Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 7502294, 10 pages
https://doi.org/10.1155/2022/7502294

Research Article

Detecting Unknown Threat Based on Continuous-Time Dynamic Heterogeneous Graph Network

Peng Gao,[1,2] Weiyong Yang,[1,2,3] Haotian Zhang,[1,2] Xingshen Wei,[1,2] Hao Huang,[3] Wang Luo,[1,2] Zhimin Guo,[4] and Yunhe Hao[1,2]

[1]*NARI Group Corporation/State Grid Electric Power Research Institute, Nanjing 21003, China*
[2]*Nanjing NARI Information & Communication Technology Co., Ltd., Nanjing 21003, China*
[3]*Nanjing University, Nanjing 210008, China*
[4]*State Grid Henan Electric Power Research Institute, Zhengzhou 450000, China*

Correspondence should be addressed to Peng Gao; gao.itslab@gmail.com

Received 1 June 2022; Revised 26 June 2022; Accepted 8 July 2022; Published 25 July 2022

Academic Editor: Kuruva Lakshmanna

Unknown threats have caused severe damage in critical infrastructures. To solve this issue, the graph-based methods have been proposed because of their ability for learning complex interaction patterns of network entities with discrete graph snapshots. However, such methods are challenged by the computer networking model characterized by the natural continuous-time dynamic heterogeneous graph (CDHG). In this paper, we propose a CDHG-based graph neural network model, namely, CDHGN, for unknown threat detection. It first constructs the CDHG using interaction relationships among network entities extracted from various log records. Then, it trains the detection model based on a heterogeneous attention network and performs streaming detection for live online network events. We implement a prototype and conduct extensive experiments on a comprehensive cybersecurity dataset with more than nine million records. Experimental result shows that the proposed method can achieve superior detection performance than the state-of-the-art methods.

1. Introduction

The unknown threats of cyberspace represented by advanced persistent threat (APT) attacks have become a worldwide security problem. In the past few years, the national power grids of Ukraine and Venezuela were attacked by unknown threats and caused large-scale power outages. In the meanwhile, the industrial control systems (ICS) gradually changed from isolated and static to open and interactive. Worryingly, thousands of significant cyber incidents caused by such cyberattack occur every year around the world and have caused serious political, economic, and social adverse impacts [1].

In ICS environments, there are a large number of real-time interactive IoT entities, such as mobile phones/tablets, drones, hosts, and other devices. Unfortunately, it is hard for the existing methods to detect malicious behaviours of unknown threat in the continuous interactions among massive heterogeneous entities. To solve this issue, we aim to develop a model to accurately detect the unknown threat by capture latent features of heterogeneous entities and their continuously ongoing interactions. Heterogeneous graphs are often used to model complex systems and interactions where entities of different types interact with each other in different ways. Recently, detecting methods based on heterogeneous graph neural network have been developed to trace the malicious behaviours of unknown threat in the network. Log2vec [2] takes log records as graph nodes (entities) and heuristically defines a set of rules for generating graph edges (interactions). Then, node embeddings are calculated and clustered to recognize anomaly behaviours. However, its graph construction approach will lead to two problems: (1) it cannot fully express the context of a log entry, for instance, cannot trace or predict when who did what to whom. (2) It is not an end-to-end approach and thus cannot universally construct graphs for massive and heterogeneous behaviours in computer networks. Thus, the detection performance still needs to be improved.

To overcome these limitations and challenges, we propose a new detection method based on continuous-time dynamic heterogeneous graph network (CDHGN). The CDHG is constructed in a way that directly corresponds to the entities (nodes) and their interactions (edges) in the network. The CDHGN model can be seen as an encoder-decoder architecture, where the encoder is a function that maps a CDHG to node embeddings, and the decoder takes the node embeddings as input and makes anomaly classifications. More specifically, the CDHGN first transforms log data that carry behaviour information of network entities into a CDHG. Next, the CDHG is input into a time-encoding module to obtain the embedding representation of the time factor. Then, the CDHG incorporating the time embedding is input into heterogeneous attention networks for generating embeddings of both edges and nodes where edge embeddings representing various interactions among entities in network are obtained through an improved attention mechanism. Finally, the embedding representations of entities and their interactions are sent to the decoder that consists of linear layer and *SoftMax* layer to determine whether there are anomalous interactions, namely, malicious behaviours. The contributions of our work are summarized as follows:

(i) We propose a new unknown threat detection model based on continuous-time dynamic heterogeneous graph network (CDHGN). It captures spatiotemporal contextual information of network entities and their interactions by an end-to-end approach. The continuous-time dynamic heterogeneous graph (CDHG) can be naturally constructed by raw behaviour data generated by network entities

(ii) According to the complex behaviour characteristics of unknown threat, we improve the message passing and aggregation function of the attention block of CDHGN by assigning different weight matrices to different types of node features and edge features to obtain the optimal node embedding representation

(iii) We develop a prototype and conduct various experiments on a real-world cybersecurity dataset. Experimental results show that by introducing heterogeneous edge embedding, the proposed CDHGN outperforms previous state-of-the-art methods

The rest of this paper is organized as follows. Section 2 address the related work. Section 3 describes the design and implementation of our approach. Experiments and performance evaluation are presented in Section 4. Section 5 concludes the paper.

2. Related Work

A lot of works have been addressed in intrusion detection system (IDS) to detect unknown threat, which generally can be divided into two approaches: misuse detection and anomaly detection. Misuse detection approaches first define abnormal entities' behaviour and then defines all other behaviour as normal. Specifically, it generates and stores signatures of cyberattack behaviours in advance. Then, it monitors various network data such as network traffics, user operations, and process interactions. If a behaviour pattern is matched with an attack signature, it is detected as a malicious behaviour [3]. For instance, the network IDS Snort exploits various rules based on expert experience, which cover detection rules of timestamp, detector ID, IP address, port number, alarm type, alarm priority, TCP flag, protocol type, etc. [4]. Such signature-based methods are efficient and accurate, however cannot detect unknown behaviours.

Anomaly detection approaches learn potential associations in historical data through machine learning models. The baseline model is trained by learning behaviours of network entities. Whether a new behaviour is malicious or not is determined by the learned classifier [5–7]. In [8], researchers combined meta-learning and ensemble learning ideas to enforce anomaly detection. First, the basic detectors with poor performance are preeliminated by the static selection process based on the isolation forest. Then, the screened basic detectors are dynamically selected and integrated to mitigate the inaccurate detection problem. This kind of methods have certain generalization ability to detect unknown attack behaviours, while it usually needs a careful designed feature engineering process based on security experts' knowledge and experience. Moreover, when facing complex behaviour patterns, they were often challenged by the high false positives because a very small percentage of abnormal behaviour data is easily drowned in the massive normal behaviour data as noise. In recent years, toward massive heterogeneous network security data, deep learning-based end-to-end methods that do not need assistances of feature engineering process have received more attention. In [9], authors used long short-term memory (LSTM) network to extract latent representation in time series and then use multilayer fully connected network to match feature to complete the abnormal behaviour recognition. In [10], authors constructed a neural network model to learn feature vectors of user behaviours which are then clustered to detect whether there exist internal attack events. In [11], researchers analysed multidimensional interaction patterns between users and entities, such as email communication, web browsing, and server logins. In [12], researchers developed a traceability system, from the perspective of the calling relationship between hosts and processes, to detect anomalies by monitoring the calling relationship. Some provenance tracking systems are proposed to monitor and analyse the activities of the system [13–16]. However, the long time interval and low-profile characteristics of complex behaviours of APT kind of unknown attacks make them still face the low detecting success rate problem. Another difference between our approach and them is that we focus on analysing logs that record user behaviour in computer networks which extract relationships among log data that reflect typical behaviours of users.

The graph-based deep learning approaches have been developed to capture complex behaviour patterns in networks [2, 17]. The directed dynamic graph is used to describe operations (e.g., logon) between subjects (e.g., user) and objects (e.g., PC). Accordingly, multidimension relationships (e.g., causality) among network entities are naturally represented by the graph. Log2vec [2] developed ten heuristic rules (edges) from causality (sequence), logical (time), and interaction (interhost) aspects to connect log records (nodes) to construct the network behaviour graph and generate the corresponding graph embeddings. By employing clustering algorithm on node embeddings of the generated graph, Log2vec draws a boundary between malicious and benign behaviours which belong to different clusters. As previously mentioned, its ability of capturing contextual information of network entities' behaviours is still limited. RShield [18] presented a detection method for complex multistep attack based on the temporal graph network (TGN) model [19]. However, it can only deal with homogeneous graph network. In [20], researchers proposed an ensemble detection method using graph-based modelling of the security state of the target system and correlation of diverse indicators of anomalous host behaviour, which needs to manually extract domain-specific features. Some graph representation learning techniques also have been utilized to analyse dynamic graphs [21–24], while most of them are not applicable to unknown threats detection, because they cannot fully characterize the properties of computer networks where interactions between entities are continuous-time based. Hindroid [25] modelled Android applications, related APIs, and their relationships as heterogeneous networks. However, it measured the similarity of Android applications based on different metapaths which requires preparing man-made metapaths carefully in advance.

3. Continuous-Time Dynamic Heterogeneous Graph Network

In this section, we begin with a brief overview of continuous-time dynamic heterogeneous graph network (CDHGN). A detailed explanation of each system component will be introduced in the following parts.

3.1. Architecture Overview. An end-to-end anomaly detection model based on dynamic heterogeneous graph neural network is developed to simultaneously detect intrusions on a collection of networked hosts. The schematic diagram of the method is shown in Figure 1.

First, a heterogeneous graph is constructed based on event logs of entities' behaviours. Second, the time embedding of each behaviour is acquired. Third, the graph incorporating the time encoding is fed into the dynamic heterogeneous graph neural network to generate the optimal behaviour embedding representation. Finally, the behaviour embedding represented by corresponding nodes and edge is used to classify normal and abnormal behaviour.

3.2. Continuous-Time Dynamic Heterogeneous Graph Construction. Heterogeneous graphs are an important abstraction for modelling complex interactive relational data in the real world. We denote the edge *e* connecting the source node *src* and the destination node *dst* as $e = (src, dst)$. Its metarelationship can be represented as a triple such as <*src*, *e*, *dst*>. To better model heterogeneous computer networks, we assume that there may exist multiple relationships among entities. For instance, the relationship between *user* and *pc* can be divided into various types according to the operation type, such as *user logon pc* and *user insert-usb pc*.

As shown in Figure 2, in order to show continuous-time dynamic characteristics of data, we assign a unique timestamp *t* to each edge $e = (src, dst)$, indicating that there is a connection between the source node *src* and the destination node *dst* at time *t*. For instance, when a user (*User123*) logon a PC (*PC456*) at time *t*, the time *t* will be assigned to the edge between the user and pc. Further, the pc node *PC456* can be assigned multiple timestamps: *PC456@9* am denotes that the *PC456* performed an operation at 9 : 00 in the morning, which probably means that the employee just turned on the computer at the workstation in the morning, and *PC456@8* pm indicates that the *PC456* performed an operation at 8 : 00 in the evening, which probably means that the employee turns off the computer after work. In order to generate data for constructing CDHG, we need to parse log records (events) field by field and extract the metadata. The format of an event metadata is defined as (*src_id*, *dst_id*, *src_type*, *dst_type*, *src_feature*, *dst_feature*, *edge_type*, *edge_attr*, *t*). Here, let *src_id* and *dst_id* be the identity of source node and destination node, respectively; *src_type* and *dst_type* the type of source node and destination node, respectively; *src_feature* and *dst_feature* the node features of source node and destination node, respectively; *edge_type* the connection type between source node and destination node; *edge_attr* the edge features; and *t* the timestamp of the event. As far as we know, there have been some research results in complex network attack modelling methods for unknown threats, such as sequence-based and static isomorphic graphs, while anomaly detection for continuous-time dynamic heterogeneous graphs has not been yet explored.

3.3. Time-Encoding Function. Traditional graph construction methods build a static graph in each time slot. However, this may lose some structural dependencies between different time periods. At the same time, edge connections occurring at one moment may affect node representation at another moment. Therefore, in order to make better use of the time characteristics contained in the CDHG and allow nodes and edges to better carry the information contained in different time points in the aggregation stage, we develop a relative dynamic encoding mechanism. Specifically, given a source node *src* with timestamp *ts* and a destination node *dst* with timestamps *td*, their time representations are denoted by $T(src@ts)$ and $T(dst@td)$, respectively. We compute the relative time interval by $\Delta T(dst@ts, src@td) = T(dst@ts) - T(src@td)$, which is denoted by ΔT for brevity. Then, we will feed the ΔT into the time-encoding function to get the corresponding time embedding.

We use the sinusoid function as time-encoding function to calculate the time embedding over an interval. Because the periodic variation law of the sin/cos function is stable, the encoding function has certain invariance. The simple function can take the form as Equation (1), where ΔT is

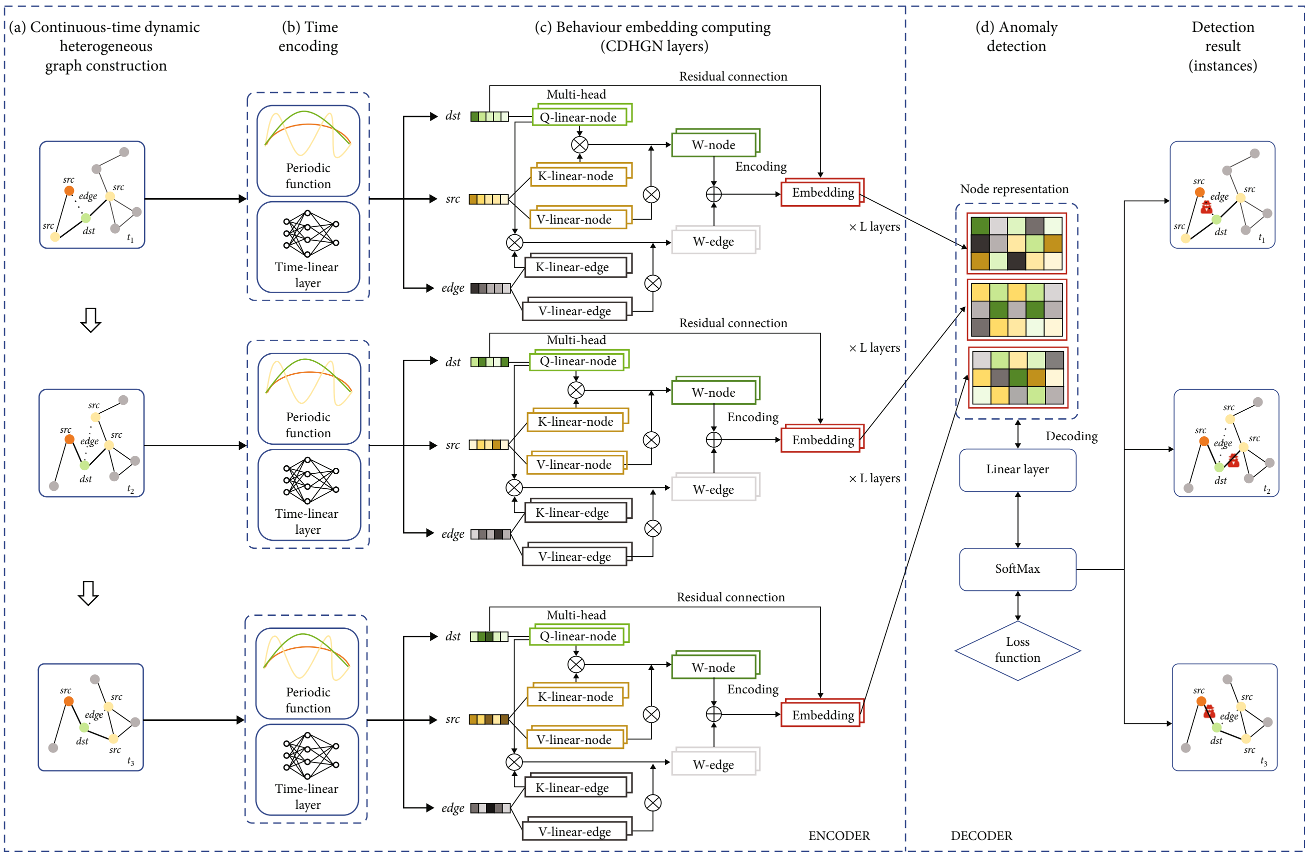

FIGURE 1: The schematic diagram of spatiotemporal graph network based on CTDHG. (a) Continuous-time dynamic heterogeneous graph construction. The green node is the destination node, and the yellow and orange are different types of source nodes connected to the destination node. The solid and dashed lines represent two different types of edge connections. (b) Input the graph data into periodic and time linear function to obtain time embedding. (c) The data integrating graph data and time embedding is input into the behaviour embedding computing layers (CDHGN layers): (i) the destination node is mapped through Q-linear-node. (ii) The node features of the source node are mapped with K-linear-node and V-linear-node, respectively. (iii) The edge features of the source node are mapped with K-linear-edge and V-linear-edge, respectively. Assign different weights according to different node types and edge types. Use the self-attention mechanism to learn the adjacency information that contributes the most to downstream tasks, and aggregate the adjacency information to obtain the best embedding representation. (d) Input the embedding representation into the anomaly detection module in decoder. In the training phase, backpropagation is performed through the loss function to update the model parameters; in the testing phase, the anomaly detection results are obtained through the linear layer and *SoftMax* layer.

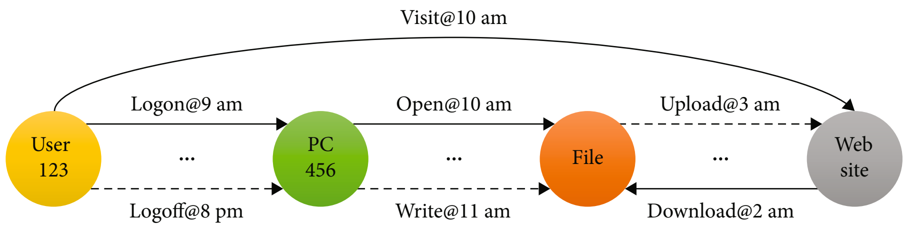

FIGURE 2: An instance of continuous-time dynamic heterogeneous graph. Different colours represent different types of nodes. Solid and dotted lines represent different types of edges.

the position and dim is the dimension, which means each dimension of the positional encoding corresponds to a sinusoid.

$$\mathrm{RT}(\Delta T, \mathrm{dim}) = \sin\left(\frac{\Delta T}{\mathrm{dim}}\right), \tag{1}$$

since the mapping interval of this approach is only within [-1, 1], which results in a limited range of spatial representation. Thus, we need to compute temporal encoding with different functions in different dimensions. Inspired by the *Transformer* [26], the relative dynamic encoding mechanism is presented as follows.

$$\mathrm{RT}(\Delta T, 2\,\mathrm{dim}) = \sin\left(\frac{\Delta T}{10000^{(2\,\mathrm{dim}/d)}}\right), \tag{2}$$

$$RT(\Delta T, 2\dim + 1) = \cos\left(\frac{\Delta T}{10000^{((2\dim+1)/d)}}\right). \tag{3}$$

In Equations (2) and (3), d represents the dimension of the vector. The wavelengths form a geometric progression from 2π to $10000{\cdot}2\pi$. Then, we use a linear function in Equation (4) to get the final time-encoding function, where $\mathrm{RT}(\Delta T)$ is the general term for Equations (2) and (3), and TimeLinear is a simple learnable linear projection function.

$$\mathrm{RTE}(\Delta T) = \mathrm{TimeLinear}(\mathrm{RT}(\Delta T)). \tag{4}$$

Finally, the relative time embedding information generated by the RTE function is to be added to the feature representation (presented in Section 3.4) of a source node to capture the relative dynamic information between the source node and the destination node.

3.4. Heterogeneous Behaviour Embedding and Anomaly Detection. The behaviour embeddings are computed to anomaly edge prediction. Embedding representation of each edge mainly includes three basic operators calculating: (1) *Attention*, which calculates the importance of the source nodes connected to each different edge; (2) *Message*, which extracts the information of source nodes and edges; and (3) *Aggregate*, which passes the neighbour information of the destination node through aggregation of attention weight coefficients.

Inspired by the *Transformer* model [26], the destination node *dst* is mapped to a *Query* vector, and the source node *src* is mapped to a *Key* vector. In the unknown complex attack detection task, in order to better exploit the embedding information contained in edges, we calculate node features and edge features' attention score, respectively. In order to maximize parameter sharing while still maintaining the uniqueness among different relationships, we propose to use corresponding parameter matrices for different types of nodes and edges. The attention score computing mechanism is presented as Equations (5)–(11):

$$\mathrm{Attention}(\mathrm{src}, \mathrm{dst}) = \underset{\forall \mathrm{src}\in N(dst)}{\mathrm{Softmax}}\left(\underset{i\in[1,m]}{\|} A_{\mathrm{head}}{}^{i}(\mathrm{src}, \mathrm{dst})\right), \tag{5}$$

$$A_{\mathrm{head}}{}^{i}(\mathrm{src}, \mathrm{dst}) = \frac{\left(K^i(\mathrm{src}_n) Q^i(\mathrm{dst}_n)\right)}{\sqrt{d}}, \tag{6}$$

$$K^i(\mathrm{src}_n) = K\text{-linear-node}^i\left(H^{(l-1)}[\mathrm{src}]\right), \tag{7}$$

$$Q^i(\mathrm{dst}_n) = Q\text{-linear-node}^i\left(H^{(l-1)}[\mathrm{dst}]\right), \tag{8}$$

$$\mathrm{Attention}(\mathrm{src}, \mathrm{e}, \mathrm{dst}) = \underset{\forall \mathrm{src}\in N(\mathrm{dst})}{\mathrm{Softmax}}\left(\underset{i\in[1,m]}{\|} A_{\mathrm{head}}{}^{i}(\mathrm{src}, e, \mathrm{dst})\right), \tag{9}$$

$$A_{\mathrm{head}}{}^{i}(\mathrm{src}, e, \mathrm{dst}) = \frac{\left(K^i(e) W_e^A Q^i(\mathrm{dst}_n)\right)}{\sqrt{d}}, \tag{10}$$

$$K^i(\mathrm{e}) = K\text{-linear-edge}^i\left(H^{(l-1)}[e]\right). \tag{11}$$

First, for the i-th node attention head $A_{\mathrm{head}}{}^{i}(src, dst)$, we need to calculate the similarity between the *Query* vector and the *Key* vector. We input the destination node features and the source node features into Qlinearnode and Klinearnode, respectively, for feature mapping. Then, we calculate the dot product of the $K^i(\mathrm{src}_n)$ and the $Q^i(\mathrm{dst}_n)$. Since there are different connecting edges between different node pairs, unlike the original Transformer which directly calculates the dot product of the $K^i(e)$ vector and the $Q^i(\mathrm{dst}_n)$ vector, we assign a specific weight matrix W_e^A to each edge type. In this way, the model can capture different semantic information contained in node pairs. Finally, for the destination node *dst*, we aggregate the attention scores of all adjacent nodes *N(dst)* into the *SoftMax* layer to normalize the attention scores, where m denotes the total number of multiheads.

Next, in the message passing stage, the information of the source node is passed to the destination node while computing the heterogeneous mutual attention as shown in Equations (12)–(15). Similar to the attention computing process, different weight matrices are set according to different edge types in the message passing process to alleviate the distribution differences of different types of nodes and edges.

$$\mathrm{Message}(\mathrm{src}, \mathrm{dst}) = \underset{i\in[1,m]}{\|} M^i_{\mathrm{head-node}}, \tag{12}$$

$$M^i_{\mathrm{head-node}} = V\text{-linear-node}^i\left(H^{(l-1)}[\mathrm{src}]\right) W_n^M, \tag{13}$$

$$\mathrm{Message}\ (\mathrm{src}, \mathrm{e}, \mathrm{dst}) = \underset{i\in[1,m]}{\|} M^i_{\mathrm{head-edge}}, \tag{14}$$

$$M^i_{\mathrm{head-edge}} = V\text{-linear-edge}^i\left(H^{(l-1)}[e]\right) W_e^M. \tag{15}$$

For the i-th node message head $M^i_{\mathrm{head-node}}$, we project the source node features $H^{(l-1)}[\mathrm{src}]$ into the i-th message vector with a linear projection Vlinearnode^i. For the i-th edge message head $M^i_{\mathrm{head-edge}}$, we project the edge features $H^{(l-1)}[e]$ into the i-th message vector $M^i_{\mathrm{head-edge}}$ with a linear projection Vlinearedge^i. We use a matrix W_e^M to incorporate the edge dependency and assign different message weight matrices W_n^M to different node types.

Finally, in the aggregation stage, the information of the source node and the destination node is aggregated according to different edge connection relationships as shown in Equation (16). Since the sum of attention has been normalized using *SoftMax* before, the attention score can be directly used as a weight to equalize the embedding representation of nodes whose neighbour node information has been updated:

$$H^l[\mathrm{dst}] = \underset{\forall \mathrm{src}\in N(\mathrm{dst})}{\oplus}(\mathrm{Attention}(\mathrm{src}, \mathrm{dst}) \cdot \mathrm{Message}(\mathrm{src}, \mathrm{dst}), \mathrm{Attention}(\mathrm{src}, \mathrm{e}, \mathrm{dst}) \cdot \mathrm{Message}(\mathrm{src}, \mathrm{e}, \mathrm{dst})). \tag{16}$$

```
Input: $src$, $dst$, $e$, $src$ node type, $dst$ node type, $edge$ type
Output: $dst$'s node embedding $H^L[dst]$
for $l \leftarrow 1$ to $L$ do
  for $i \leftarrow 1$ to $m$ do
    for $\forall src \in N(dst)$ do
      for $src$ node type, $dst$ node type do
        calculate node attention score ${A_{\text{head}}}^i(src, dst)$.
        extract node $src$'s feature $M^i_{\text{head-node}}(src)$ and assign/update $W^M_n$
      end for
      for $edge$ type
        calculate edge attention score ${A_{\text{head}}}^i(src, e, dst)$ and assign/update $W^A_e$
        extract edge $e$'s feature $M^i_{\text{head-edge}}(e)$ and assign/update $W^M_e$
      end for
    end for
  end for
  concat all node attention $Attention\,(src, dst\,)$ and feature $Message(src)$ respectively
  concat all edge attention $Attention\,(src, e, dst)$ and feature $Message\,(e)$ respectively
  Aggregate $Attention$ and $Message$ by dot product to get $H^l[dst]$
  $dst \leftarrow H^l[dst]$
end for
return $H^L[dst]$
```

Algorithm 1: Heterogeneous behaviour embedding.

For brevity, we give the pseudocode of our algorithm as shown in Algorithm 1, which outlines the heterogeneous behaviour embedding algorithm for a node through L CDHGN layers. Let l be the l -th CDHGN layer, L the total number of CDHGN layers, i the i-th multihead, m the total number of multihead, $\forall src \in N(\text{dst})$ all the source nodes which connected to the destination node, ${A_{\text{head}}}^i(\text{src}, \text{dst})$ the i-th node attention head, $M^i_{\text{head-node}}(\text{src})$ the i-th node message head, ${A_{\text{head}}}^i(\text{src}, e, \text{dst})$ the i-th edge attention head, $M^i_{\text{head-edge}}(e)$ the i-th edge message head, $\text{Attention}(\text{src}, \text{dst})$ the attention score of node, $\text{Message}(\text{src})$ the feature representation extracted from the node, $\text{Attention}(\text{src}, e, \text{dst})$ the attention score of edge, $\text{Message}\,(e)$ the feature representation extracted from the edge, $H^l[\text{dst}]$ the *dst*'s node embedding in the l-th CDHGN layer, and $H^L[\text{dst}]$ the final *dst*'s node embedding after L rounds.

Most graph neural networks focus on fetching the embedded representation of nodes. However, the complex unknown threat detection task relies on the relationship of edges to determine whether it is an attack behaviour. To this end, we splice the embedding representations of the nodes on both sides of the edge to obtain the embedding representation of the edge according to the type of the edge. Then, we use the fully connected layer to map the embedding representation of the edge back to its unique distribution and finally rely on the *SoftMax* layer to get the probability that a connected edge belongs to an attack event. As shown in Equation (17), in the decoder module, we use the cross-entropy loss function for backward propagation:

$$\text{loss}\,(x, \text{class}) = -\log\left(\frac{\exp\,(x[\text{class}])}{\sum_j \exp(x[j])}\right), \tag{17}$$

where x represents embedding representation, j represents the index of elements of x, and class represent the ground truth index of the x.

4. Experiment and Evaluation

4.1. Experiment Setup. In this section, we set up experiments to verify the research question that compared with state-of-the-art baselines: does the proposed method improve the detection performance of unknown threat?

A real-world cybersecurity dataset is used in the experiment: Los Alamos National Lab's (LANL's) comprehensive cybersecurity events dataset [27]. The LANL dataset represents 58 consecutive days of event data collected from four sources (authentication, process, network flow, DNS, and *redteam*) within LANL's internal computer network. The authentication events of LANL dataset includes 1,648,275,307 log records collected over 58 days for 12,425 users and 17,684 computers, within LANL's corporate, internal computer network. The *redteam* data presents specific events taken from the authentication data that present known compromise events, which is used as ground truth of abnormal behaviour that is different from normal users' behaviours. We use the authentication data to form the continuous-time dynamic heterogeneous graph for detecting malicious logs. We randomly selected a subset of the LANL dataset which contains 9,918,928 total edges generated from 10,895 nodes (user host pairs), and all 691 malicious interactions are generated by 104 users.

To answer the research question, we compare our method with the state-of-the-art methods on the same dataset. Baselines are Tiresias [28], ensemble method [20], and Log2vec [2], which cover both supervised and unsupervised methods. Tiresias is an advanced log-entry-level supervised approach on anomaly detection for security event prediction in various events with noise. Ensemble method and Log2vec are unsupervised approach to separate malicious and benign activities into different clusters and identify malicious ones.

We adopt the area under the curve (AUC) as the main performance metric. It is used as a summary of the receiver characteristic operator (ROC) curve, which is insensitive to the imbalance of dataset. An AUC value reaches its best value at 1 and worst at 0. The higher the AUC, the better the performance of the model at predicting between the positive and negative classes.

4.2. Evaluation. To prove the effectiveness of the proposed method CDHGN, the LANL dataset is divided into training set, validation set, and test set. The CDHGN model is trained with the training set. The specific computing environment and parameters are as follows: the training and testing platform is a workstation with GV100 graphics card, and the model optimizer uses Adam algorithm [29], the learning rate is set to 0.01, and the model is iteratively trained for 300 rounds. Experimental results are shown in the following tables. In Table 1, we summarize the AUC results of baseline methods and our method in LANL dataset. In Table 2, we present the detection result of our method on different division of train, validation, and test sets on the dataset. The result includes specific percentage of division, number of malicious logs in training, validation, and testing phases and their corresponding AUC value. In Table 3, we show the detection result of the proposed method only based on the quantity of the abnormal entries.

In Table 1, we see that the CDHGN method performs better than baseline methods. CDHGN increased the AUC by 14.1%, 9.0%, and 6.6%, compared to Tiresias, ensemble method, and Log2vec, respectively. In Table 2, we see that, on the one hand, when more data is used for training, that is, when the training set, validation set, and test set are divided according to 0.8 : 0.1 : 0.1, the AUC value can reach 0.9844. On the other hand, when less data is used for training, that is, when the training set, validation set, and test set are divided according to 0.22 : 0.04 : 0.74, the AUC can still reach 0.9121. In Table 3, we divide the dataset into training set, validation set, and test set only based on the quantity of the abnormal entries. We can see that the proposed CDHGN has a better detection effect than baselines, while it does not need a large number of training samples.

We conduct an experimental analysis of the hyperparameters in the CDHGN network structure as illustrated in Figures 3–5, whose experiment results are shown in Tables 4–6. There are three hyperparameters involved as follows.

(i) Whether to Use Layer Normalization (CDHGN structure, Figure 3 and Table 4)

(ii) The number of Attention Heads in a CDHGN Layer (#Multihead, Figure 4 and Table 5)

(iii) The number of CDHGN Convolution Layer (#CDHGNLayer, Figure 5 and Table 6)

From the three results of comparison experiments shown in Tables 4–6, for the LANL dataset, the CDHGN network can effectively identify anomaly behaviours. (1) After adding the *LayerNorm* layer to normalize layer data, the influence of the dimension between the data is eliminated, and the recognition effect can be effectively improved by about 2.8 points. (2) When the number of attention head is increased, there are slight fluctuations in the recognition results. It is considered that because the number of feature dimensions in the LANL dataset is not large, the 2-head attention has been able to extract the effective features of the data. (3) When the number of CDHGN convolution layer is increased, the detection effect has little effect, indicating that the shallow network structure has been able to learn effective features.

Since the LANL dataset in the experiment is a mature dataset that have been widely used, which also were used by baseline methods for their experiments. Therefore, we consider that the experiment we conducted on the datasets is sufficiently generalized to keep external validity.

4.3. Case Study. With the development of the Energy Internet and the wide application of advanced information and communication technologies, such as Internet+, in the power grid, the power system has gradually broken the previous closed and proprietary boundary. We developed a CDHGN prototype to monitor unknown anomaly behaviour in a power information network.

Table 1: Detection result of different methods (Table 1 is reproduced from Yang et al., 2022).

Approach	AUC
Tiresias	0.85
Ensemble	0.89
Log2vec	0.91
CDHGN (proposed)	0.97

Table 2: Detection result on different dataset divisions of CDHGN.

Dataset division	#Train	#Val	#Test	AUC (test)
0.8 : 0.1 : 0.1	1673	209	209	0.9844
0.7 : 0.1 : 0.2	1463	209	419	0.9787
0.6 : 0.2 : 0.2	1253	419	419	0.9714
0.5 : 0.3 : 0.2	1045	627	419	0.9569
0.27 : 0.03 : 0.7	565	63	1463	0.9344
0.22 : 0.04 : 0.74	460	84	1547	0.9121
0.258 : 0.002 : 0.74	539	5	1547	0.9412

Table 3: Detection result on different number of the abnormal samples of CDHGN.

Dataset division	Abnormal sample			AUC (test)
	#Train	#Val	#Test	
0.27 : 0.03 : 0.7	187	21	483	0.9366
0.22 : 0.04 : 0.74	152	28	511	0.9400
0.258 : 0.002 : 0.74	179	1	511	0.9400

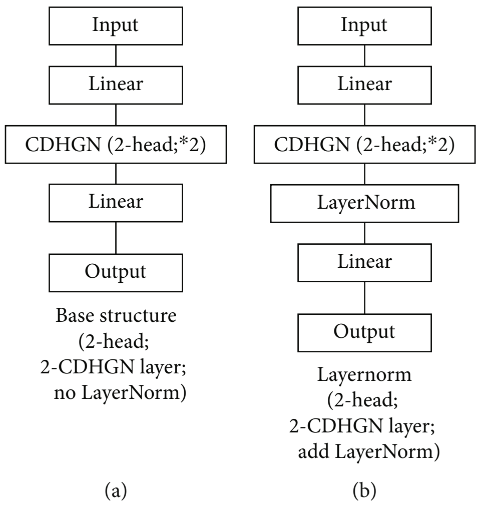

FIGURE 3: Abstract structure of Base and LayerNorm of CDHGN.

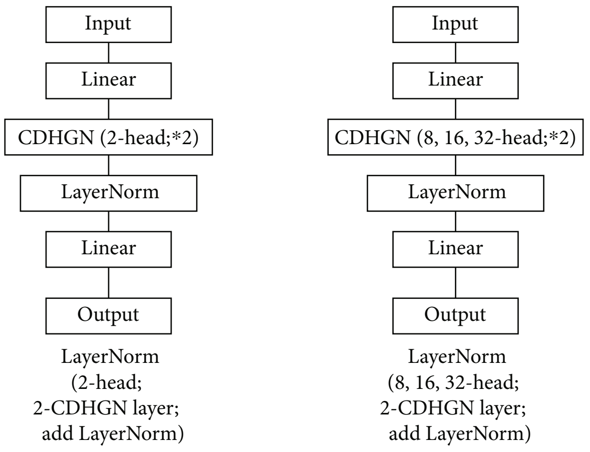

FIGURE 4: Abstract structure of LayerNorm with different number of attention heads in a CDHGN layer.

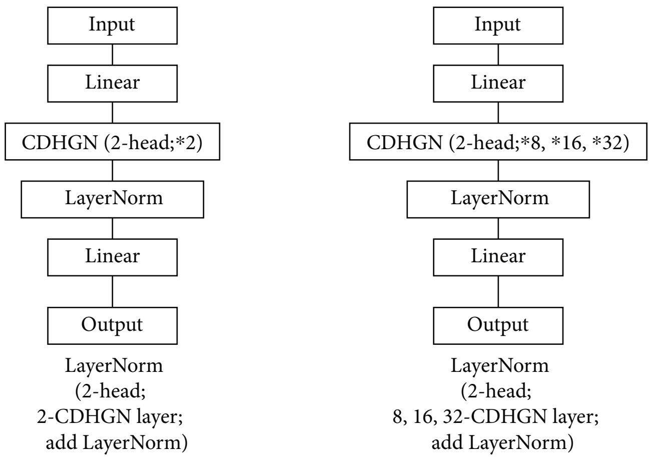

FIGURE 5: Abstract structure of LayerNorm with different number of CDHGN convolution layers.

TABLE 4: Detection result on Base and LayerNorm structure of CDHGN.

CDHGN structure	Dataset division	#Train	#Val	#Test	AUC (test)
Base	0.22 : 0.04 : 0.74	152	28	511	0.9127
LayerNorm	0.22 : 0.04 : 0.74	152	28	511	0.9400

TABLE 5: Detection result on different number of attention heads in a CDHGN layer.

#Multihead (LayerNorm)	Dataset division	#Train	#Val	#Test	AUC (test)
2-head	0.22 : 0.04 : 0.74	152	28	511	0.9400
8-head	0.22 : 0.04 : 0.74	152	28	511	0.9348
16-head	0.22 : 0.04 : 0.74	152	28	511	0.9400
32-head	0.22 : 0.04 : 0.74	152	28	511	0.9316

TABLE 6: Detection result on different number of CDHGN convolution layers.

#CDHGNLayer (LayerNorm)	Dataset division	#Train	#Val	#Test	AUC (test)
∗2-layer	0.22 : 0.04 : 0.74	152	28	511	0.9400
∗8-layer	0.22 : 0.04 : 0.74	152	28	511	0.9398
∗16-layer	0.22 : 0.04 : 0.74	152	28	511	0.9401
∗32-layer	0.22 : 0.04 : 0.74	152	28	511	0.9400

(i) Malicious scanning attacks were tracked during the monitoring process. In March of a certain year, a large-scale network scanning attack was encountered on an external power information network and CDHGN reported an alarm. Network security analysts took advantage of CDHG's ability to continuously capture events and found the IP address of the attack source from massive logs. The attacker was quickly banned, and thus, subsequent attack behaviours were effectively curbed

(ii) During a major international conference, CDHGN noticed an internal email account and found that this account would send emails to an Internet mail address every 450 seconds periodically, with a total of 2,882 emails sent. After retrospective analysis by network security analysts, it was found that the user's Outlook was infected with a virus Trojan, and the Trojan automatically collected documents containing the "schemes" text and sent them to the designated outside mail address

(iii) In January of a certain year, the CDHGN prototype found that the operation of some hosts on the intranet showed unknown anomalies. They frequently initiated communication with other hosts on the intranet and regularly sent data packets to the external network. After the prototype system issued the abnormal alarm, a malicious file sample was found through the traceability analysis by security experts. Through in-depth tracking of the sample file, we collected 7 C&C hosts and 18 IP addresses used by them and finally recovered the attacker's attack steps and purpose. The attacker first remotely controlled the target server through the Weblogic XMLDecoder deserialization vulnerability and then used multiple exploit tools such as EternalBlue and Doublepulsar backdoor program to infect the internal network of the previous controlled server and finally implanted mining programs and remote control programs. Attackers mainly target corporate users. They use extranet sites as a breakthrough to spread malware in the corporate intranet, mine Monero through mining programs to gain profits, and use remote control programs to steal corporate internal data

(iv) In addition, some port scans and *webshell* upload attacks were found, and network security analysts blocked malicious IP addresses through firewalls in a timely manner

With the popularization of the open and interconnected characteristic, the power web service system has become the fortress of the power network security. The adoption of intelligent intrusion detection techniques like CDHGN in this environment has become an urgent need.

5. Conclusion

Based on the analysis of complex behaviours of unknown threat, this paper proposes a detection model based on continuous-time dynamic heterogeneous graph network (CDHGN). It first constructs the continuous-time dynamic heterogeneous graph (CDHG) based on event logs of entities' behaviours. Next, it computes the time embedding of each behaviour based on relative time-encoding function. Then, the CDHG incorporating the time encoding is fed into CDHGN to compute behaviour embedding representation. At last, the behaviour embedding represented by corresponding nodes and edge is used to detect abnormal behaviours. To the best of our knowledge, it is the first to use continuous-time heterogeneous graph construction to detect anomaly events in computer networks. We have implemented a prototype. The experimental evaluation demonstrates that our method outperforms other state-of-the-art methods.

As future work, we will focus on the following issues. The sinusoidal time-encoding function is neat, while many attacks of unknown threat will lurk for a long time interval. A more refined model is needed to capture the complex attack pattern in the time dimension. Further, we believe that by introducing malicious network behaviour knowledge bases, such as attack chain models, to map and weight network behaviours, certain network behaviours that were originally "inconspicuous" can be amplified and well correlated in the spatial dimension. Thereby, it would further improve the detection performance. On the other hand, we are interested in associating unknown attacks by graph similarity. The calculation of entity similarity of knowledge graph has realized many industrial applications, such as recommendation systems. In the network security, users' behaviours on important assets are recorded in real time, and each behaviour process and its related equipment information can be mapped into the CDHG. Therefore, the graph similarity algorithm can be applied to calculate the similarity between the CDHG and the attack graph in the security knowledge base. Then, we can leverage this similarity to further improve the detection performance.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors have no conflict of interest to declare.

Acknowledgments

This work was supported by the State Grid Science and Technology Project: Research on key technology and equipment development of network security access for power monitoring system based on zero trust (5108-202224046A-1-1-ZN).

References

[1] A. Keliris and M. Maniatakos, "Demystifying advanced persistent threats for industrial control systems," *Mechanical Engineering*, vol. 139, no. 3, pp. S13–S17, 2017.

[2] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: a heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Pro-ceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1777–1794, London, United Kingdom, 2019.

[3] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications*, vol. 84, no. 84, pp. 25–37, 2017.

[4] I. Karim, Q.-T. Vien, T. A. Le, and G. Mapp, "A comparative experimental design and performance analysis of snort-based intrusion detection system in practical computer networks," *Computers*, vol. 6, no. 1, p. 6, 2017.

[5] Z. Yu, S. Liu, and W. Wang, "Dynamic threat weight of network security communication based on multisource data analysis," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 6729827, 11 pages, 2022.

[6] G. Li, M. Yin, S. Jing, and B. Guo, "An effective algorithm for intrusion detection using random shapelet forest," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 4214784, 9 pages, 2021.

[7] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," 2017, https://arxiv.org/abs/1710.00811.

[8] F. Shiyuan, G. Xin, Z. Hao, L. Meng, L. Junliang, and X. Jianhang, "An anomaly detection method for power dispatching data based on meta-learning dynamic ensemble selection," *Power System Technology*, pp. 1–13, 2022.

[9] S. Maleki, S. Maleki, and N. R. Jennings, "Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering," *Applied Soft Computing*, vol. 108, no. 108, article 107443, 2021.

[10] A. Legg, O. Buckley, and M. Goldsmith, "Caught in the act of an insider attack: detection and assessment of insider threat," in *2015 IEEE International Symposium on Technologies for Homeland Security (HST)*, pp. 1–6, Waltham, MA, USA, 2015.

[11] T. Senator, G. Goldberg, and A. Memory, "Detecting insider threats in a real corporate database of computer usage activity," in *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1393–1401, Chicago, Illinois, USA, 2013.

[12] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan, "HOLMES: real-time APT detection through correlation of suspicious information flows," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1137–1152, San Francisco, CA, USA, 2019.

[13] C. Wei, Q. Li, D. Guo, and X. Meng, "Toward identifying APT malware through API system calls," *Security and Communication Networks*, vol. 2021, Article ID 8077220, 14 pages, 2021.

[14] S. Ma, J. Zhai, F. Wang, K. H. Lee, X. Zhang, and D. Xu, "{MPI}: multiple perspective attack investigation with semantic aware execution partitioning," in *26th USENIX Security Symposium (USENIX Security 17)*, pp. 1111–1128, Vancouver, BC, Canada, 2017.

[15] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," *arXiv preprint arXiv*, vol. 2001.01525, 2020.

[16] R. Coulter, J. Zhang, L. Pan, and Y. Xiang, "Domain adaptation for windows advanced persistent threat detection," *Computers & Security*, vol. 112, no. 112, article 102496, 2022.

[17] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "HGT-heterogeneous graph transformer," in *Proceedings of The Web Conference*, pp. 2704–2710, Taipei Taiwan, 2020.

[18] W. Yang, P. Gao, H. Huang et al., "RShield: a refined shield for complex multi-step attack detection based on temporal graph network," in *International Conference on Database Systems for Advanced Applications*, pp. 468–480, Cham, 2022.

[19] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," 2020, https://arxiv.org/abs/2006.10637.

[20] A. Bohara, M. A. Noureddine, A. Fawaz, and W. H. Sanders, "An unsupervised multi-detector approach for identifying malicious lateral movement," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 224–233, Hong Kong, China, 2017.

[21] G. Lombardo, A. Poggi, and M. Tomaiuolo, "Continual representation learning for node classification in power-law graphs," *Future Generation Computer Systems*, vol. 128, no. 128, pp. 420–428, 2022.

[22] S. M. Kazemi, R. Goel, K. Jain et al., "Representation learning for dynamic graphs: a survey," *Journal of Machine Learning Research*, vol. 21, no. 70, pp. 1–73, 2020.

[23] C. Song, K. Shu, and B. Wu, "Temporally evolving graph neural network for fake news detection," *Information Processing & Management*, vol. 58, no. 6, article 102712, 2021.

[24] H. Wang, G. Ye, Z. Tang et al., "Combining graph-based learning with automated data collection for code vulnerability detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1943–1958, 2021.

[25] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Hindroid: an intelligent android malware detection system based on structured heterogeneous information network," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1507–1515, Halifax, NS, Canada, 2017.

[26] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[27] A. D. Kent, "Cyber security data sources for dynamic network research," in *Dynamic Networks and Cyber-Security*, pp. 37–65, World Scientific, 2016.

[28] Y. Shen, E. Mariconti, P. A. Vervier, and G. Stringhini, "Tiresias: predicting security events through deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 592–605, Toronto, Canada, 2018.

[29] P. Kingma Diederik and J. B. Adam, "A method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.