

A Graph Learning-Based Approach for Lateral Movement Detection

Mahdi Rabbani¹, Leila Rashidi, and Ali A. Ghorbani¹, *Senior Member, IEEE*

Abstract—Lateral movement, a crucial phase in the Advanced Persistent Threat (APT) life cycle, refers to a strategy employed by adversaries to traverse horizontally within a network. The aim is to gain access to various systems or resources, thereby expanding their control and potential access to valuable targets. Detecting these attacks becomes challenging for conventional detection systems due to various factors, including the complexity of pathways, the mimicking of legitimate user behavior by attackers, and limited network visibility. To address these challenges, advanced detection techniques are required to effectively and dynamically analyze multiple features within the interconnected structure of the network. This paper introduces an innovative approach to detect malicious lateral movement paths by leveraging authentication events and graph learning techniques. The proposed method involves constructing a heterogeneous graph, and employing DeepWalk for node embedding. By combining node embedding features with the temporal information of authentication events, feature vectors are generated for each authentication request. These features are then used to train multiple machine learning-based classifiers to detect malicious lateral movement paths. Furthermore, to assess the model's performance in a more realistic scenario, a series of additional experiments were conducted. These experiments provided further validation of the model's robustness and its capability for forward prediction.

Index Terms—Graph learning, machine learning, lateral movement detection, advanced persistent threat.

I. INTRODUCTION

LATERAL movement is a tactic used by cyber attackers to traverse horizontally within a network. They use this method to gain access to valuable targets by escalating privileges while mimicking the behavior of legitimate users [1]. The consequences of undetected lateral movement can be severe, including unauthorized access to sensitive systems and theft of data. Therefore, it is crucial to have robust detection mechanisms in place. Traditional techniques for detecting lateral movement are inefficient due to their limited visibility and inability to handle the complexity of modern

network architectures [2]. They also generate a high rate of false positives or false negatives. These techniques rely on signature-based or rule-based systems, which need to be more adaptable enough to keep up with evolving attack methods.

The way that graphs are represented often starts with an adjacency matrix, which shows the connections between adjacent vertices in the graph. However, this method becomes less effective when dealing with complex and irregular graphs, especially those that come from heterogeneous networks. Additionally, traditional graph representations can be difficult to work with and time-consuming to use in real-world situations [3]. Therefore, there is a need for a strong graph representation method that can extract and transform various types of graph-related data (including vertex-centric, edge-centric, and subgraph-centric data) while also revealing hidden relationships within the dataset [4]. Moreover, while graph theory provides the basic principles for comprehending complex network structures, it does not possess the inherent ability to utilize information (features) and make decisions based on the features embedded in the graph structure. To practically apply these principles, especially in problem-solving within a network, additional methodologies such as graph learning are required.

Graph learning involves employing machine learning techniques to facilitate a more dynamic and feature-driven decision making process and discover the relationships among various elements within a network. The essential nature of graph learning is its capability to transform complex graph properties into feature vectors. These vectors serve as a foundation for training machine learning models. Unlike conventional approaches that requires projecting graphs into low-dimensional spaces, graph learning techniques adeptly convert graph-based data elements (like nodes, edges, weights, etc.) into corresponding outputs, known as graph embeddings [5]. This transformative process enhances downstream tasks such as node classification and link prediction without the need for a separate embedding step [6]. There are various graph learning methods, including Matrix Factorization, Graph Signal Processing, Deep Learning, and Deep Walk-based techniques [7]. Out of the graph embedding methods, the DeepWalk technique stands out as a compelling choice for graph embedding. It generates sequences of nodes by carefully preserving inter-node relationships, thereby revealing detailed network information and transforming it into a more compact, lower-dimensional space. The DeepWalk technique mainly works by traversing neighboring nodes (first-order proximity) and collecting relevant

Manuscript received 20 November 2023; revised 26 March 2024; accepted 10 June 2024. Date of publication 13 June 2024; date of current version 16 October 2024. The authors sincerely appreciate the support received from the Canadian Institute for Cybersecurity (CIC), the funding provided through the CIC/IBM CRD project. The associate editor coordinating the review of this article and approving it for publication was M. Tornatore. (Corresponding author: Mahdi Rabbani.)

The authors are with the Canadian Institute for Cybersecurity, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: m.rabbani@unb.ca).

Digital Object Identifier 10.1109/TNSM.2024.3414267

information about the source node (the origin of the walk). The walk then progresses to more distant neighbors (second-order proximity) and keeps acquiring relevant insights. Eventually, this accumulated information becomes embedded within the source node, creating a comprehensive feature vector for the node [8], [9].

Practically, lateral movement occurs within a networked environment (such as an Intranet or an industry network), where all computers are interconnected [10]. In this scenario, a cybercriminal attempts to breach a host, infecting it with malicious software to gain remote control. This insidious control serves as a conduit for the host to traverse laterally, infiltrating other nodes and seeking valuable assets [11]. To identify lateral movements in such environments, it is crucial to accurately comprehend the structure of these interconnected hosts, considering the relationship between every pair of them [10]. We believe that the graph learning technique can effectively address these challenges through the following two phases. In the initial phase, graph learning facilitates the representation of the network's structure as a heterogeneous directed graph, utilizing node embedding to generate efficient feature vectors. Moving into the second phase, these generated feature vectors are employed to train a machine learning-based classifier, ultimately establishing an effective and efficient detection system for identifying malicious lateral movement paths among benign activities. The paper's main contributions are outlined as follows:

- We have developed an innovative approach for identifying malicious lateral movement paths based on network authentication logs. Our approach involves constructing a heterogeneous graph structure that traces the authentication sequences and creates a comprehensive feature vector. This feature vector includes both embedded features extracted through DeepWalk and non-embedding attributes. We then train various machine learning models to detect and identify these malicious lateral movement paths.
- We expanded the original dataset with additional information, incorporating both embedding and non-embedding features to enrich feature representation. Utilizing these enhanced features, we conducted a series of additional experiments to evaluate the model's performance in more realistic scenarios. This provided further validation of its robustness and its capability for forward prediction.
- We have created and made available the proposed approach, which can be accessed through a Python analytics interface. This contribution allows users to easily apply the method to their own datasets and scenarios in a practical and user-friendly way.

The rest of the paper is organized as follows. Section II provides an overview of existing studies on lateral movement detection using graph learning techniques. In Section III, we discuss the details of the proposed approach. The experimental analyses and corresponding results are presented in Section IV. Finally, Section V concludes the paper, providing final reflections and remarks.

II. LITERATURE REVIEW

In this section, we review and discuss recent approaches in the literature for lateral movement detection using graph learning techniques. The collaborative integration of graph embedding and machine learning techniques enhances the process of feature extraction by generating supplementary information, achieved through node embedding. The features generated through node embedding provide additional insight into the characteristics of path samples, effectively assisting the classifier in differentiating between malicious and normal paths.

Zhao et al. [12] introduced the Continuous-Temporal Lateral Movement Detection (CTLMD) technique, employing graph learning for lateral movement identification. Authentication events, both remote and local, were transformed into a directed homogeneous graph with timestamp attributes. This graph-based mapping effectively captures temporal lateral movement paths within complex network structures, resulting in the creation of a temporal path connection graph. The authors constructed a bipartite heterogeneous graph to extract implicit information from users and hosts ($G_B = (U, H, E_B)$, ($E_B \subseteq U \times H$)). The graph structure indicates directed connections between users and hosts, as well as mutual connections from hosts to users. This study explores lateral movement paths by assessing similarity among elements along those paths, computing both the cosine similarity of edge features within the path and the features of login entities independently.

Fang et al. [13] proposed LMTracker, a lateral movement path detection method grounded in graph embedding techniques. In this study, the authors modeled the network topology as a heterogeneous graph encompassing various nodes such as computers, users, processes, and files. The graph also incorporated multiple employee activities like logon, use, and create, representing the edges connecting these nodes. Authentication events and process start/stop events were utilized to construct the graph, effectively capturing both authentication and process activities. LMTracker's key components include the structure of the heterogeneous graph, node embedding, path vector representation, and the attack path AutoEncoder. After the graph embedding phase, each path within the graph was transformed into a feature vector consisting of 128 features. These path vectors were then used as input for an attack path detector based on the AutoEncoder algorithm. While LMTracker explored a broader scope of users' activities compared to our approach, it omitted the crucial temporal aspect, which is a pivotal feature in the progression of lateral movements. Additionally, their approach was evaluated using the LANL dataset, achieving an accuracy rate of 0.91.

Similarly, Bian et al. [14] utilized machine learning methods for lateral movement detection. They employed the graph representation of authentication events within the LANL dataset (the authentication graph $G = (U, V, E)$), extracting 29 features primarily based on the in/out-degree characteristics of distinct hosts (nodes). Following the feature extraction process, the authentication events within the initial 30-day period of the LANL dataset were used for training and

evaluating various machine learning classifiers. Due to the dataset's inherent imbalance, with a larger number of normal authentication paths compared to malicious paths, the authors applied a combination of under-sampling and over-sampling techniques. This approach aimed to create a balanced version of the LANL dataset, enhancing the model's ability to handle both classes effectively. Compared to our work, our proposed model demonstrates superior performance in both accuracy and processing time, achieving a higher F_1 score.

Zhao et al. [12] proposed a continuous-temporal lateral movement detection model (CTLMD) using graph embedding techniques. The CTLMD architecture consists of two primary phases: the graph pre-processing unit and the anomaly detection unit. In the pre-processing phase, the remote and local authentication events from the LANL dataset are transformed into distinct graph structures: the *Path Connection Graph* and the *Bipartite Graph*. Two graph embedding techniques, CTDNE and BiNE, are introduced to measure the familiarity of lateral movement paths and map nodes into d-dimensional vectors. Subsequently, these feature vectors are then used for model training and calculating path similarity to identify normal or malicious paths. The authors also extract additional features by leveraging user and host embeddings, representing a path within the graph. These features include *path edge features*, computed through the vector of the login entity v derived from the path connection graph, as well as *login entity features* indicating the normality of the login entity within the path l . Additionally, *user features* and *host features* were also incorporated. The authors utilized a logistic regression technique for model training, using 80 percent of the path data, and testing is conducted on the remaining 20 percent. An interesting aspect of this study is the exploration of different hyperparameters, with a special focus on the (Δt) parameter. The performance results indicate that $\Delta t = 3h$ outperformed $\Delta t = 1h$. The authors argue that with $\Delta t = 3h$, longer intervals between login events are generated, providing more accurate extraction of normal path patterns and encapsulating the familiarity of lateral movements among normal users. In a recent study by King and Huang [15], they introduced a lateral movement detection framework comprising a model-agnostic graph neural network and a model-agnostic sequence encoding layer, such as a recurrent neural network. They approached lateral movement detection as a temporal graph link prediction task, where discrete-time interactions in a network are treated as a series of graphs. Using a temporal link prediction model, the framework captures typical behavior patterns from historical snapshots and assigns probability scores to edges in subsequent instances. Edges with lower probability scores are representative of anomalous connections within the network. These anomalous connections are often indicative of lateral movement [16].

Kushwaha et al. [17] proposed a lateral movement detection approach based on user behavioral analysis. They combined user behavioral analysis and machine learning to develop domain-specific features for identifying such behavior at the individual user level. Utilizing the LANL dataset, they employed supervised machine learning algorithms to uncover

Lateral Movement behaviors within enterprise networks. Their approach involves analyzing user-based behavior related to authentication and process events, utilizing domain-specific features that represent indicators commonly used in lateral movement activities. During the model training phase, they employed the XGBoost classifier, achieving an average recall score of 86.51%. Comparatively, our proposed XGBoost model, based on node embedding features, demonstrates superior performance. Liu et al. [18] proposed Latte, a graph-based system designed to detect malicious lateral movement paths using network connection graphs from Windows security events. They demonstrated the effectiveness of graph-based algorithms in detecting lateral movement and incorporated a forensic analysis module to identify confirmed malicious entities. Furthermore, they integrated a general detection module to filter out benign paths to improve the detection accuracy. To assess Latte's scalability for general detection tasks, they conducted experiments on the penetration test dataset obtained from a large-scale network penetration test. They evaluated Latte's performance by ranking the graph's 2-hop paths using path-rate scores and a remote file execution detector individually. Notably, their detection system does not rely on machine learning and differs from our proposed model in terms of its capability for forward prediction. To summarize, our proposed work differs from previous studies in the following ways:

- Rather than dealing with complex graphs, our approach focuses on creating a lightweight graph structure that can convey the essential information obtained from authentication activities in enterprise networks. This is useful for detecting lateral movement.
- While using the DeepWalk technique to conduct node embedding features, we only explore first-order, second-order, and third-order proximate nodes. This approach has proven to be more effective than other methods that involve long walks, such as [13].
- Unlike previous works that aim to embed all attributes from authentication events, our model combines temporal non-embedding and embedding features to provide better feature representation and extraction. This helps to distinguish between normal and malicious paths, especially for forward prediction.
- When compared to previous methods, our model has demonstrated superior performance in terms of accuracy, processing time, and F_1 score. This highlights its effectiveness in detecting lateral movement.

III. PROPOSED APPROACH

In this section, we utilize a subset of the LANL dataset [19], [20] focused on authentication events, which includes logs recorded by the Windows Active Directory. This dataset serves as the basis for the analysis and experimentation presented in this paper. Each row of the dataset, containing benign authentication events, can be represented as the following tuple.

$$(time, C_{src}, U, C_{dst}, T, A)$$

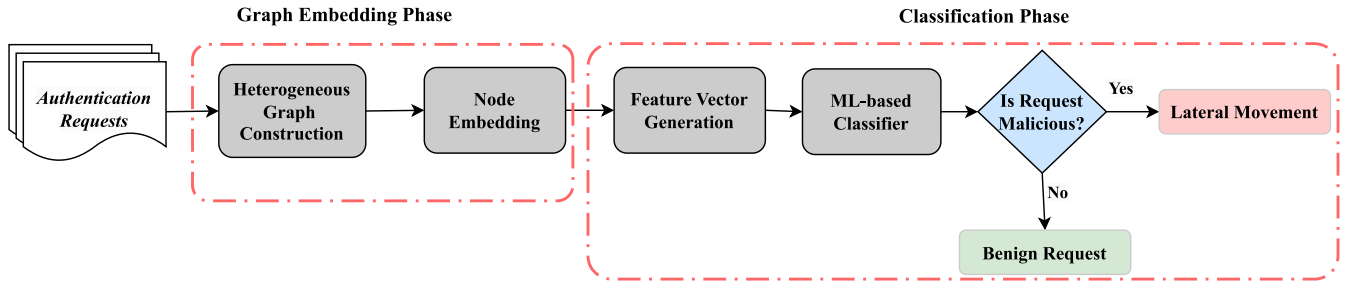


Fig. 1. Workflow of the proposed methodology for lateral movement detection.

In this tuple, the variable *time* denotes the timestamp at which the authentication request was submitted, while C_{src} and C_{dst} respectively represent the source and destination computers involved in the authentication event. Moreover, within the tuple, U signifies the user, T denotes the event type (including login, logout, process creation, file creation, etc.), and A represents the authentication attributes (such as login type, authentication type, and login result). Due to the absence of information for T and A in the malicious authentication records (redteam records) within the dataset, we could not utilize this data for analysis. As a result, the dataset comprises a restricted set of features for each authentication request. Upon closer inspection, certain authentication records exhibit exactly similar attributes ($Computer_A \rightarrow User_1 \rightarrow Computer_B$), yet are associated with distinct labels. Indeed, such scenarios are reflective of real-world lateral movement situations. For instance, in a compromised network within a company, a legitimate user (U_1) might regularly use a computer (C_{src}) to connect to workstation (C_{dst}) during normal working hours. In this case, since the movement is benign, it would be labeled as 0. However, if the same path is utilized for a malicious activity, especially during non-working hours like midnight, the path is labeled as 1 to signify its malicious nature.

Utilizing such features for training a machine learning classifier could introduce ambiguity and negatively impact the model's detection performance. Graph learning offers a promising solution to tackle this challenge. Given the involvement of three entities (source, destination, and user) in each authentication request, it is possible to construct a graph based on these authentication requests. Subsequently, this graph can be utilized to generate an array of features for each entity by using graph embedding.

A. Graph Structure

To create a graph learning-based model, the first step involves constructing a topological structure, which is essentially a graph made up of interconnected nodes. The purpose of this graph is to represent the relationships between the various elements of the network in a coherent and meaningful way. A graph structure is capable of retaining the structural information of the network while also incorporating relevant details from complex and semantic data sources. In this step, we convert the information related to each authentication

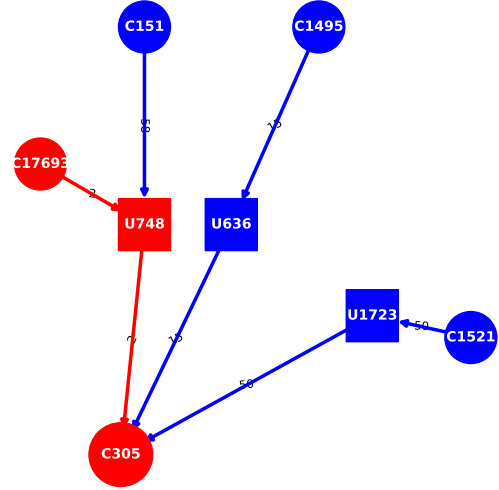


Fig. 2. A part of the graph comprising the edges that represent the requests outlined in Table I.

TABLE I
THE PROVIDED INFORMATION PERTAINS TO A SUBSET OF REQUESTS
AIMING TO ACCESS HOST C305 WITHIN THE LANL DATASET

	Time (day)	C_{src}	U	C_{dst}	Status
Request 1	50	C1521	U1723	C305	Benign
Request 2	50	C1495	U636	C305	Benign
Request 3	58	C151	U748	C305	Benign
Request 4	2	C17693	U748	C305	Malicious

request, including the source and destination hosts and the user, into a directed heterogeneous graph representation.

The graph depicted in Figure 2 is a directed heterogeneous graph created from the authentication records listed in Table I. In this figure, C305 serves as the common host destination in four distinct authentication records. Among these, three records indicate benign activity (depicted in blue), while one record signifies lateral movement (highlighted in red). It is important to note that the illustrated figure represents only a very small subset of our proposed directed heterogeneous graph.

As explained in the previous section, the authentication records including source computer (C_{src}), user (U), destination computer (C_{dst}), and timestamp provide us with crucial insights into an authentication path. This path originates from a designated source host, is executed by a specific user, and concludes at a destination host. In the subsequent step, we map each authentication record into a path within the directed

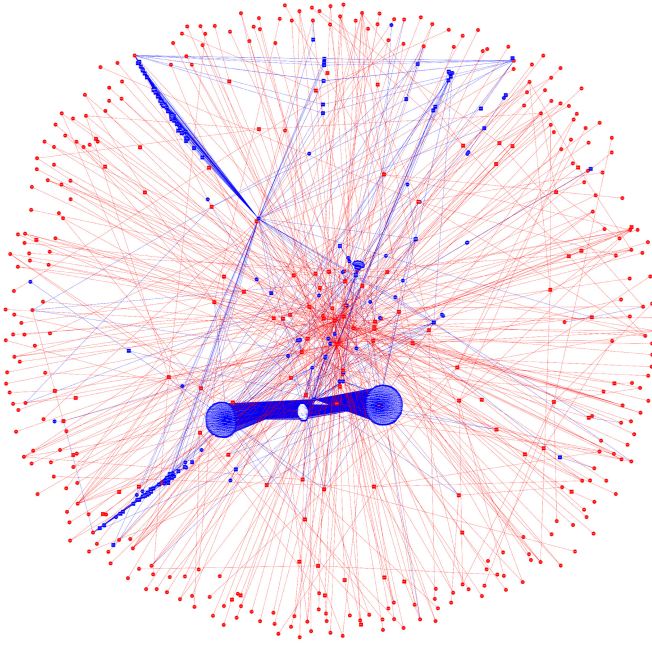


Fig. 3. The proposed directed heterogeneous graph representing an extremely limited subset of authentication events extracted from the dataset.

heterogeneous graph. In the context of directed graphs, a path comprises multiple directed edges, illustrated as follows:

$$\text{Time} \xrightarrow{\text{occur/Second}} \text{Computer} \xrightarrow{\text{Use}} \text{User} \xrightarrow{\text{Login}} \text{Computer}$$

Utilizing the above-mentioned path structure, we proceeded to construct a directed heterogeneous graph. A larger portion of the original graph based on real records in the LANL dataset is represented in Figure 3. As depicted in the provided figure, a significant number of normal paths are concentrated at the center (depicted in blue). These paths signify the usual connections between nodes within an organization or an Intranet. In addition to the centralized blue paths, there are also sprawling red paths that carry a significant implication regarding lateral movement paths. When a computer within an intranet becomes infected with malware for the purpose of executing malicious lateral movements, the subsequent movements from the compromised host to other hosts do not follow any predefined pattern. These movements occur blindly, with the purpose of inspecting each successive computer for significant assets or valuable information. Algorithm 1 provides the key steps used to convert authentication events from the dataset into a heterogeneous graph structure. The output of this algorithm enhances our ability to visualize the relationships among various network elements within a graph-based topology.

Two primary phases are involved in the proposed methodology for lateral movement detection: graph embedding and classification, as shown in Figure 1. We will provide detailed insights into each individual component's internal architecture in the following sections.

Algorithm 1: Graph Generation and Visualization

Input: Authentication events from the LANL dataset

Result: The equivalent graph representation

- 1 Extract authentication events from the dataset and assign labels;
 - 2 Data Preprocessing the concatenate dataframes;
 - 3 Create a directed heterogeneous graph using the preprocessed data;
 - 4 Define node shapes for visualization;
 - 5 **for** each row in the preprocessed data **do**
 - 6 **if** the label indicates malicious activity **then**
 - 7 set the node color to red;
 - 8 **else**
 - 9 set it to blue.
 - 10 **end**
 - 11 **end**
 - 12 **end**
 - 13 Add nodes to the graph for source, destination, and intermediary nodes;
 - 14 Add edges between source-destination and destination-intermediary nodes with appropriate color and weight;
 - 15 Layout the graph and draw edge labels and edges with arrowheads;
 - 16 Set arrow style, arrow size, width, and edge colors based on their attributes;
 - 17 Draw node labels and nodes;
 - 18 Set node font color and weight for labels;
 - 19 **for** each defined node shape **do**
 - 20 draw nodes with specified sizes and colors;
 - 21 **end**
-

B. Authentication Graph Embedding and Feature Vector Generation

Graph Embedding refers to a mathematical function that transforms a high-dimensional representation of a node in a graph into a lower-dimensional space. The primary goal is to generate appropriate feature vectors that can be employed as input to train machine learning models for the objective of anomaly detection. Indeed, there are various graph embedding algorithms, including DeepWalk [8], node2vec [9], structure2vec [21], and LINE [22]. Many of these algorithms draw inspiration from the word2vec concept [13]. Among these, DeepWalk is particularly well-suited for heterogeneous graphs. Therefore, in our approach, we employ Random Walk with a predetermined walk length. Additionally, for node representation, we utilize the Skip-Gram method to sample the sequence set of random walks. This technique not only captures the structural information within the graph but also preserves its semantic attributes [13].

Continuous Bag Of Words (CBOW) and Skip-Gram [23] are two different architectures employed for generating word embedding. CBOW receives context as input to predict words, while skip-gram receives input words to predict context [23]. In our proposed node embedding model, the goal is to extract

Algorithm 2: Node Embedding Method

Input: Graph G , Walk length L , Number of walks R , Number of workers, dimension of the embedded vector d , Learning rate α

Result: Node embeddings dictionary

```

1  $sequences \leftarrow []$ ;
2 Function  $DeepWalk(G, L, R, workers)$ :
  // Generate random walk sequences;
3   foreach  $node$  in  $G.nodes$  do
4     for  $r$  in  $range(R)$  do
5        $walk \leftarrow [node]$ ;
6       while  $len(walk) < L$  and  $currentNode$  has neighbors do
7          $cur \leftarrow walk[-1]$ ;
8          $currentNode \leftarrow$  a random neighbor of  $cur$ ;
9         append  $currentNode$  to  $walk$ ;
10      append  $walk$  to  $sequences$ ;

  // Training the Node2Vec model using Skip-Gram;
11   $node\_embeddings \leftarrow random\_init(G.num\_nodes, d)$ ;
12  foreach  $sequence$  in  $sequences$  do
13    for  $i$  in  $range(len(sequence))$  do
14       $center\_node \leftarrow sequence[i]$ ;
15       $context\_nodes \leftarrow sequence[\max(0, i - L) : i] + sequence[i + 1 : \min(i + L + 1, len(sequence))]$ ;
16      foreach  $context\_node$  in  $context\_nodes$  do
17         $center\_embedding \leftarrow node\_embeddings[center\_node]$ ;
18         $context\_embedding \leftarrow node\_embeddings[context\_node]$ ;
19         $dot\_product \leftarrow dot(center\_embedding, context\_embedding)$ ;
20         $predicted\_prob \leftarrow softmax(dot\_product)$ ;
21         $error \leftarrow predicted\_prob - actual\_prob$ ;
22         $node\_embeddings[center\_node] - = \alpha \times error \times context\_embedding$ ;
23         $node\_embeddings[context\_node] - = \alpha \times error \times center\_embedding$ ;
24  return  $Node\_embeddings$ 

```

additional information from both the first-order proximity and the second-order proximity of graph's nodes. This information is then converted into a normalized numerical feature vector. Subsequently, these feature vectors will be employed to train the proposed machine learning-based classifier model.

The Skip-Gram model is constructed with a neural network architecture that includes a single hidden layer, which is utilized to train the weights associated with this hidden layer. The Skip-Gram model aims to maximize the conditional probability of a node v ($N_t(v), t \in T_V$) and obtain the probability distribution of nodes that are likely to appear together with it within a context window of a specified size in the proposed heterogeneous graph $G = \langle V, E, T \rangle$ [13].

The objective function of SkipGram, also known as the negative log likelihood (F_{sg}), aims to maximize the likelihood of predicting context nodes given a center node. This function is expressed in Eq. (1).

$$F_{sg} = \max \sum_{v \in V} \sum_{t \in T_V} \sum_{c_t \in N_t(v)} \log P(c_t|v, \theta), \quad (1)$$

Here, V represents the set of all nodes in the graph, T_V is the set of types associated with nodes in V . $N_t(v)$ is the set of nodes in the context of node v associated with type t . The

probability $P(c_t|v, \theta)$ is the Softmax activation function and is calculated using Eq. (2). θ represents the parameters to be learned.

$$P(c_t|v, \theta) = \frac{e^{X_{c_t} X_v}}{\sum_{(u_t \in N_t(v))} e^{X_{u_t} X_v}}, \quad (2)$$

where X_{c_t} is the embedding of context node c_t , and X_v is the embedding of center node v . $\sum_{(u_t \in N_t(v))}$ represents the sum over all context nodes in the context of node v associated with type t .

As illustrated in Figure 4, the output features have been extracted as a feature vector for each authentication path, comprising 128 features for each node. Consequently, a total of 385 features have been extracted for each authentication path. This includes 384 features corresponding to the three nodes in the path ($Computer_{(A)} \rightarrow User_{(1)} \rightarrow Computer_{(B)}$), along with an additional feature indicating the request time. Algorithm 2 outlines the essential steps for generating node embeddings using the DeepWalk method. The pseudocode includes a function, `random_walk`, responsible for executing a random walk of length L starting from a given node in the graph G . The Skip-Gram algorithm is then applied to learn

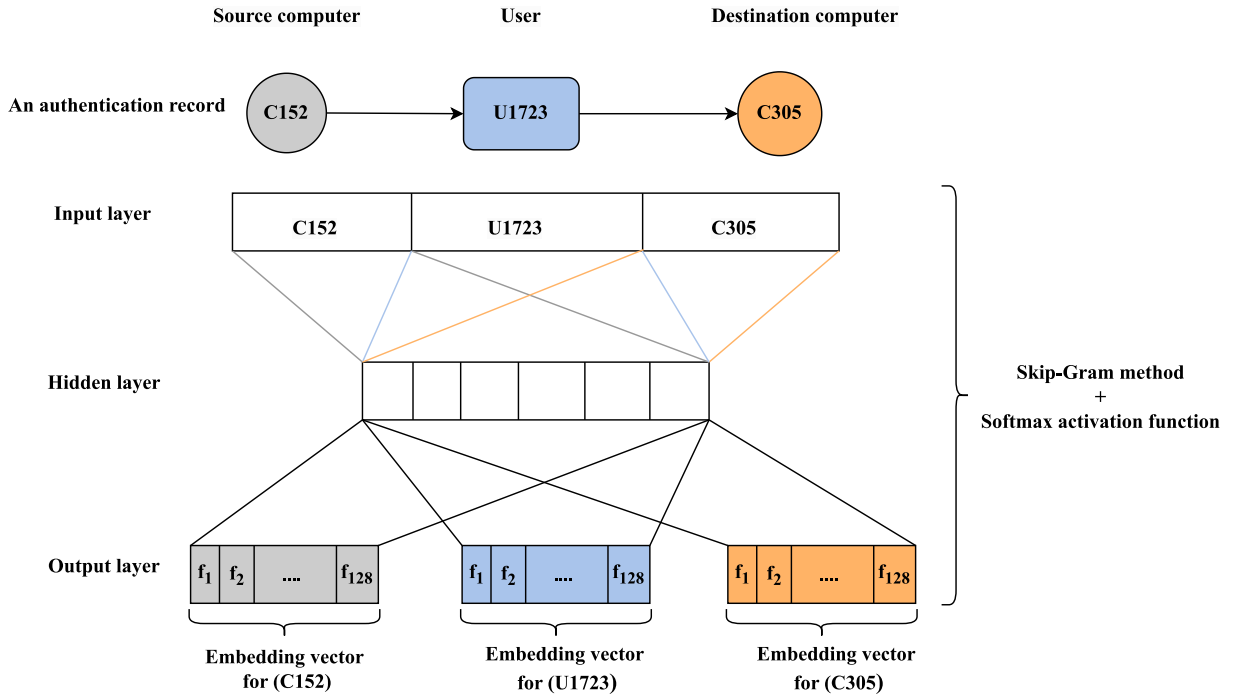


Fig. 4. Illustration of the node embedding process.

node embeddings based on the sequences generated by these random walks.

C. A Machine Learning-Based Classifier for Detecting Lateral Movements

In the previous section, we explained how node embedding works and how it generates features through graph embedding techniques. We used node embedding to create 385 normalized features for each authentication record in our dataset. We then used these features to train a machine learning-based classifier. This classifier is a critical part of our lateral movement detection system as it serves as the detection engine. It carefully identifies malicious paths among a significant number of normal paths.

To establish an efficient and resilient classifier model through supervised learning methods, we incorporated a range of techniques such as Support Vector Machine (SVM), K-Nearest Neighbors (K-NN), logistic regression, gradient boosting (GB), and Extreme Gradient Boosting (XGBoost), while carefully tuning hyperparameters for optimal performance. The goal of applying different machine learning techniques is to identify the best approach for developing a lightweight model that can quickly detect anomalies with high accuracy. In order to choose the optimal method, we evaluated each technique based on various evaluation metrics, such as accuracy, precision, recall, F_1 score, and processing time. We have presented the performance results of all the supervised learning techniques we evaluated in Table XII. According to our analysis, XGBoost technique [24] demonstrated superior performance compared to other techniques in terms of accuracy, recall, precision, F_1 score, and AUC. Although Logistic Regression provides faster detection

time, it has significantly lower performance in terms of precision, recall, and F_1 score. Therefore, we decided to train the proposed model using XGBoost classifier, which is a scalable distributed gradient-boosted decision tree machine learning technique. XGBoost offers parallel tree boosting, an efficient and rapid machine learning approach designed to address regression, classification, and ranking problems effectively [24].

In Section III-B, we explained that authentication events contain embedded features that provide information about the entities involved (source computer (C_{src}), user (U), destination computer (C_{dst})). Time is another crucial attribute that needs to be integrated into the authentication paths to generate final feature vectors. Algorithm 3 outlines the primary steps for node concatenation, training, and prediction. The *Explode* function generates feature vectors that combine both embedded features generated from Algorithm 2 and non-embedded features. This combined feature set is used to create a new dataset for training and prediction. The algorithm prepares the data and utilizes a machine learning classifier to distinguish between malicious and normal paths using the feature vectors.

IV. EXPERIMENTAL RESULTS AND EVALUATION

We evaluate the proposed approach using the Comprehensive Multi-source Cybersecurity Events dataset, which was collected over a span of 58 days from Los Alamos National Lab's network. This dataset includes logs of 1,648,275,307 authentication requests, involving 12,425 users and 17,684 computers [19], [20]. Performance metrics are explained in Section IV-A, followed by a detailed explanation of experiments and results in Section IV-B.

Algorithm 3: New Dataset Generation and Model Training

Input: Embeddings data, Initialized ML-based Model
Result: New path features (f), Path classification
 // Creating the final path features

```

1 Function explode( $x$ ):
2    $node \leftarrow$  split elements of  $x$  (elements of events);
3    $f \leftarrow$  empty feature vector;
4   foreach  $i$  in  $node$  do
5     if  $i$  starts with “U” or “C” then
6        $f \leftarrow$  append embeddings of  $i$ ;
7     else
8        $f \leftarrow$  append time of the event;
9   return  $f$ ;

  // Training the XGBoost model
10 Function train( $X, Y$ ):
11    $X_{\text{train}} \leftarrow$  feature vectors created using explode;
12   train classifier using  $X_{\text{train}}$  and  $Y_{\text{train}}$ ;
  // prediction using classifier
13 Function predict( $X$ ):
14    $X_{\text{pred}} \leftarrow$  feature vectors created using explode;
15    $Y_{\text{predicted}} \leftarrow$  predict labels using ML classifier;
16   return  $Y_{\text{predicted}}$ ;

  // Evaluate the classifier
17 Function evaluate( $Y, Y_{\text{predicted}}$ ):
18   results  $\leftarrow$  compute_metrics( $Y, Y_{\text{predicted}}$ );
19   return results;
```

TABLE II
CONFUSION MATRIX

		Actual	
		Lateral Movement	Benign Requests
Predicted	Lateral Movement	TP	FP
	Benign Requests	FN	TN

A. Evaluation Metrics

In the performance evaluation of machine learning-based approaches, a confusion matrix is often generated to assess the effectiveness of the classifier model. When it comes to detecting malicious lateral movement paths, the usual scenario involves identifying malicious paths (class 1) versus benign paths (class 2).

- **Confusion Matrix:** This matrix compares predicted patterns with actual ones. Table II presents the evaluation components for a system designed to detect malicious lateral movement paths, distinguishing between malicious and benign paths.
- **Overall Accuracy:** This metric evaluates the classifier’s performance by calculating the percentage of correctly classified patterns (TP, TN) compared to misclassified patterns (FP, FN). The following equation depicts the calculation of overall accuracy:

$$\text{Overall Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

- **Recall (Sensitivity or True Positive Rate):** This metric represents the proportion of malicious authentication requests that have been correctly classified as lateral movement.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4)$$

- **Precision:** This metric quantifies the ratio of accurately classified malicious lateral movement paths to the total number of predicted paths (both true and false positives) in the test data. It is calculated using the formula:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5)$$

- **F_1 Score:** This metric is a harmonic mean of precision and recall, providing a balanced measure of a classifier’s performance. It is calculated using the following formula:

$$F_1 \text{ Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6)$$

Indeed, the F_1 score is particularly useful in situations where there is an imbalance between the classes in a dataset, making it a suitable metric for evaluating network anomaly detection systems. It provides a balanced assessment of a classifier’s ability to capture both precision and recall, considering both false positives and false negatives.

- **Area under the ROC Curve (AUC):** is a well-known metric, particularly for binary classification problems, as it measures the area under the ROC curve. In classifiers that produce continuous values such as XGBoost, a threshold is employed to distinguish between positive and negative samples. The ROC curve illustrates the True Positive Rate (TPR) against the False Positive Rate (FPR) for various thresholds ranging from 0 to 1. The AUC value can be calculated using the following equation:

$$AUC = \int_0^1 \text{TPR} d(\text{FPR}) \quad (7)$$

This integral represents the area under the curve of the ROC plot, providing a summary measure of the classifier’s performance. Figure 5 depicts the classifier’s performance based on the ROC curve.

B. Evaluation Results

The experiments conducted on the LANL dataset to evaluate the performance of the proposed graph learning-based technique using the specified evaluation metrics. In this experiment, we considered both normal and malicious lateral movement paths. The testing dataset comprised 218 records for malicious lateral movement paths and 90,104 records for normal paths. These records were utilized to comprehensively evaluate the performance of the proposed technique. The confusion matrix presented in Table III demonstrates promising results, indicating the accuracy of the proposed

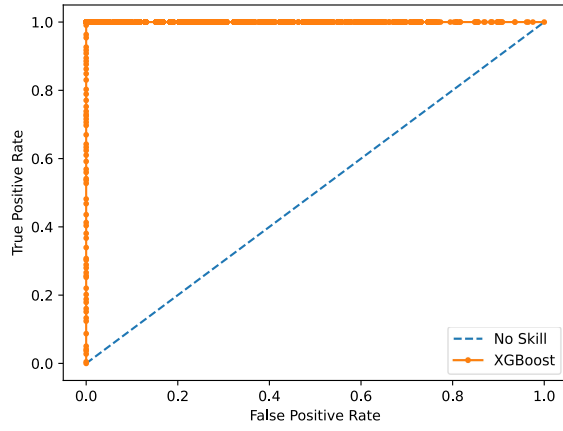


Fig. 5. The performance of the classifier based on the ROC curve.

TABLE III

CONFUSION MATRIX FOR LATERAL MOVEMENT DETECTION USING THE PROPOSED CLASSIFIER ON THE LANL DATASET

		Benign paths		Malicious paths	N_t
		Class 1	Class 2		
Benign paths	Class 1	90,104	0	90,104	
Malicious paths	Class 2	0	218	218	

Note: N_t represents the number of samples used for testing.

TABLE IV

RESULTS OF PERFORMANCE METRICS USING THE LANL DATASET (BEST RESULTS)

	Benign paths		Malicious paths		Macro average	Weighted average
	Class 1	Class 2				
Precision	1.00	1.00	1.00	1.00	1.00	1.00
Recall	1.00	1.00	1.00	1.00	1.00	1.00
F_1 Score	1.00	1.00	1.00	1.00	1.00	1.00
Accuracy	N/A	N/A	1.00	1.00	1.00	1.00
AUC	N/A	N/A	1.00	1.00	1.00	1.00

approach in classifying authentication requests. Considering the inherent class imbalance in the dataset, we conducted a detailed analysis by computing the recall, precision, and F_1 score for each class individually. This step ensured the verification of any potential misclassification between the normal class (Class 1) and the malicious class (Class 2). Furthermore, we calculated the area under the curve (AUC) value, a widely recognized metric, particularly suitable for binary classification problems.

Table IV presents all the specified metrics, including their weighted averages and macro averages. Additionally, Figure 6 provides visual representations of the confusion matrices for two different experiments, representing the best and worst outcomes. These results were obtained after conducting 10 iterations of randomly shuffling the dataset to create training and testing subsets. Specifically, Figure 6(a) illustrates the confusion matrix for the best outcome, while Figure 6(b) presents the one for the worst outcome along with the results of the performance metrics for the worst case presented in Table V. In order to represent the binary classification results in a 2-D space, Figure 7 shows the projection of the input features from their n-dimensional space into a two-dimensional plane. This is done using the t-SNE algorithm, which visually

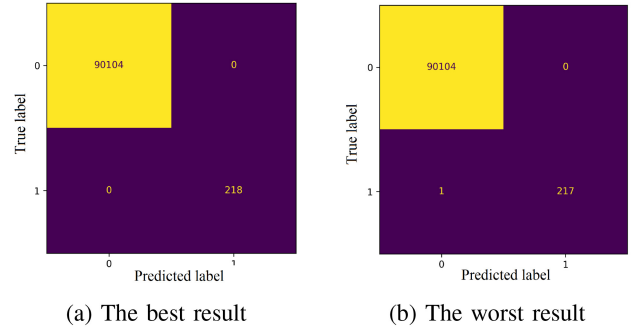


Fig. 6. The best and worst confusion matrices resulting from 10 experiments, where training and testing data were chosen randomly. Labels 0 and 1 correspond to Class 1 (benign path) and Class 2 (malicious path), respectively.

TABLE V

RESULTS OF PERFORMANCE METRICS USING THE LANL DATASET (WORST RESULTS)

	Benign paths		Malicious paths		Macro average	Weighted average
	Class 1	Class 2				
Precision	1.00	1.00	1.00	1.00	1.00	0.99
Recall	0.99	1.00	0.99	0.99	0.99	0.99
F_1 Score	0.99	1.00	0.99	0.99	0.99	0.99
Accuracy	N/A	N/A	0.99	0.99	0.99	0.99

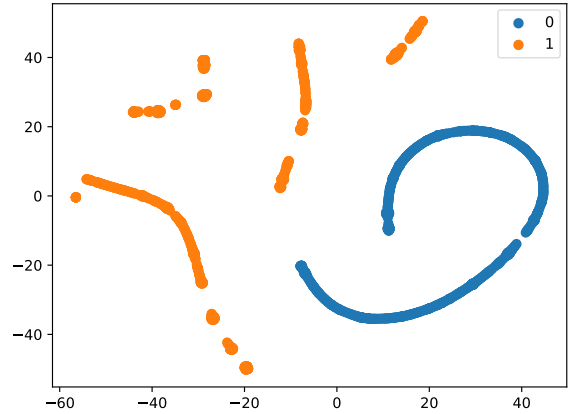


Fig. 7. A 2-D visualization of binary classification results. Labels 0 and 1 correspond to Class 1 (benign path) and Class 2 (malicious path), respectively.

highlights the difference between lateral movement paths and normal paths. Additionally, Figure 5 shows the ROC curve which illustrates the performance of the XGBoost classifier in detecting malicious lateral movement paths. It is evident from the figure that maximizing the area under the ROC curve indicates promising performance achieved by the model in detecting malicious paths.

C. Forward Prediction

In the previous experiment, we explained the training and testing procedures of the proposed model for malicious path detection. The dataset was randomly shuffled and subsequently divided into 70% for training and 30% for testing purposes. We conducted additional experiments to enhance the applicability of the model in real-time scenarios. In other words, the model will exclusively operate in a *forward prediction* manner. In this experiment, we refrained from splitting the dataset into

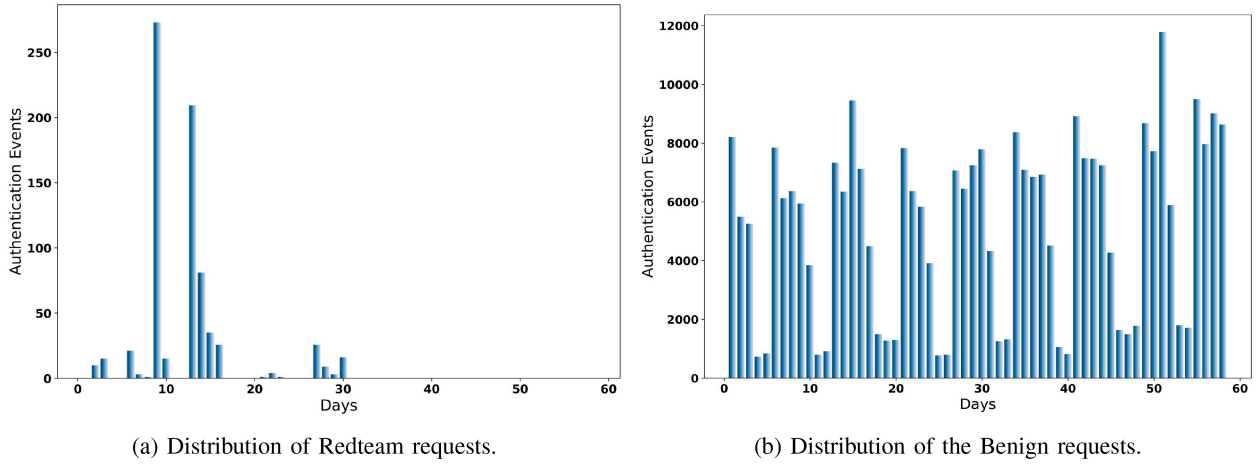


Fig. 8. Distribution of the authentication requests in the LANL dataset.

TABLE VI
CONFUSION MATRIX FOR THE SECOND EXPERIMENT

		Benign paths		Malicious paths	N_t
		Class 1	Class 2		
Benign paths	Class 1	201,417	2		201,419
Malicious paths	Class 2	0	59		59

Note: N_t refers to the number of samples used to test.TABLE VII
PERFORMANCE MEASUREMENTS FOR THE SECOND EXPERIMENT

	Benign paths		Malicious paths	
	Class 1	Class 2	Macro average	Weighted average
Precision	1.00	0.96	0.98	0.99
Recall	1.00	1.00	1.00	1.00
F_1 Score	1.00	0.98	0.99	0.99

TABLE VIII
CONFUSION MATRIX FOR THE THIRD EXPERIMENT

		Benign paths		Malicious paths
		Class 1	Class 2	
Benign paths	Class 1	53,923	0	
Malicious paths	Class 2	0	60	

TABLE IX
PERFORMANCE MEASUREMENTS FOR THE THIRD EXPERIMENT

	Benign paths		Malicious paths	
	Class 1	Class 2	Macro average	Weighted average
Precision	1.00	1.00	1.00	1.00
Recall	1.00	1.00	1.00	1.00
F_1 Score	1.00	1.00	1.00	1.00

two portions. Instead, we trained the model using the sample paths from the initial 10 (or 20) days and proceeded to predict malicious paths for the subsequent days, without performing any *backward prediction*. To assess the performance of the proposed approach in extended prediction scenarios, we conducted four experiments. This section outlines the experiments along with the corresponding results.

In the first experiment, the classifier was trained using authentication records spanning from day 1 to day 30, and subsequently tested on the remaining days (days 31 to 58). The classifier did not identify any malicious authentication requests. In Figure 8, we have illustrated the distribution of authentication requests within the LANL dataset. This includes Red-team requests (lateral movements) displayed in Figure 8(a), and benign requests shown in Figure 8(b). As evident from the Redteam requests graph (Figure 8(a)), the outcome of our proposed model in this experiment aligns accurately with real-world scenarios. In the second experiment, the aim was to reduce the number of training samples while increasing the number of testing samples. The classifier was trained using the authentication records from day 1 to day 20 and subsequently tested with the data from the remaining days (days 21 to 58). Table VI presents the confusion matrix for this experiment. Notably, the table indicates the misclassification of only two normal paths that were erroneously identified as malicious. Table VII also displays the calculated performance metrics, including precision, recall, and F_1 score.

In the third experiment, the aim was to further reduce the number of training samples and increase the number of testing samples. The purpose of conducting this experiment is to evaluate the classifier with an equal number of training and testing samples. Typically, a machine learning-based classifier

is trained using a larger portion (approximately 70 percent) of the dataset patterns and then tested using the remaining smaller portion. In this scenario, the classifier was trained using authentication records from a limited period of 10 days (from day 10 to day 19) and then tested using the subsequent 10 days (days 20 to 29). Table VIII illustrates the confusion matrix for this experiment, showing that there were no misclassified paths. Additionally, Table IX presents the calculated performance metrics, which demonstrate promising results.

In the context of the fourth experiment, we employed the authentication records from the initial 10 days (day 1 to day 10) to train the classifier. As depicted in Figure 8, these 10 days include both normal and malicious paths. The number of malicious paths is significantly lower compared to the normal paths. However, from days 10 to 17, there is an increase in the number of malicious paths compared to normal paths, which could be attributed to weekends or possibly

TABLE X
CONFUSION MATRIX FOR THE FOURTH EXPERIMENT

		Benign paths	Malicious paths
		Class 1	Class 2
Benign paths	Class 1	37113	0
Malicious paths	Class 2	0	351

TABLE XI
PERFORMANCE MEASUREMENTS FOR THE FOURTH EXPERIMENT

	Benign paths		Malicious paths	
	Class 1	Class 2	Macro average	Weighted average
Precision	1.00	1.00	1.00	1.00
Recall	1.00	1.00	1.00	1.00
F_1 Score	1.00	1.00	1.00	1.00

holidays. Therefore, we specifically chose these peak days, which include 351 malicious paths, to evaluate the proposed model's performance on this heterogeneous network. The obtained results, as shown in Tables X and XI, demonstrate that the proposed model accurately classified all paths with outstanding precision, recall, and F_1 score.

D. Discussion and Comparison

In this section, we provide a comparative analysis and discussion of the proposed lateral movement detection system, examining it from two different perspectives as outlined below:

1) *The Impact of Classifier on Lateral Movement Detection Performance:* Regarding the detection technique, Table XII provides an initial comparison between the performance of the XGBoost classifier and various supervised learning techniques, including Support Vector Machine (SVM), K-Nearest Neighbors (K-NN), logistic regression, and Gradient Boosting (GB). In terms of the performance of the proposed system, both execution time and various evaluation metrics, such as overall accuracy, Area Under the Curve (AUC), recall, precision, and F_1 score, have been considered. In this evaluation phase, the performance of the classifier is compared individually with the specified supervised learning techniques, without taking into consideration the complexity of graph embedding processes.

To ensure a fair comparison, we also examined recently published graph learning-based lateral movement detection approaches from the literature and compared their results with our proposed model. Furthermore, in this comparative study, we have exclusively focused on lateral movement detection approaches that have been evaluated using the LANL dataset [19], [20], [25]. As indicated in Table XIII, the study by Fang et al. [13] reported precision, recall, and F_1 score values of 0.86, 0.93, and 0.89, respectively. Additionally, the research conducted by Bian et al. [14] yielded precision, recall, and F_1 score results of 0.97, 0.93, and 0.95, respectively. The performance results displayed in Tables XII and XIII clearly highlight the superiority of our proposed lateral movement detection technique in terms of both efficiency and accuracy.

2) *The Impact of Random Walk Parameters on the Classification Results:* Apart from time and accuracy, which are primarily associated with the performance of the detection technique (classifier), several other parameters tied to the

TABLE XII
THE MODEL'S PERFORMANCE USING VARIOUS MACHINE LEARNING TECHNIQUES

Techniques	Time (s)	Precision (MA)	Recall (MA)	F_1 Score (WA)	F_1 Score (MA)	Accuracy	AUC
XGBoost	234	1.00	0.99	1.00	0.99	0.99	1.00
Logistic Regression	107	0.49	0.50	0.49	0.99	0.99	0.80
K-Nearest Neighbors	2484	0.78	0.60	0.65	0.99	0.99	0.74
Support Vector Machine	629	0.49	0.50	0.49	0.99	0.99	0.88

Note: MA= Macro Average, WA= Weighted Average, AUC= Area under the Roc curve.

TABLE XIII
THE PERFORMANCE OF OUR PROPOSED MODEL COMPARED TO RECENT APPROACHES

Techniques	Time (s)	No. of Walk Walks	Length	Precision (MA)	Recall (MA)	F_1 Score (WA)	F_1 Score (MA)	Accuracy	AUC
Our Approach	234	10	3	1.00	0.99	1.00	0.99	0.99	1.00
[13]	N/R	8	100	0.86	0.93	N/R	0.89	0.91	N/R
[14]	475	N/R	N/R	0.97	0.93	N/R	0.95	N/R	N/R
[12]	N/R	N/R	N/R	0.85	0.90	N/R	0.87	0.91	0.92

Note: MA= Macro Average, WA= Weighted Average, AUC= Area under the Roc curve, N/R= No available result

feature generation process (random walks in node embedding), such as the number of walks and the length of walks, are crucial and directly influence the overall system's performance. Therefore, this aspect should also be taken into consideration. We have incorporated these two parameters into Table XIII to offer an additional comparison related to node embedding. To acquire additional features during random walks over neighboring nodes, we allocated a maximum length of 3 to each walk and instructed the walker to repetitively undergo this process for 10 iterations. We experimented with increasing the number of walks to 100 and the length of walks to 10 for all the supervised learning techniques mentioned earlier. However, the improvement in accuracy was not significant. It's worth noting that in the study conducted by [13], they examined their model with a walk length of 100, yet their F_1 score is still not comparable to the performance achieved by our model.

E. Using the Proposed Approach in Kestrel

We have successfully implemented and released the proposed approach as a Kestrel analytics tool [26]. This tool is accessible through a Python analytics interface. In addition to featuring the XGBoost classifier, this Kestrel analytics tool provides users with a choice among four other classifiers. Users can specify their preferred classifier through an input parameter. The analytics itself requires two input parameters. The first parameter, named *walkLength*, determines the maximum length of the random walks used in the node embedding process. The second parameter, named *classifier*, allows users to select a classifier from the available options: "SVM", "K-NN", "Logistic Regression", "XGBoost", or "Random Forest". If the variable *observations* and the tables *users_obs* and *connections_obs* have been derived, users can apply this Kestrel analytics tool, as illustrated in Figure 9. The proposed Kestrel analytics enhances the variable *observations* by incorporating five additional attributes: *destination*, *source*, *status*, and *user_id*, to all entities. Following these modifications,

```

APPLY docker://detect_lm ON observations, users_obs, connections_obs WITH walkLength=3,
classifier=xgboost

```

Fig. 9. Developing the Kestrel Analytics Tool (implementing the proposed approach through a Python analytics interface) [26].

each entity represents an authentication request, with its status classified as either malicious or benign, denoted by the value of the *status* attribute.

V. CONCLUSION

This research paper introduces a novel technique for identifying malicious lateral movement paths through the use of authentication events and graph learning methods. The technique involves building a graph that represents hosts and users and utilizing DeepWalk to embed nodes. The proposed approach uses features from node and host embedding, along with the time of authentication events, to create feature vectors for each authentication request. These features are then used to train an XGBoost classifier. Furthermore, the model's performance is tested through several experiments in realistic scenarios, which confirms its robustness and predictive ability. The positive results and practical applicability of this model highlight its potential significance in enhancing network defenses against sophisticated threats. Additionally, we have developed this approach into a Kestrel analytics tool accessible via a Python interface to facilitate its application to various datasets and scenarios.

ACKNOWLEDGMENT

The authors express their gratitude to the anonymous reviewers for their valuable feedback.

REFERENCES

- [1] M. Li, W. Huang, Y. Wang, W. Fan, and J. Li, "The study of APT attack stage model," in *Proc. IEEE/ACIS 15th Int. Conf. Comput. Inf. Sci. (ICIS)*, 2016, pp. 1–5.
- [2] A. Niakanlahiji, J. Wei, M. R. Alam, Q. Wang, and B.-T. Chu, "ShadowMove: A stealthy lateral movement strategy," in *Proc. 29th USENIX Security Symp.*, 2020, pp. 559–576. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity20/presentation/niakanlahiji>
- [3] M. Xu, "Understanding graph embedding methods and their applications," *SIAM Rev.*, vol. 63, no. 4, pp. 825–853, 2021.
- [4] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Proc. Web Conf., Compan.*, 2018, pp. 969–976.
- [5] F. Xia et al., "Graph learning: A survey," *IEEE Trans. Artif. Intell.*, vol. 2, no. 2, pp. 109–127, Apr. 2021.
- [6] I. Chami, A. Wolf, D.-C. Juan, F. Sala, S. Ravi, and C. Ré, "Low-dimensional hyperbolic knowledge graph embeddings," 2020, *arXiv:2005.00545*.
- [7] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.
- [8] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2014, pp. 701–710.
- [9] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 855–864.
- [10] M. Chen, Y. Yao, J. Liu, B. Jiang, L. Su, and Z. Lu, "A novel approach for identifying lateral movement attacks based on network embedding," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl., Ubiquitous Comput. Commun., Big Data Cloud Comput., Soc. Comput. Netw., Sustain. Comput. Commun. (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, 2018, pp. 708–715.
- [11] M. Morgan, J. Sexton, J. Neil, A. Ricciardi, and J. Theimer, "Network attacks and the data they affect," in *Dynamic Networks and Cyber-Security*. Singapore: World Sci., 2016, pp. 1–36.
- [12] S. Zhao, R. Wei, L. Cai, A. Yu, and D. Meng, "CTLMD: Continuous-temporal lateral movement detection using graph embedding," in *Proc. 21st Int. Conf. Inf. Commun. Secur. (ICICS)*, 2020, pp. 181–196.
- [13] Y. Fang, C. Wang, Z. Fang, and C. Huang, "LMTracker: Lateral movement path detection based on heterogeneous graph embedding," *Neurocomputing*, vol. 474, pp. 37–47, Feb. 2022.
- [14] H. Bian, T. Bai, M. A. Salahuddin, N. Limam, A. Abou Daya, and R. Boutaba, "Uncovering lateral movement using authentication logs," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 1049–1063, Mar. 2021.
- [15] I. J. King and H. H. Huang, "Euler: Detecting network lateral movement via scalable temporal link prediction," *ACM Trans. Privacy Security*, vol. 26, no. 3, pp. 1–36, 2023.
- [16] B. Bowman, C. Laprade, Y. Ji, and H. H. Huang, "Detecting lateral movement in enterprise computer networks with unsupervised graph AI," in *Proc. 23rd Int. Symp. Res. Attacks, Intrus. Defenses (RAID)*, 2020, pp. 257–268.
- [17] D. Kushwaha et al., "Lateral movement detection using user behavioral analysis," 2022, *arXiv:2208.13524*.
- [18] Q. Liu et al., "Latte: Large-scale lateral movement detection," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, 2018, pp. 1–6.
- [19] A. D. Kent, "Cyber security data sources for dynamic network research," in *Dynamic Netw. Cybersecurity*. London, U.K.: Imperial College Press, 2015.
- [20] A. D. Kent, *Comprehensive, Multi-Source Cyber-Security Events*, Los Alamos Nat. Lab., Los Alamos, NM, USA, 2015.
- [21] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2702–2711.
- [22] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.
- [23] X. Rong, "Word2vec parameter learning explained," 2014, *arXiv:1411.2738*.
- [24] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 785–794.
- [25] T. Bai, H. Bian, A. Abou Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "A machine learning approach for RDP-based lateral movement detection," in *Proc. IEEE 44th Conf. Local Comput. Netw. (LCN)*, 2019, pp. 242–245.
- [26] L. Rashidi, "Graph learning-based lateral movement detection," 2022. [Online]. Available: <https://github.com/opencybersecurityalliance/kestrel-analytics/tree/release/analytics/Graph/%20Learning-based/%20Lateral/%20Movement/%20Detection>



Mahdi Rabbani received the M.Tech. degree in artificial intelligence from the University of Mysore, India, and the Doctoral degree in computer science and technology from the Nanjing University of Science and Technology, China, in 2022. He is a Postdoctoral Fellow with the Canadian Institute for Cyber Security, University of New Brunswick, Fredericton, NB, Canada. Prior to his current role, he served as a Visiting Researcher with Dalhousie University, focusing on research related to machine learning and deep learning approaches for detecting Distributed Denial of Service attacks in Internet of Things devices. Throughout his career, he has contributed to the field of cybersecurity, with numerous publications in various journals. His research interests encompass AI for cybersecurity, malware analysis, machine learning, and deep learning-based network anomaly detection and recognition, as well as privacy-enhancing techniques for IoT devices.



Leila Rashidi received the Ph.D. degree from the Sharif University of Technology, Tehran, Iran. She is a Senior Researcher with Huawei Technologies Company, Ottawa, ON, Canada. Before joining Huawei, she was a Postdoctoral Fellow with the Canadian Institute for Cybersecurity, Fredericton, New Brunswick, Canada, from February 2022 to July 2023. Furthermore, she was a Postdoctoral Associate with the University of Calgary from January 2020 to December 2021. Her research interests are data center networks, security, access

control, and performance modeling.



Ali A. Ghorbani (Senior Member, IEEE) has held a variety of positions in academia for the past 37 years and is currently a Professor of Computer Science, the Tier 1 Canada Research Chair of Cybersecurity, and the Director of the Canadian Institute for Cybersecurity, which he established in 2016. He served as the Dean of the Faculty of Computer Science, University of New Brunswick from 2008 to 2017 and spent over 27 years of his 37-year academic career. He is also the Founding Director of the Laboratory for Intelligence and

Adaptive Systems Research. He is the co-founder of the Privacy, Security, Trust Network in Canada and its international annual conference and the co-inventor of three awarded patents and has published over 260 peer-reviewed articles during his career. He has supervised over 170 research associates, postdoctoral fellows, graduate, and undergraduate students during his career. His book, *Intrusion Detection and Prevention Systems: Concepts and Techniques* (Springer, October 2010). His current research focus is cybersecurity, Web intelligence, and critical infrastructure protection. He was twice one of the three finalists for the Special Recognition Award at the 2013 and 2016 New Brunswick KIRA Award for the knowledge industry. He is the recipient of the Startup Canada Senior Entrepreneur Award in 2017. He served as the Co-Editor-In-Chief of *Computational Intelligence* from 2007 to 2017. He co-founded two startups, Sentrant Security and EyesOver Technologies in 2013 and 2015, respectively.