2331–2341, Apr. 2020. arXiv:2002.01680 [cs].

[16] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," June 2017. arXiv:1704.01212 [cs].

[17] Z. Sun, M. Chen, W. Hu, C. Wang, J. Dai, and W. Zhang, "Knowledge Association with Hyperbolic Knowledge Graph Embeddings," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (B. Webber, T. Cohn, Y. He, and Y. Liu, eds.), (Online), pp. 5704–5716, Association for Computational Linguistics, Nov. 2020.

[18] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, "{ATLAS}: A Sequence-based Learning Approach for Attack Investigation," pp. 3005–3022, 2021.

[19] "Transparent-Computing/README-E3.md at master · darpa-i2o/Transparent-Computing."