applied sciences

MDPI

Article

Accurate Encrypted Malicious Traffic Identification via Traffic Interaction Pattern Using Graph Convolutional Network

Guoqiang Ren [1,2,3], Guang Cheng [1,2,3,*] and Nan Fu [1,2]

[1] School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China
[2] Jiangsu Province Engineering Research Center of Security for Ubiquitous Network, Nanjing 211189, China
[3] Purple Mountain Laboratories, Nanjing 211189, China
* Correspondence: chengguang@seu.edu.cn

Featured Application: The results of this paper can be used in the related fields of encrypted malicious traffic analysis, especially in identifying different attacks based on the graph.

Abstract: Telecommuting and telelearning have gradually become mainstream lifestyles in the post-epidemic era. The extensive interconnection of massive terminals gives attackers more opportunities, which brings more significant challenges to network traffic security analysis. The existing attacks, often using encryption technology and distributed attack methods, increase the number and complexity of attacks. However, the traditional methods need more analysis of encrypted malicious traffic interaction patterns and cannot explore the potential correlations of interaction patterns in a macroscopic and comprehensive manner. Anyway, the changes in interaction patterns caused by attacks also need further study. Therefore, to achieve accurate and effective identification of attacks, it is essential to comprehensively describe the interaction patterns of malicious traffic and portray the relations of interaction patterns with the appearance of attacks. We propose a method for classifying attacks based on the traffic interaction attribute graph, named G-TIAG. At first, the G-TIAG studies interaction patterns of traffic describes the construction rule of the graphs and selects the attributive features of nodes in each graph. Then, it uses a convolutional graph network with a GRU and self-attention to classify benign data and different attacks. Our approach achieved the best classification results, with 89% accuracy and F1-Score, 88% recall, respectively, on publicly available datasets. The improvement is about 7% compared to traditional machine learning classification results and about 6% compared to deep learning classification results, which finally successfully achieved the classification of attacks.

Keywords: encrypted malicious traffic identification; traffic interaction pattern; graph feature; deep learning; graph convolutional network

Citation: Ren, G.; Cheng, G.; Fu, N. Accurate Encrypted Malicious Traffic Identification via Traffic Interaction Pattern Using Graph Convolutional Network. *Appl. Sci.* **2023**, *13*, 1483. https://doi.org/10.3390/app13031483

Academic Editor: David Megías

Received: 27 November 2022
Revised: 20 January 2023
Accepted: 21 January 2023
Published: 23 January 2023

1. Introduction

Telecommuting and telelearning are gradually becoming mainstream lifestyles in the post-epidemic era, leading to a dramatic increase in the number of network devices, cyber threats, and the failure of traditional border security systems. Therefore, it is urgent to accurately identify the encrypted malicious attack traffic in the network.

Cisco's annual network report [1] indicates that the number of global IP devices is expected to exceed three times the world's total population by 2023. The "2021 Communications Industry Statistical Bulletin" published by the Ministry of Industry and Information Technology [2] shows that mobile Internet traffic in China is maintaining rapid growth, and cloud services, mobile networks, and IoT devices are becoming targets for cybercriminals. A report by Zscaler [3] shows that phishing, malicious websites, and malware targeting remote users increased approximately 300 times from January to April 2020, up to 380,000 cases. The "2021 Annual Cyber Threat Defense Report" published by CyberEdge [4]

shows that the number of organizations attacked by cyberattacks worldwide rose 5.5%, the highest level in the past seven years. These attacks have resulted in information leakage and property damage to individuals, companies, and countries, making identifying malicious traffic critical in maintaining cybersecurity and national security.

In complex networks, the emergence of various new applications and attacks leads to the traditional feature representation methods that can no longer be applied to attack identification, and the data interaction of new attacks has become more complex. Therefore, the key to achieving high network security management is to capture the increasingly complex correlation features of malicious traffic and explore the generality of interaction patterns to support accurate, efficient, and generalization-capable attack identification.

To address the above problems, research in attack identification has evolved significantly over time, as shown in Figure 1. Early works [5,6] used plaintext information extracted from packet headers to make inferences about the degree of anomalies, such as IP address and packet header size, as shown in Figure 1a. However, these methods only rely on header information with a short length and single feature dimension, needing more analytical study of correlation features among packets from a holistic perspective, and the available plaintext information becomes more ambiguous. Therefore, these methods are no longer applicable to complex attack identification.

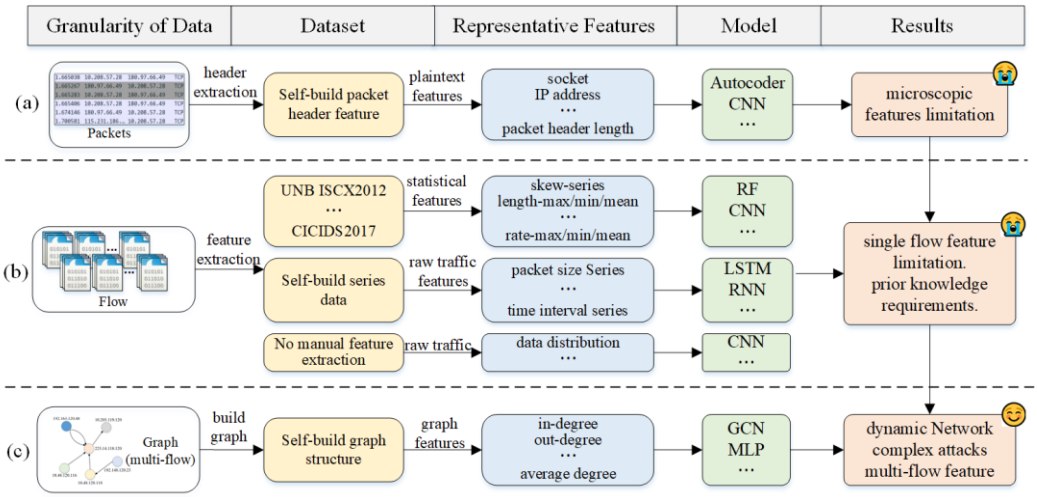


Figure 1. Three different attack identification methods.

Subsequently, many studies [7,8] capture the interaction patterns between hosts based on single flows and mine the correlations between different packet features, as shown in Figure 1b. According to the features, these studies are mainly divided into identification methods based on single-flow statistical features, identification methods based on single-flow raw features, and deep learning identification methods based on single-flow raw data. For the methods with single-flow statistical features, some representative public datasets exist, such as KDD99, NSL-KDD, UNB ISCX2012, UNSW-NB15, and CICIDS2017.

However, the description of a complex attack is often a combination of different types of flows. In reality, a part of flows does not show obvious anomalies. However, they may be an essential set of traffic at the beginning or end of an attack. Detecting this part of pseudo-normal traffic can intercept or discover the attack in time, which cannot be achieved using single flows. An efficient attack identification method must be able to describe network activities from a higher level and more macroscopic perspective and explore the behavioral characteristics of network nodes.

Therefore, graph-based attack identification methods are rapidly developing, as shown in Figure 1c. A study [9] believes that all data instances in network communication are intrinsically related. Graphs are an effective method to obtain data correlation, where dependencies between related objects are expressed through graph connections. A study [10] shows that graphs are an alternative to breaking through the limitations of network flow

models by obtaining information from flow-level data that reflects the actual behavior of hosts. Using graphs can improve the robustness of complex communication patterns and unknown attacks. The robust structural feature and spatial representation capabilities of graphs have a substantial value for malicious traffic identification methods.

Based on the above problems, we use the graphs' spatial feature representation capability to explore the interaction pattern of networks for identifying malicious traffic. In this paper, we propose an identification method of attack that combines spatial features of graphs and statistical features of flows, named G-TIAG. It aims to learn the interaction patterns from large-scale network traffic. Specifically, we first justify using graphs to represent different attacks and propose a rule for constructing the traffic interaction attribute graph (TIAG). Then, we select the statistical features of nodes for TIAG and quantitatively demonstrate the effectiveness of TIAG. Next, we propose a graph convolutional network (GCN) model combining GRU and self-attention. This model can capture the spatial features using GCN, the temporal features using GRU, and the differences in the importance of temporal features using self-attention. Finally, the G-TIAG accurately identifies malicious traffic by using TIAGs as the input of the GCN model and expressing the correlation of intrinsic features.

Our contribution can be briefly summarized as follows.

- We propose a rule for constructing the network traffic interaction attribute graph. Compared with other studies, TIAG combines the spatial features of graph structure and the statistical features of each flow, which is proven to distinguish well between normal data and attack data. Especially, TIAG can also characterize the type of attacks with little data. Therefore, TIAG can describe the difference in interaction patterns of different attacks in a specific time period. Moreover, adjacent TIAGs also can describe the difference in the changes in interaction patterns of different attacks in a time period.
- We construct a convolutional graph network with TIAGs as input. Compared with other studies, G-TIAG combines GRU and self-attention with GCN. It pays more attention to the relation between TIAGs of adjacent inputs using GRU and focuses on the essential features evaluated by self-attention. Anyway, we demonstrate through ablation experiments that the introduction of GRU and self-attention leads to improved performance of G-TIAG in identifying attacks with little data.

The rest of this paper is organized as follows. Section 2 summarizes existing works related to attack identification. Section 3 presents the fundamentals of the TIAG used to represent traffic interaction patterns, and Section 4 designs a classifier based on GCNs. Section 5 shows the results of G-TIAG against existing methods through a comparative analysis. Finally, we provide a discussion and conclusion of our work.

2. Related Work

Depending on the granularity of the studied data, current attack identification methods are mainly classified into packet-based identification methods, flow-based identification methods, and graph-based identification methods.

2.1. Packet-Based Attack Identification Methods

Packet-based attack identification methods use packet header information, such as IP address and header size, to infer the degree of anomaly. Wang et al. [5] used packet header information and correlation between the target IP address and port of edge routers for identification. LUCID [6] represented the packet header data sequence as an image and used CNN for feature learning. Kitsune [11] was a packet-based method using an integrated autoencoder (AEs) as a detector. It extracted the source host, channel (source and target host pair), and socket characteristics and performed attack detection using an unsupervised learning model. The result showed that deep learning could improve the learning of host interaction patterns. In addition, related studies treated packet header information as words in natural language and used embedding methods to generate numerical representations

for detection. A study in [12] mentioned an IP2Vec applicable to logging flows for learning embeddings of IP addresses, port numbers, and protocols.

Although packet header information is easy to extract, it has low confidence in dynamic and high-speed networks, which is also ineffective in detecting attacks with complex behaviors, and large-scale packet-level identification incurs huge resource and time costs. Especially the features of packets are microscopic representations of traffic and cannot globally express complex attack behavior patterns.

2.2. Flow-Based Attack Identification Methods

Flow-based attack identification methods are mainly divided into flow statistical feature-based identification, flow raw feature-based identification, and flow raw data-based deep learning identification, depending on the features. They can capture the interaction pattern between hosts and mine the feature relation among packets. ZED-IDS [7] trained deep learning models based on statistical features generated by the CICFlowMeter to achieve the detection of potential zero-day attacks. FS-Net [8] used a recurrent neural network for feature learning based on the sequence of packet size in a flow. STIDM [13], on the other hand, classified attacks based on the sequence of packet size and time intervals using a combination of CNN and LSTM. Based on raw traffic, Xiao et al. [14] transformed the traffic into a two-dimensional matrix and used CNN to extract features automatically to mine the correlation features between different instances.

However, the flow-based attack identification method has limitations. Because the representation of an attack behavior may be jointly determined by the features of multiple flows, focusing on a single flow only can make the feature representation less comprehensive and complete.

2.3. Graph-Based Attack Identification Methods

Data interactions in the network can be represented as graphs. Therefore, there are many studies based on graphs to identify attacks. Wang et al. [15] built a graph structure based on the original packets and analyzed the degree distribution through large deviation theory to achieve the detection of Bot. Daya et al. [9] constructed a graph structure based on the packets and achieved the identification of the Bot by the indegree and outdegree features. Tian et al. [16] built a graph named DNG for DNS and found that the DNG enabled a differentiated description of DNS query failure instances. Protogerou et al. [17] proposed a distributed method for DDoS attacks using monitors at the active network nodes. This method created simulated datasets of normal and abnormal flows, extracted content features, and used the graph neural network for learning. Khalaf et al. [18] used a collaborative environment of intelligent agents to detect attacks. Each agent deployed a hierarchical graph-based learning network to effectively monitor the entire infrastructure and counter the propagation of attacks.

Anyway, Do et al. [19] constructed a behavior profile file of malware for advanced persistent threats and used a graph network to analyze the behavior profile to identify attacks. Liu et al. [20] pointed out that the existing DDoS detection works to face the problem of non-negligible time delay and the poor generality of detection models for attacks with different types and rates. Therefore, they established for the first time a directed graph kernel to measure the divergence features between the current graph and the normalized graph and constructed a dynamic threshold and a freezing mechanism to display changes in the standard traffic. Then, Wang et al. [21] raised the issue that the existing studies are insensitive to a small number of attacks. Therefore, they proposed the THREATRACE, a real-time detection system that detected host threats at the system entity level without prior knowledge of attack patterns. The THREATRACE uses an inductive graph neural network to learn the role of each entity in a graph. Yang et al. [22] proposed a temporal graph-based method. The continuous-time dynamic graphs were constructed from the log records based on the interactions between users and entities. The results show

that this method possesses better detection performance than state-of-the-art methods in both direct push learning and inductive learning.

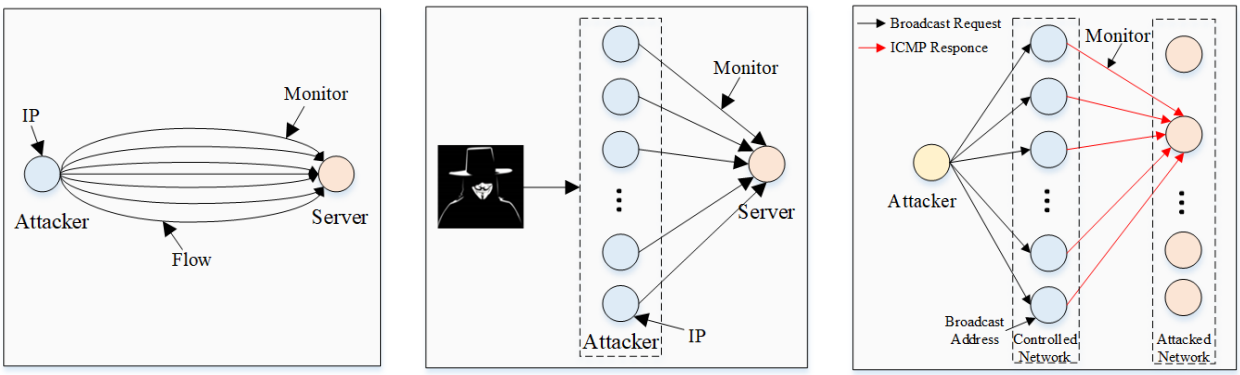
The identification methods based on the graph can provide a more comprehensive description of the traffic composition of an attack behavior from a multi-flow perspective. Moreover, these methods can reflect the changes in an attack behavior over time, which are more adaptable to identifying attacks in dynamic networks.

3. Graph Structure Abstraction for Malicious Attracts

In this section, before formally describing the network traffic interaction attribute graph (TIAG), we first illustrate the rationality of using graphs to represent the feature of attacks, and we secondly describe the principles of TIAG construction. Finally, we show that benign data and five attacks exhibit significant differences in TIAG.

3.1. Interactive Mode for Malicious Attacks

To illustrate that the graph representation is distinguishable for different attacks, we first select three common attacks among the five attacks studied in this paper and use the graph to describe the interaction pattern between the attacking entity and the attacked entity. The results are shown in Figure 2.



(**a**) Port Scans (**b**) DDoS (**c**) Smurf Attack of Doc

Figure 2. Interaction patterns of three common attacks. (**a**) Description of the interaction pattern of Port Scans Attack. (**b**) Description of the interaction pattern of DDoS. (**c**) Description of the interaction pattern of Smurf Attack.

Each node in the graph represents an independent IP in the real world, and each directed edge represents an interaction between two hosts. The monitoring points of the attacks are labeled in Figure 2. The graphs constructed from the network traffic obtained at the monitoring points can distinguish attacks based on features such as IP address, data direction, and interaction pattern.

For example, (1) The port scanning attack detects the availability of the target port by performing a handshake. From the monitoring points, the interaction patterns of this attack are as follows. The data comes from a relatively fixed IP, and only the port number changes. Anyway, the data interaction is limited to the handshake process, and the data have a particular temporal order. (2) The DDoS attack launches large-scale requests to the target host by controlling many distributed hosts. From the monitoring points, the interaction patterns of this attack are as follows. The data are from different domains and network segments of distributed IPs, and the ports are common. Anyway, many data are transmitted from attackers to attacked parties, and the data tend to arrive at the target in a tiny time fragment. (3) The representative Smurf attack in the Doc makes all hosts in the attacked network hit by many packets. This attack sends ICMP packets to the broadcast IP of the controlled network and sets the reply address to the broadcast IP of the attacked network. From the monitoring points, the interaction patterns of such attacks are as follows. The IPs of the attacker and the victims are often located in the same network segment, and

the data interaction is unidirectional, with a large amount of data arriving at the target in a tiny time interval.

Based on the above description of the three attack interaction patterns, graphs have a reasonable contribution to characterizing, classifying, and identifying different attacks in the real world.

3.2. Traffic Interaction Attribute Graph

In this subsection, we will introduce the construction process of TIAG and perform a simple transformation of TIAG to satisfy the input of GCN. Finally, we add attributive features to each node in the TIAG.

3.2.1. Construction of TIAG

From the analysis of interaction patterns in Section 3.1, we find that using IP addresses as nodes of the graph and interaction patterns between different IP addresses as edges of the graph can effectively differentiate attacks. Therefore, based on the above principles, we design and implement the construction of TIAG, a graph of traffic interaction patterns, and use it to describe the interaction patterns between network hosts. We consider the limitation of using a single flow to reflect user behavior, and multiple flows in a particular time can reflect the user behavior of that network at the moment. The features in a specific time granularity range can reflect user behavior from a more macro and holistic perspective. In addition, when attacks occur, different TIAGs constructed in adjacent periods have significant differences. We abstract each IP address in a specific time range ΔT as a node v in TIAG and abstract the interaction between node v_i and node v_k as a set of directed edges $\{e_{ik}\}$, where there are often multiple edges of both directions between two nodes. Thus, the TIAG is defined as follows.

Definition 1(Traffic Interaction Attribute Graph): A TIAG is a directed graph, which is represented by a four-tuple, TIAG = (V, E, F, T), such that:

- V is a set of vertices represented by independent and non-repeating IP addresses. Each vertex $v \in V$ represents a user involved in interaction in $\Delta T \in T$, and each vertex of the transformed TIAG (3.2.2) possesses a 9-dimensional feature vector $f \in F$ (3.2.3).
- E is a set of edges. Each edge $e \in E$ connects two interacting nodes, and an edge represents one flow of interaction between two nodes.

We illustrate the traffic interaction pattern with a sequence of packets in the ΔT time range as an example, as shown in Figure 3.

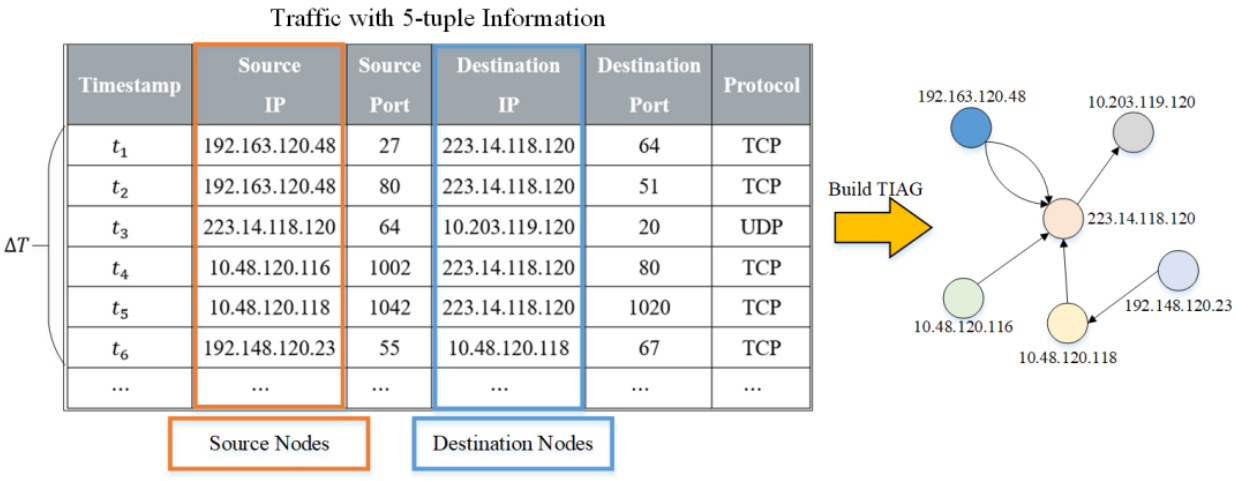


Figure 3. Interaction patterns of three common attacks.

Since the packet sequence usually consists of multiple flows, we divide the whole sequence into individual flows, where each flow is defined as a series of packets with the same 5-tuple (source IP, source port, destination IP, destination port, protocol). In our experiment, we consider the two flows to be different communications for the attacks when two flows have the same IP and port number but different protocols. Each flow has a timestamp which is the arrival time of the first packet in this flow. In addition, the flows are listed in order according to the timestamp. Next, we use the flows in ΔT to construct TIAG, and ΔT is 50 in our experiment. Anyway, the interval time between two TIAGs is T, which will be discussed in Section 5.3.4. T is finally set to 200 in our experiment. If a packet disorder problem is found in grouping flows, the flows with disordered packets must be rearranged.

In the time interval of ΔT, A TIAG is denoted as $G_t = (V_t, E_t, F_t)$, the source node set in the time interval of ΔT is denoted as $V_{SIP} = \{SIP_1, SIP_2, SIP_3, \ldots, SIP_m\}$, the destination node set is denoted as $V_{DIP} = \{DIP_1, DIP_2, DIP_3, \ldots, DIP_n\}$, and the edges are represented as $Edges$=$Edges_{m\times n} = \{e_{ik}\}$, when there are edges between two nodes SIP_i and DIP_k, then e_{ik} = 1, otherwise e_{ik} = 0. The specific construction algorithm of TIAG is shown Algorithm 1.

Algorithm 1 Construction of TIAG

Input: Packet sequence $P = \{p_1, p_2, \ldots, p_N\}$ in ΔT
Output: A $TIAG_t$
1: Traffic = Group_Flow($p_1, p_2, \ldots, p_N$), len = len(Traffic) //group flows
2: Initialization(V_{SIP}, V_{DIP}) // Initialize a vertex set and edge set
3. Initialization_E($Edges_{len\times len}$) // Initialize a largest directed edge record array
4. s = d = 0 //Initialize location records
5: **for** flow ∈ Traffic **do**
6: SIP_i = flow.source_IP, DIP_i = flow.destination_IP //Mapping nodes to IP
7: **if** $SIP_i \notin V_{SIP}$ **then** V_{SIP} = add(SIP_i,V_{SIP}), index_s = s; s = s + 1
8: **else** index_s = get_index(SIP_i) // Find the first index of SIP_i node
9: **if** $DIP_i \notin V_{DIP}$ **then** V_{DIP} = add(DIP_i, V_{DIP}), index_d = d, d = d + 1
10: **else** index_d = get_index(DIP_i)
11: $e_{index\_s,index\_d} = e_{index\_s,index\_d} + 1$
12: **end for**
13: len_SIP = len(V_{SIP}), len_DIP = len(V_{DIP}) //Calculate the number of non-duplicate IPs
14: $Edges_{len\_SIP\times len\_DIP}$ = Delete($Edges_{len\times len}$, len_SIP, len_DIP) //Reduce the array size
15: **Return** $TIAG_t$

Algorithm 1 takes as input a sequence of packets at ΔT. It first groups flow, processes the disorder, and calculates the number of flows (line 1). Then, it initializes the source and destination nodes, the directed edge record array, and the location record (lines 2-4). For each flow, Algorithm 1 records the source and destination IP (line 6), and if the node does not exist in the corresponding node-set, algorithm 1 joins the node and records the location index of the node in the corresponding node-set, and update the location records (lines 7–9). If the node already exists, it finds the position index of the first occurrence of the node in the set (lines 8–10), and the index obtained in the source nodes is used together with the index obtained in the destination nodes to update the directed edge records array (line 11). After the flows are processed, it calculates the number of unduplicated IPs in the two node sets (line 13) and reduces the size of the directed edge record array (line 14). Finally, the two node sets and the directed edge record array obtained construct the TIAG (line 15).

3.2.2. Transformation of TIAG

When a node has a closer connection with other nodes in TIAG, we call it an important node, corresponding to the data interaction pattern. If an IP has a complex and frequent interaction pattern with another IP, we call this IP a suspected important node. To focus more attention on these critical nodes and to satisfy the input requirements of the node

classification of GCN, we perform a simple transformation of TIAG. A node is transformed from representing an IP to representing a flow, and when there is a common IP between two flows, it is connected by an edge. An edge is transformed from representing the interactive flow between two IPs to a common IP existing between two flows. The TIAG is transformed from a directed unweighted graph to an undirected weighted graph, and the weights represent the number of shared IPs. We use the TIAG constructed in the ΔT as an example to illustrate the specific transformation process, as shown in Figure 4.

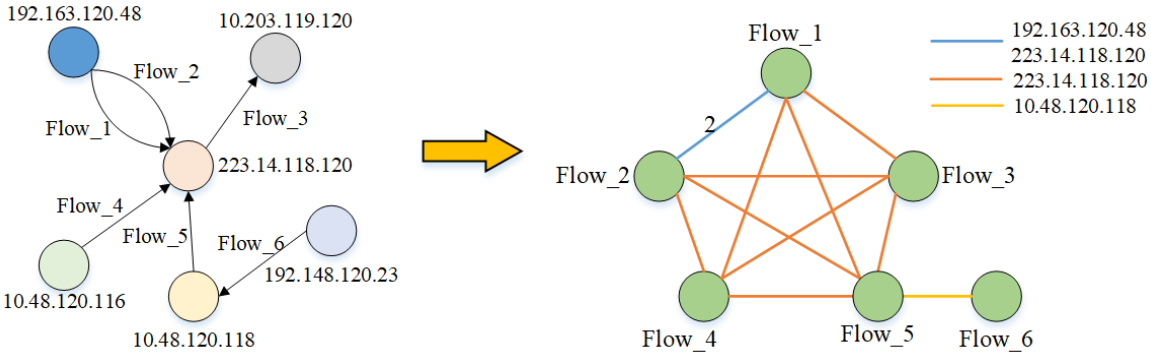


Figure 4. The transformation rule of TIAG.

Figure 4 shows that the transformed TIAG has three different types of edges, representing the more critical three nodes in the left figure, corresponding to the IP: 192.163.120.48, 223.14.118.120, and 10.48.120.118. It achieves to put more attention on the essential nodes and characterizes the connection relation between important nodes and other nodes to a greater extent.

3.2.3. Selection of Attributive Features

Each node of the transformed TIAG represents a flow, and we select the feature that carries the most information for each node as the attributive feature. We use the 85-dimensional flow statistics features obtained by the CICFlowmeter feature extraction tool as the basis for feature selection using the information gain indicator, which uses attributive rank ordering to reduce the effect of noise caused by irrelevant features. For dataset S containing k categories, the information gain of each feature *feat* is calculated by the entropy value, and the formulas for calculating the entropy value and information gain are shown below.

$$\text{Entropy(S)} = -\sum_{k=1}^{k} P_k log_2 p_k \tag{1}$$

$$Gain(\text{S}, feat) = \text{Entropy(S)} - \sum_{value(f)=v}^{m} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2}$$

where p_k denotes the occurrence probability of the kth category, $value(feat) = v$ denotes the $feat$ value is v, and S_v denotes the set of samples with $feat = v$, the m value is denoted as the different values of v.

Information gain has an excellent performance in facing the overfitting problem, and we use it as the basis for selecting nine features as the attributive features of TIAG. The feature information is shown in Table 1. For traditional machine learning methods used in our experiment, we set selected 9-dimensional features for each flow as baseline features to provide support for comparing with G-TIAG.

Table 1. Attributive features of each node.

No.	Feature	Infor Gain	No.	Feature	Infor Gain
1	Packets Length Std	0.64	6	Flow Duration	0.44
2	Total Length of Bwd Packets	0.61	7	Flow Bytes/s	0.38
3	Packets Length Variance	0.58	8	Flow Packets/s	0.31
4	Total Length of Fwd Packets	0.55	9	Total Fwd Packets	0.29
5	Average Packet Size	0.53			

3.3. The Benefits of TIAG

(1) The representation ability of raw TIAG: To prove that the description of features, such as the number of nodes, the number of edges, the average degree, and the direction, can be reflected in the original TIAG, we first define the features of the graph, then we classify the features in the original TIAG into three main categories: basic features, state features, and statistical features, and their specific descriptions are shown in Table 2.

Table 2. Description of the original TIAG features.

Feature	Name	Description
Basic feature	Average degree	Average of the number of edges directly connected to the node
	Node directivity	The ratio of in degree and out-degree nodes to all nodes
State feature	Relative connection density	The ratio of node degree to degree maximum
	Maximum depth	Maximum number of links in one direction
Statistical feature	Node degree distribution	Probability distribution of the node degree
	Joint degree distribution	The ratio of the number of edges between nodes of different degrees to the total number of edges

Basic features describe the basic parameters of the topology, such as the number of nodes, the number of edges, the average degree, and the node orientation. Some abnormal attacks may cause the basic features of TIAG to change significantly. For example, the proliferation of worms may cause the average degree and the node orientation features to increase significantly. State features refer to the impact of the position and state of nodes and edges on the whole topology graph. Some abnormal behaviors may not cause changes in the topology of TIAG but may affect the state between nodes and edges. For example, malicious scanning may cause an extreme increase in the relative connection density of TIAG. Statistical features describe the uncertainty, stability, and probability distribution of nodes and edges of TIAG. Compared with other features, anomalous behaviors have a more significant impact on the statistical features. For example, DDoS attacks can change significantly in statistical features. Therefore, for different anomalous behaviors, we find that the anomalies can be better identified and handled by analyzing the changes in the corresponding features of TIAG in adjacent time periods.

We used the CAIDA [23] dataset to evaluate the representation of the original TIAG on benign and two abnormal data, and the results are shown in Table 3.

DDoS attacks control many hosts to send packets to the target host simultaneously, and their interaction patterns are highly similar. For example, the data sending time is close, and the port number of the target host is the same. Table 3 shows that the original TIAG shows a surge in the proportion of in-degree nodes and the maximum connection density when a DDoS attack occurs. The worm can quickly increase the number of infected hosts, and the network structure shows a significant proliferation feature. We find from the above table that when worm propagation occurs, the original TIAG shows a decreasing trend in the average degree and a significant increase in the maximum depth. The following conclusions are obtained by splitting the original TIAG into three features for comparative analysis: the

emergence of anomalous attacks makes the interaction pattern change accordingly, and the use of TIAG can effectively distinguish benign data and abnormal data.

Table 3. Feature representation of raw TIAG.

Type		Node Number	Edge Number	Average Degree	Maximum Depth	Maximum Connection Density	In Degree Nodes	Out Degree Nodes
Benign data	HTTP	10,365	11,287	2.11	4	0.06	0.26	0.74
	eMule	9812	11,069	2.59	115	0.01	0.28	0.72
abnormal data	DDoS	2145	2560	1.89	3	0.79	0.93	0.07
	Worms	5240	5379	1.42	254	0.01	0.20	0.80

(2) Visualization of the representation ability of TIAG: We first used the splicing function to convert the feature and adjacency matrix into vectors to visually demonstrate that TIAG can effectively represent the benign data and the five attacks. Then, we used the t-SNE method to reduce the high-dimensional features to a two-dimensional space and visualize them. Specifically, we randomly selected 100 samples from each of the six data types in the test set for analysis, and the results are shown in Figure 5.

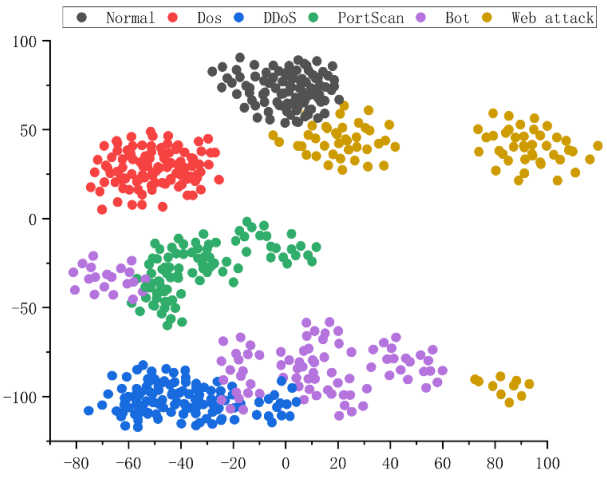


Figure 5. Similarity measure of TIAG.

We find that benign data, DDoS, DoS, and Portscan data are each clustered together sizably and are distant from the other categories. However, some of the Bot and Web Attack data are also clustered together, but there are multiple cluster centers in the same class, and the distance between each cluster center in the same class is not close. The classification results of the subsequent experiments also show that these two data are closer to other classes of data, especially the Portscan. The reason for the above problem is that the training samples of Bot and Web Attack are small, and the number of training samples of these two attacks still cannot reach the average value due to the objective quantitative limitation.

4. The Proposed G-TIAG

With the help of TIAG, the classification problem of benign and attack data is transformed into a graph classification problem. The convolutional graph network (GCN) has advantages over other networks for processing non-equal length data and can deeply learn the structure correlation between nodes and edges, gaining the spatial features of each region from a graph. In this subsection, we will present the design details of the classifier based on GCN and TIAG, which is named G-TIAG.

4.1. G-TIAG Overview

The overview of training G-TIAG is presented in Figure 6. In the training process, we first collect traffic with different attacks and normal traffic to obtain the TIAGs using Algorithm 1, described in Section 3.2. Then, we compute the adjacency matrix and feature matrix from the TIAGs as the input to the learning network. The GCN learns the features of TIAG under a specific moment. After two convolutional layers, the outputs are further input to the GRU for time series modeling to deeply capture the correlation features of TIAG between several neighboring moments. The outputs of GRU further use self-attention to capture adequate information. Finally, the classification results are output through the fully connected layer.

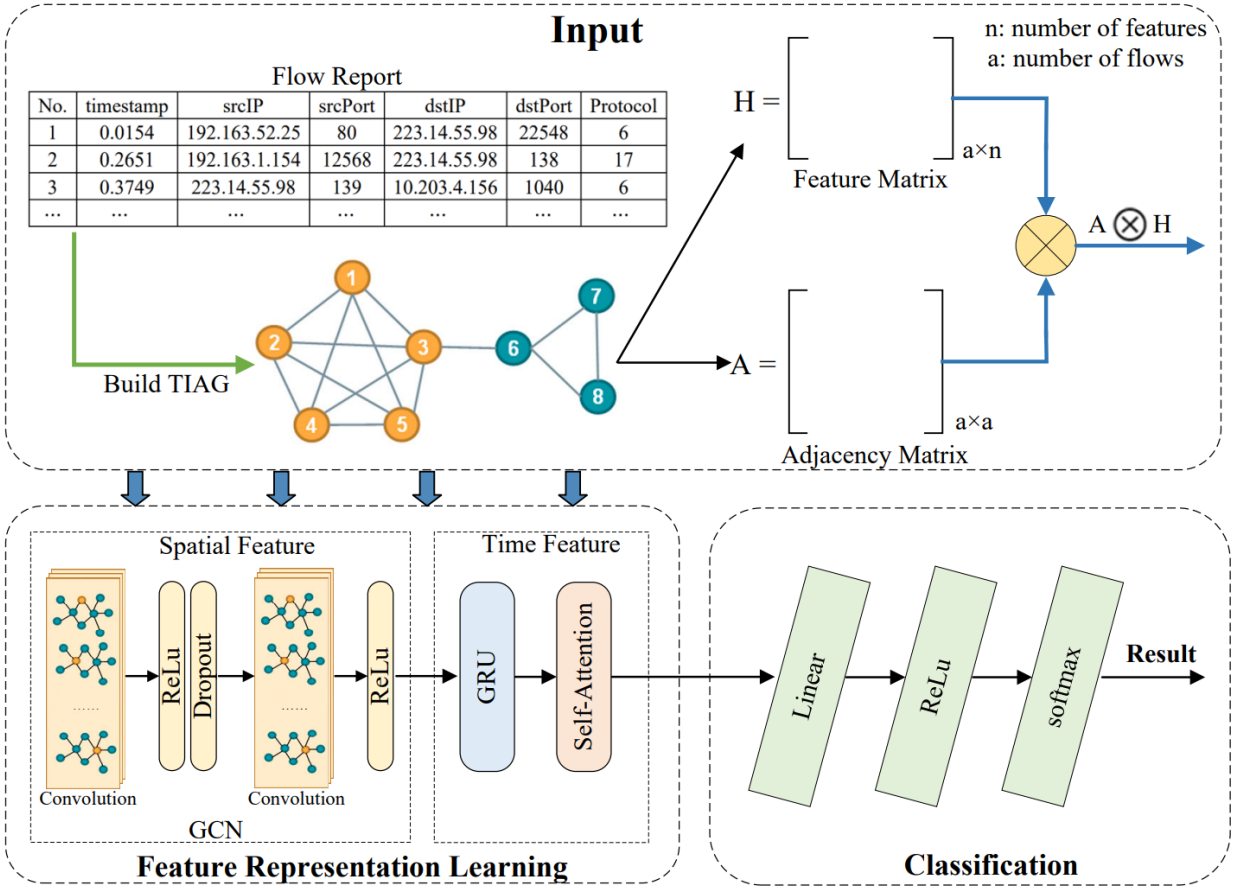


Figure 6. Structure of G-TIAG.

4.2. G-TIAG Architecture

Given a set of TIAGs containing benign data and five different attack data $\{G_1, G_2, \ldots, G_n\}$ and their labels $\{y_1, y_2, \ldots, y_n\}$. G-TIAG aims to learn a representation vector $f(G)$ that can predict each TIAG more accurately, i.e., $\hat{y_n} = g(f(G_n))$.

The most critical part of G-TIAG is the feature representation learning part, which is mainly based on the GCN with the GRU and self-attention as an aid for extracting intrinsic features from TIAGs for training, and the fully connected layer is mainly used for classification. The loss function $\mathcal{L}$ quantifies the difference between the predicted results and the actual labels, and the value of $\mathcal{L}$ needs to converge gradually during training. The notations used in the G-TIAG are shown in Table 4.

Table 4. List of notations.

Notation	Meaning	Notation	Meaning
G	A TIAG as the input of GCNs	X	The output vector of GCN
A	Adjacency matrix of TIAG	$f(G)$	The output vector of G
H	Feature matrix of TIAG	$F(G)$	Representation in a latent space
L	Duration to build a TIAG	$\mathbb{R}^n$	A subset of a n-dim countable set
Z_t	The output of representation layer	y_i	Actual label vector
A_{attr}	Weight vector of self-attention	$\hat{y}_i$	Predicted label vector
h_t	State vector of GRU at a moment	$\mathcal{L}$	Loss function

(1) Feature Representation Layer: In G-TIAG, the intrinsic features of each TIAG are learned by a feature learning structure based on the GCN model, aided by the GRU and self-attention. The GCN uses two convolutional layers for feature extraction for the graph constructed under a specific moment. Then, the nonlinear activation function is used to enhance the model's ability to express nonlinear features. The Dropout function is used to avoid the overfitting problem during the forward propagation. The output vector by the GCN is noted as X. Next, X is further input into the GRU unit to extract the graph's change at adjacent moments. Then, we calculate the state vector h_t in each time step t, obtain the weight vector through the self-attention, and output the final feature representation Z_t at the moment t according to the weights provided by self-attention. We will describe the representation layer's design in the following subsection.

(2) Fully Connected Layer: After the feature learning, G-TIAG mainly contains three Linear layers, two activation functions (ReLu), and a Softmax function to obtain the final result. In G-TIAG, we use two ReLu functions between the three Linear functions to increase the ability to represent nonlinear features. For each G_i, we treat the output of the Linear functions as a feature vector of G_i, denoted as $f(G_i)$. Subsequently, $f(G_i)$ is mapped into a new hidden space $F(G_i) \in \mathbb{R}^c$, where c is the number of independent categories and c is taken as 6 in this paper. Finally, the probability of G_i belonging to each category is obtained using the Softmax function, which is denoted as $\hat{y}_i$, i.e., $\hat{y}_i = Softmax(F(G_i))$.

(3) Loss Function: G-TIAG uses the cross-entropy $\mathcal{L}$ as the loss function. $\mathcal{L}$ is usually used in multi-classification problems to calculate the loss between the predicted and true labels, as shown below.

$$\mathcal{L} = -\frac{1}{|N|}\sum_{i=1}^{|N|}\sum_{c=1}^{C} y_i log(\hat{y}_i) \tag{3}$$

where $|N|$ is the number of training instances and C is the number of categories. The experiments find that the gradient of the cross-entropy loss function at the last layer is proportional to the difference between the actual and predicted labels, implying that it converges faster than the mean square loss error function.

(4) Optimizer: G-TIAG uses the Adam optimizer, which is an algorithm for stepwise optimization of stochastic objective functions based on adaptive moment estimation. We define the optimizer while initializing the learning rate to 0.001 and control the update step size by considering the gradient's first-order moment estimation and second-order moment estimation.

4.3. Design of Feature Representation Layers

The feature representation layer mainly includes spatial feature representation based on GCN and temporal feature representation based on GRU and self-attention. GCN aims to learn the topological structures and node attribute information in the TIAG under each timestamp. The purpose of GRU and self-attention is to further learn the change and evolution patterns of TIAGs over time under adjacent timestamps.

4.3.1. Design of GCN

The GCN learns the topological relations between nodes and nodes, edges and edges, and between nodes and edges for the TIAGs, and extracts two matrices from each TIAG as the input to the GCN, and we first define the input and output of the GCN.

- Feature matrix $H_{v\times x}$, which is formed from the feature representations of each node, where v denotes the number of vertices in the TIAG, and x denotes the number of attributive features of each node.
- Adjacency matrix $A_{v\times v}$, which is obtained from each TIAG.
- The output vector, which is denoted as $X_{t-L+1}^{t} = \{X_{t-L+1}, X_{t-L+2}, \ldots, X_t\}$.

We design a two-layer convolutional structure to implement low-order graph embedding. Each convolutional layer can be written as a nonlinear function with the following rules.

$$H^{l+1} = f\left(H^l, A\right) \tag{4}$$

$H^0 = H$, $H^l = Z$ (H^l is the output of layer l), l presents the number of layers, and the models differ in the choice and parameterization settings of the $f(\cdot,\cdot)$ function. Specifically, the propagation rules for each layer are as follows.

$$f\left(H^l, A\right) = \sigma\left(AH^lW^l\right) \tag{5}$$

where W^l is the weight matrix of the l-layer neural network, and $\sigma(\cdot)$ is the ReLU nonlinear activation function. In addition, we add a unit matrix I to the matrix A to achieve self-loop and normalize A. To make each row of A add up to 1, we multiply by the inverse of a degree matrix D. D^{-1} multiplied with A indicates averaging over adjacent node features. We adopt the new propagation rule proposed by Welling et al. [24], as shown in the following equation.

$$f\left(H^l, A\right) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^lW^l\right) \tag{6}$$

where $\hat{A} = A + I$, $\hat{D}$ is the degree matrix of $\hat{A}$, $H^l \in \mathbb{R}^{N\times D_i}$ is the activation matrix in layer l, $W^l \in \mathbb{R}^{D_i\times D_{i+1}}$. Finally, we obtain the output of the GCN by averaging the output vectors of the second convolutional layer, as shown in the following equation.

$$X = f\left(f\left(H^0, A\right), A\right) = \sigma\left(A\sigma\left(AH^0W^0\right)W^1\right) \tag{7}$$

4.3.2. Design of GRU Structure

Different attacks differ in their topological changes over a specific time interval. To effectively extract the variability between TIAGs in adjacent time intervals, we use the GRU as the basis for temporal feature extraction. GRUs use a threshold mechanism to remember as much long-term information as possible. They can capture time-dependent features by capturing the current network information and preserving the changing trend of historical network information. Compared with LSTM, GRU units have fewer parameters and lower complexity, and compared with RNN, GRU can solve the gradient disappearance and gradient explosion problems. We combine GRU with a self-attention capable of capturing critical information to achieve the representation of the changes and evolutionary patterns of TIAGs in adjacent time intervals.

We design this part to consist of a GRU and a self-attention. First, the output vector X of GCN is input to the GRU unit to obtain the state vector h_t in each time step L. Using the hidden state at the previous moment $t-1$ and the input vector at the current moment t, the feature vector t is obtained using the following equation.

$$z_t = \sigma_s(x_tU^z + h_{t-1}W^z + b^z) \tag{8}$$

$$r_t = \sigma_s(x_t U^r + h_{t-1} W^r + b^r) \tag{9}$$

$$\widetilde{h}_t = \tanh\left(x_t U^h + (r_t \odot h_{t-1}) W^h + b^h\right) \tag{10}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \tag{11}$$

where z_t is the output of the update gate to control the effect of the feature h_{t-1} at moment $t-1$ on the current moment t. r_t is the output of the reset gate to control the extent to which features h_{t-1} at the moment $t-1$ are ignored. U, W is the training weight matrix, σ_s is the Sigmoid activation function and b^z, b^r, b^h are the bias terms. In experiments, we find that GRU has difficulty processing longer vectors, and the hidden representation of longer vectors appears in the next moment. Therefore, we design to add self-attention after the GRU to capture adequate information. The output vector h_t of the GRU is sent to self-attention to calculate the attention values of neighboring nodes and assign different weights to each neighboring node. First, for an input vector h_i, we use Equation (12) to obtain the e_i, which is responsible for measuring the importance of h_i. $W_{(1)}$and$W_{(2)}$ are the hyperparameters, and $b_{(1)}$ and $b_{(2)}$ are the bias terms. Then, we use Equation (13) to obtain $A_{attr(i)}$, which denotes the probability of selecting the ith input vector. The n value is the number of input vectors.

$$e_i = W_{(2)}\left(W_{(1)} h_i + b_{(1)}\right) + b_{(2)} \tag{12}$$

$$A_{attr(i)} = \frac{exp(e_i)}{\sum_{k=1}^{n} exp(e_k)} \tag{13}$$

Finally, the final representation is obtained by using the weight sum method according to the input vector h_i and $A_{attr(i)}$, as shown in equation (14).

$$Z_t = \sum_{i=1}^{n} A_{attr(i)} \times h_i \tag{14}$$

With all the above designs, G-TIAG can capture spatial features of TIAGs at each moment and mine the difference of TIAGs at adjacent moments. It will be verified in the experimental results in the next section.

5. Performance Evaluation

In this section, we will evaluate the performance of G-TIAG. Specifically, we first introduce the experimental setup and the dataset used for evaluation, then tune the parameters involved in G-TIAG, compare the performance with existing state-of-the-art classification methods, and finally analyze the results.

5.1. Preliminary

(1) Methods in Evaluation: To fully understand the performance of G-TIAG, we leverage four typical methods for comparison, which are briefly described as follows. All the methods are fine-tuned to achieve their best classification performance to gain a fair comparison result.

- Random Forest (RF), is an algorithm that integrates multiple decision trees through integration learning, where each decision tree is a classifier. It uses the 9-dimensional features mentioned in Section 3.2 as baseline features to classify benign data and five types of attack data.
- Support Vector Machine (SVM), which is a generalized linear classifier that performs binary classification with a decision boundary that is the maximum margin hyperplane solved for the learned samples. It also classifies benign data and five types of attack data using the 9-dimensional features.

- Convolutional Neural Networks (CNN), which is a feedforward network including convolutional computation, has a deep structure with good representational learning capability. In addition, it can classify the input in a translation-invariant manner according to its network hierarchy. It also classifies benign data and five types of attack data using the 9-dimensional features obtained in Section 3.2.3.
- Graph Convolution Network (GCN), which learns the graph's structural information and node attributes while considering the correlation between nodes, enabling a more effective and comprehensive characterization of the nodes and the nodes connected to them [25]. The GCN uses the TIAG constructed in Section 3.2 to classify the data.

(2) Validation: We conducted 10 tests for each method, randomly dividing the original dataset into separate training and testing sets in the ratio of 6:4, and the average of the 10 classification results was the result. All experiments are conducted on a server with Intel(R) Core (TM) i7-8650U @ 1.90GHz CPU and GTX1060 GPU, and the programming environment is Python 3.7, PyToch1.10, and anaconda3-5.2.0.

(3) Evaluation Indicators: Since this experiment is faced with a multi-classification problem, it is necessary to select both indicators describing the classification performance of each class of samples and indicators describing the global classification performance. We use precision, recall, and F1-score for each class of samples. We use accuracy for global samples and average the results of each class to obtain the average precision, average recall, and average F1 score. In addition, we introduce a processing rate indicator to evaluate the time cost of the classifier.

- Precision, which is used to describe the proportion of samples predicted to be class A that are class A, where TP_a denotes the number of samples that are class A and predicted to be class A, and FP_a denotes the number of samples that are not class A but predicted to be class A. The specific formula is as follows.

$$Pre_a = \frac{TP_a}{TP_a + FP_a} \tag{15}$$

- Recall, which is used to describe the proportion of samples predicted to be class A that are truly class A, where TP_a denotes the number of samples that are class A and predicted to be class A, and FN_a denotes the number of samples that are class A but predicted not to be class A. The specific formula is as follows.

$$Recall_a = \frac{TP_a}{\mathrm{TP}_a + \mathrm{FN}_a} \tag{16}$$

- F1-score, which is the geometric mean of precision and recall, where Pre_a denotes the precision of class A samples, and $Recall_a$ denotes the recall of class A samples. The specific formula is as follows.

$$F1 - Score = \frac{2 \times Pre_a \times Recall_a}{Pre_a + Recall_a} \tag{17}$$

- Accuracy, which is used to describe the proportion of samples with correct predictions for all categories in the total samples, where y_i denotes whether the prediction of the ith sample is correct and if it is correct, then $y_i = 1$, otherwise $y_i = 0$.

$$ACC = \frac{\sum_{i=1}^{N} y_i}{N} \tag{18}$$

- Processing Rate, which is used to describe the cost of time consumption of the classification method, and the higher the processing rate, the lower the time cost. We specify that the number of input samples processed by the classifier in one second as the processing rate in $s/10^6$.

5.2. Dataset Collection

The CIC-IDS-2017 public dataset, an open dataset collected by the Canadian Institute for Cybersecurity Research, is used in the experiments for analysis, including raw network traffic in PCAP format and labeled feature data in CSV format collected by simulating realistic attack scenarios. The dataset collects normal traffic and attack traffic of 15 types, including DoS, DDoS, Heartbleed, Web attacks, Infiltration, and Botnets. We remove the very small and uncommon types of attacks, such as FTP-Parator, SSH-Parato, Infiltration, and Heartbleed, and select five common types of attacks with sufficient data volume: Dos, DDoS, PortScan, Bot, and Web Attacks. In addition, there is a severe data imbalance problem between benign and attack traffic. To construct a fair classifier, we adjust the ratio between benign and attack data to 10:7 by randomly filtering out benign data. The composition of the dataset is shown in Table 5.

Table 5. Datasets description.

Type	Benign	Dos	DDoS	PortScan	Bot	Web Attack
Number of Flows	103,403	24,742	21,623	24,420	1966	2180

5.3. Parameter Tuning of G-TIAG

This subsection will introduce the hyperparameters used in G-TIAG and evaluate the trade-offs between classification accuracy and efficiency.

5.3.1. Hyperparameter Selection

An important step in training GCNs is to tune the hyperparameters, which adjusts the trade-offs between model bias and classification performance. Because the number of training instances and hyperparameters in G-TIAG is large, it is challenging to find the optimal combination of hyperparameters. Therefore, we search the hyperparameters from an interval and select the best combination.

The final selection of hyperparameters involved in G-TIAG is shown in Table 6, where the learning rate is initialized to 0.001, the training epochs are 200, and the batch size is 64. Next, the GCN involved in G-TIAG uses two convolutional layers for spatial feature extraction, with the number of hidden units in the first layer being 256 and the number of hidden units in the second layer being 128. In addition, the number of hidden units in GRU is 256, the number of hidden units in the self-attention is 256, and the time interval of graph construction is 200.

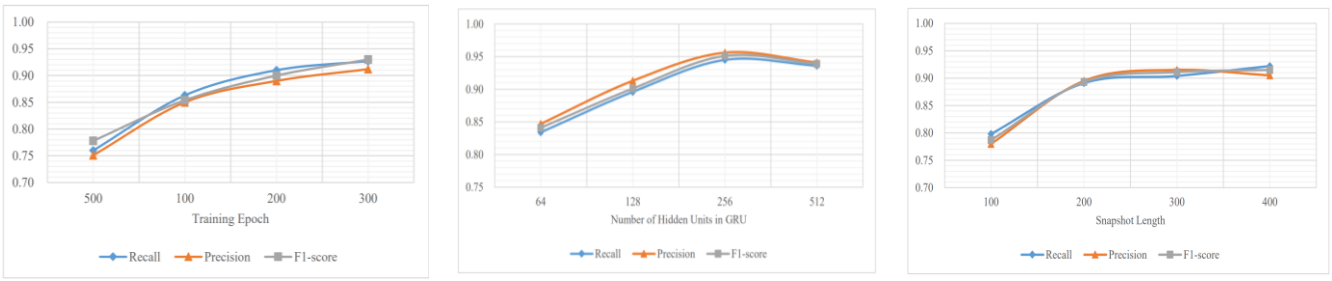
Table 6. Hyperparameters selections for G-TIAG.

Hyperparameters	Adaptive Interval	Final
Training Epochs	[50, 100, 200, 300]	200
Learning Rate	[0.01, 0.005, 0.001, 0.0005]	0.001
Batch Size	[32, 64, 128, 256]	64
Dropout	[0.1, 0.2, 0.4, 0.5]	0.5
GRU Hidden Units	[64, 128, 256, 512]	256
Time Interval of Graph Construction	[100, 200, 300, 400]	200

5.3.2. Epochs

In the attack identification work, we investigate the selection of four epochs separately and explore the change of three indicators under different epochs to spend as little time as possible to achieve a high classification performance, as shown in Figure 7a. We find that G-TIAG achieves precision, recall, and F1-score of about 0.85 in only 100 epochs and about 0.90 in 200 epochs. With the increase in the training epochs, the classification ability generally improves. When the epochs are more extensive than 200, the increase in three indicators from about 0.90 to 0.93, however, is negligible since the training time increases

exponentially. Therefore, we choose 200 epochs to balance classification performance and time cost.



(**a**) (**b**) (**c**)

Figure 7. Parameters selection of G-TIAG. (**a**) Epochs; (**b**) Number of GRU hidden units; (**c**) Time interval of construction.

5.3.3. Number of GRU Hidden Units

When the number of GRU hidden units is too small, the learning capability of the G-TIAG needs to be improved. Correspondingly, with the increase in hidden units, the G-TIAG faces two problems: overfitting and high time cost. In order to find the trade-offs among learning ability, classification performance, and time cost, we select four commonly used choices of the hidden unit number and explore the changes of precision, recall, and F1-Score in the test data, as shown in Figure 7b.

We can see that the G-TIAG classification performance improves significantly as the increase in hidden unit number, and all three indicators are around 0.85 to 0.95. Notably, when the number of units is higher than 256, the G-TIAG shows a stable classification performance on the training set. However, the classification performance starts to decrease slightly on the testing set. The above result is that too many GRU units increase the complexity of G-TIAG, facing the overfitting problem and the decrease in the generalization ability. Therefore, we set the number of GRU units as 256.

5.3.4. Time Interval of Graph Construction

The TIAGs change significantly within a period after the attack occurs, so comparing TIAGs of different periods can identify attacks. Specifically, if the time interval between constructing two TIAGs is too small, the variability between TIAGs is small, which may directly affect the final performance; if the interval is too large, there is the problem of missing attack features. Therefore, we explore the four different time intervals, and the performance of G-TIAG is shown in Figure 7c. As the increase in time interval, the classification performance shows an overall increasing trend from about 0.78 to 0.92. However, the increase is very slight for time intervals from 200 to 400, and we finally choose 200 as the time interval of graph construction.

5.4. Experiment

In this subsection, we investigate the classification ability of G-TIAG for five malicious attack types and the impact of each component on the classification results and compare the classification ability of five classifiers in a closed world.

5.4.1. Classification Performance of G-TIAG

We evaluate the classification capability of G-TIAG using the three indicators of precision, recall, and F1-Score with the optimal parameter selection obtained in Section 5.3. The results are shown in Table 7. The results show that the overall classification capability of G-TIAG is strong. The precision of the three attacks, Dos, DDoS, and PortScan, remains around 0.88 to 0.91, the recall remains around 0.87 to 0.89, and the F1-Score remains

around 0.87 to 0.90. However, compared to the three attacks mentioned above, the Bot and Web Attacks are around 0.70 in the three indicators. We consider that the reason for this problem is data imbalance because the number of pure traffic for both Bot and Web attacks is less than other types, and the amount of data classified into TP is less than others. This issue will be discussed further in the Discussion subsection.

Table 7. Classification results of G-TIAG.

Types	Precision	Recall	F1-Score	Accuracy
Normal	0.91	0.90	0.90	0.94
Dos	0.91	0.89	0.90	0.93
DDoS	0.88	0.88	0.87	0.89
PortScan	0.91	0.87	0.89	0.90
Bot	0.69	0.71	0.70	0.77
Web attack	0.66	0.72	0.69	0.76

To understand the obfuscation relations among the five attack and normal data, we visualize the obfuscation matrix of G-TIAG, as shown in Figure 8. The recall of benign data is 0.9, and the recalls of the five attacks are 0.89, 0.88, 0.87, 0.71, and 0.72, respectively. The experiments show that G-TIAG can effectively distinguish benign data from the five attacks and the Dos, DDoS, and PortScan attacks from other categories of data. However, Bot and Web Attacks are more likely to be misclassified into PortScan and benign data.

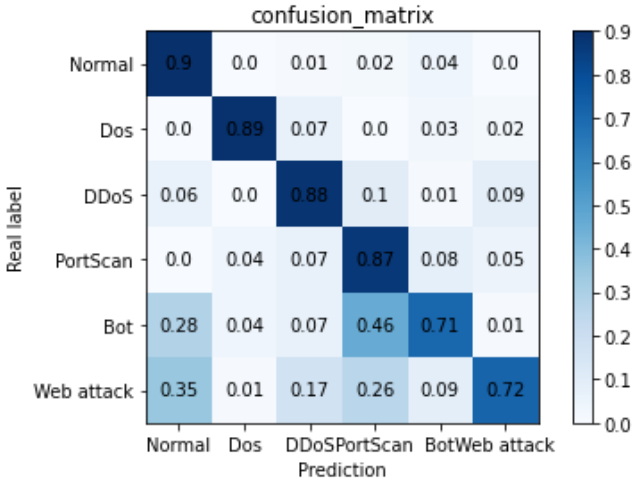


Figure 8. Confusion matrix of G-TIAG.

5.4.2. Ablation Study of G-TIAG

To explore the contribution of each component in G-TIAG to the classification results, we conduct an ablation study for G-TIAG. Specifically, we sequentially remove one component from G-TIAG and analyze the change in classification performance.

The GCN model denotes the model with the GRU component removed from G-TIAG, and the GCN-GRU model denotes the model with the self-attention removed from G-TIAG. The overall classification ability of the three classifiers is evaluated using two indicators, which are accuracy and processing rate. The results are shown in Table 8.

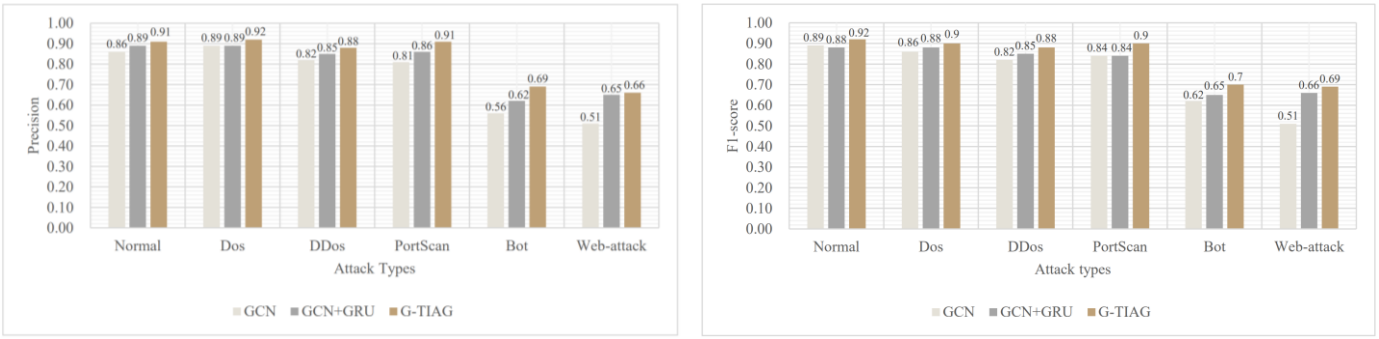
Table 8. Results of ablation study.

Method	GCN	GCN-GRU	G-TIAG
Accuracy	0.8446	0.8625	0.8897
Processing Rate (s/10^6)	2.3558	2.2974	1.9558

The results in Table 8 show a slight decrease in the classification performance when any components are removed from G-TIAG. When the GRU is removed, the accuracy decreases from 0.89 to 0.84, which indicates that the GRU can capture the correlation among longer sequence features and improve the classification results appropriately. When the self-attention is removed, the accuracy slightly decreases from 0.89 to 0.86, which indicates that the self-attention can focus on the more valuable features and fully exploit the correlation features among input instances.

Compared with G-TIAG, the processing efficiency of GCN is substantially improved, and the processing efficiency of GCN-GRU is also improved. The introduction of the GRU and self-attention increases the complexity of the G-TIAG. Therefore, if classification performance is pursued in a practical scenario, devices with higher arithmetic power can be used in exchange for higher classification results.

To detailly understand the performance of the three methods on benign data and five attacks, we choose two indicators for evaluation, which are the accuracy and the F1-Score. The results are shown in Figure 9.



(**a**) (**b**)

Figure 9. Classification results in ablation study. (**a**) Precision in ablation study. (**b**) F1-Score in ablation study.

We find that the classification ability of G-TIAG for benign data and Dos, DDoS, and PortScan has been improved compared with the other two methods. For Bot and Web Attacks, the classification ability has been improved substantially compared with GCN, and the classification ability has been improved marginally compared with GCN-GRU. Therefore, we can conclude from the above work that the GRU component contributes more to the classification results than self-attention.

5.4.3. Performance Comparison

To verify that G-TIAG can classify attacks better than existing methods, we investigate the ability of the four classifiers mentioned in Section 5.1 and the recent study proposed by Yang et al. [22] on the public dataset. We evaluate the classification ability using the average precision, recall, and F1-Score mentioned in Section 5.1, and the results are shown in Table 9.

Table 9. Performance of five classifiers.

Method	RF	SVM	CNN	GCN	RShield	G-TIAG
Average Precision	0.86	0.83	0.81	0.84	0.82	0.89
Average Recall	0.88	0.75	0.77	0.82	0.84	0.88
Average F1-score	0.87	0.77	0.85	0.81	0.83	0.89

Compared with the other four classifiers, G-TIAG improves on all three indicators. The RF method reaches above 0.86 on all three indicators, but the stability of the classification results is found to be lower in the experiments. The recall and F1-Score of the SVM are around 0.75. The recall of the CNN is around 0.77, the GCN reaches about 0.81 in the three indicators, and the G-TIAG model has a stable classification result of about 0.89. Our method outperforms RShield by about 6% in the classification work of multiple types of attacks. We consider this because RShield is better at detecting complex multi-step attacks rather than multi-type attack identification. G-TIAG is better at classifying multiple attacks with unbalanced data because it can learn the difference in interaction patterns of different attacks in a specific period. Moreover, it can also learn the difference in the changes in interaction patterns of different attacks in a time period.

Our method achieves good classification performance based on unbalanced data, including multiple attack types. G-TIAG can learn the differences in interaction patterns from the graph structure and the differences in the changes of interaction patterns at different times. Then, our work discusses the effects of GRU and self-attention on classification performance. The introduction of GRU and self-attention can improve classification performance, and the contribution of GRU is more significant than that of self-attention. We consider that GRU focuses on the correlation between adjacent inputs because the changes between adjacent inputs for different attacks are also different. Finally, after comparing with classifiers and existing studies, we can conclude that G-TIAG has strengths and outperforms other methods in classification work for multiple types of attacks with imbalanced data.

6. Discussion

According to the experimental results, the G-TIAG has better classification performance than the other four methods. However, the identification results of the Bot and Web attack by G-TIAG are much lower than those of the other three attacks. The reason for this problem is the imbalance of different kinds of data. We use the Bayesian formula to illustrate the effect of imbalanced data on classification results. Suppose C_0 is a large class, C_1 is a small class, x is a dataset, and the $P(x|C_0)$ has a similar value to the $P(x|C_1)$.Then $P(C_0|x) = P(x|C_0)P(C_0)/P(x) > P(x|C_1)P(C_1)/P(x)$=$P(C_1|x)$, that is, the probability that the predicted outcome is C_0 is greater than the probability that the predicted outcome is C_1.

Since the objective existence of small but essential attacks in the real networks and the problem of insufficient data for new attacks, there are inevitably significant differences in the amount of data for different attacks, and sampling methods cannot solve the data imbalance. The above problems require the model with full consideration of the few attacks and the algorithm with full consideration of fitting the few data. We use the GRU and self-attention to explore the relations between different data through the weight matrix. The results of the ablation study prove that the classification results can improve the accuracy by 13% and the F1-Score by 8% when using the GRU for Bot, and the accuracy by 7% and the F1-Score by 5% when using the self-attention for Bot Anyway, for the Web attack, the classification results can improve the precision by 15% and the F1-Score by 18% when the GRU is used, and the classification results are stable when self-attention is used.

Therefore, the GRU significantly improves the classification performance on unbalanced data. However, compared to the other three attacks, there is still a long road to refining the model to improve the classification ability for minority data further.

7. Conclusions

Existing studies need to include analyses of the interaction patterns of encrypted malicious traffic. To study the interaction patterns of different attacks, we propose a malicious attack identification method named G-TIAG. It combines spatial features of graphs and statistical features of flows to learn the TIAG structure from large-scale network traffic and represent the differences in interaction patterns of different attacks. Our classification model includes both feature extraction procedure and model construction procedure.

In the feature extraction procedure, we justify using graphs to represent different attacks and propose a rule for the construction of TIAG. Then, considering the statistical features of flows, we select nine attributive features for each node in TIAG. Last, we quantitatively demonstrate the effectiveness of TIAG. The results show that TIAG can effectively represent benign data, DDoS, DoS, and Portscan data. However, TIAG is less capable of representing Bot and Web attacks because there are fewer data in these two categories.

In the model construction procedure, the classification of attacks is transformed into the classification of graphs by TIAG. G-TIAG captures the spatial features using GCN, the temporal features using GRU, and the differences in the importance of temporal features using self-attention. Furthermore, our model accurately identifies malicious traffic by using TIAGs as the input of the GCN model. Experimental results show that G-TIAG can reach good classification effects and perform better than other methods.

In the future, we will further work on the identification method for malicious attack types with fewer data. In addition, to address the lack of data for some attack categories, we can focus more on the one-class classifier to identify only normality in the future and improve the adaptability of classification methods in the real network.

Author Contributions: Conceptualization, G.R. and G.C.; methodology, G.R.; software, N.F.; validation, G.R.; formal analysis, G.R.; investigation, N.F.; resources, G.R.; data curation, G.R.; writing—original draft preparation, N.F.; writing—review and editing, N.F.; visualization, N.F.; supervision, G.C.; project administration, G.C.; funding acquisition, G.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the General Program of the National Natural Science Foundation of China under grant number 62172093.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Cisco, Cisco Annual Internet Report (2018–2023) White Paper[EB/OL]. Available online: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed on 1 September 2020).
2. Ministry of Industry and Information Technology of the People's Republic of China, Communications Industry Statistics Bulletin 2021[EB/OL]. Available online: https://wap.miit.gov.cn/gxsj/tjfx/txy/art/2022/art_e8b64ba8f29d4ce18a1003c4f4d88234.html (accessed on 1 September 2021).
3. Desai, D. "30,000 Percent Increase in COVID-19-Themed Attacks"[EB/OL]. Available online: https://www.zscaler.com/blogs/security-research/30000-percent-increase-covid-19-themed-attacks (accessed on 1 September 2020).
4. CyberEdge, 2021 Cyberthreat Defense Report[EB/OL]. Available online: https://cyberedge.com/wp-content/uploads/2021/04/CyberEdge-2021-CDRReport-v1.1-1.pdf (accessed on 1 September 2021).
5. Wang, J.; Rossell, D.; Cassandras, C.G.; Paschalidis, I.C. Network anomaly detection: A survey and comparative analysis of stochastic and deterministic methods. In Proceedings of the 52nd IEEE Conference on Decision and Control, Firenze, Italy, 10–13 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 182–187.
6. Doriguzzi-Corin, R.; Millar, S.; Scott-Hayward, S.; Martinez-del-Rincon, J.; Siracusa, D. LUCID: A practical, lightweight deep learning solution for DDoS attack detection. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 876–889. [CrossRef]
7. Catillo, M.; Rak, M.; Villano, U. Discovery of DoS attacks by the ZED-IDS anomaly detector. *J. High Speed Netw.* **2019**, *25*, 349–365. [CrossRef]
8. Liu, C.; He, L.; Xiong, G.; Cao, Z.; Li, Z. Fs-net: A flow sequence network for encrypted traffic classification. In Proceedings of the IEEE Infocom 2019-IEEE Conference On Computer Communications, Paris, France, 29 April–2 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1171–1179.
9. Abou Daya, A.; Salahuddin, M.A.; Limam, N.; Boutaba, R. A graph-based machine learning approach for bot detection. In Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Washington, DC, USA, 8–12 April 2019; pp. 144–152.

10. Yao, Y.; Su, L.; Lu, Z. DeepGFL: Deep feature learning via graph for attack detection on flow-based network traffic. In Proceedings of the MILCOM 2018—2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 579–584.
11. Mirsky, Y.; Doitshman, T.; Elovicim, Y.; Shabtai, A. Kitsune: An ensemble of autoencoders for online network intrusion detection. *arXiv* **2018**, arXiv:1802.09089.
12. Ring, M.; Dallmann, A.; Landes, D.; Hotho, A. Ip2vec: Learning similarities between ip addresses. In Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW), New Orleans, LA, USA, 18 November–21 November 2017; pp. 657–666.
13. Han, X.; Yin, R.; Lu, Z.; Jiang, B.; Liu, Y.; Liu, S.; Wang, C.; Li, N. STIDM: A Spatial and Temporal Aware Intrusion Detection Model. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December 2020–1 January 2021; pp. 370–377.
14. Xiao, Y.H.; Xing, C.; Zhang, T.N.; Zhao, Z.K. An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks. *IEEE Access* **2019**, *7*, 42210–42219. [CrossRef]
15. Wang, J.; Paschalidis, I.C. Botnet detection based on anomaly and community detection. *IEEE Trans. Control. Netw. Syst.* **2016**, *4*, 392–404. [CrossRef]
16. Tian, S.; Wang, H.; Li, S.; Wu, F.; Chen, G. Trajectory-based multi-hop relay deployment in wireless networks. In Proceedings of the International Conference on Combinatorial Optimization and Applications, Shanghai, China, 16–18 December 2017; Springer: Cham, Switzerland, 2017; pp. 111–118.
17. Protogerou, A.; Papadopoulos, S.; Drosou, A.; Tzovaras, D.; Refanidis, I. A graph neural network method for distributed anomaly detection in IoT. *Evol. Syst.* **2021**, *12*, 19–36. [CrossRef]
18. Khalaf, O.I.; Ogudo, K.A.; Sangeetha, S.K.B. Design of Graph-Based Layered Learning-Driven Model for Anomaly Detection in Distributed Cloud IoT Network. *Mob. Inf. Syst.* **2022**, *2022*, 6750757. [CrossRef]
19. Do Xuan, C.; Huong, D.T. A new approach for APT malware detection based on deep graph network for endpoint systems. *Appl. Intell.* **2022**, 1–20. [CrossRef]
20. Liu, X.; Ren, J.; He, H.; Zhang, B.; Song, C.; Wang, Y. A fast all-packets-based DDoS attack detection approach based on network graph and graph kernel. *J. Netw. Comput. Appl.* **2021**, *185*, 103079. [CrossRef]
21. Wang, S.; Wang, Z.; Zhou, T.; Sun, H.; Yin, X.; Han, D.; Zhang, H.; Shi, X.; Yang, J. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*. [CrossRef]
22. Yang, W.; Gao, P.; Huang, H.; Wei, X.; Liu, W.; Zhu, S.; Luo, W. RShield: A Refined Shield for Complex Multi-step Attack Detection Based on Temporal Graph Network. In *International Conference on Database Systems for Advanced Applications*; Springer: Cham, Switzerland, 2022; pp. 468–480.
23. CAIDA, CAIDA Data Completed Datasets[DB/OL]. Available online: https://www.caida.org/catalog/datasets/completed-datasets/ (accessed on 25 November 2022).
24. Jiang, B.; Zhang, Z.; Lin, D.; Tang, J.; Luo, B. Semi-supervised learning with graph learning-convolutional networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 11313–11320.
25. Zhao, L.; Song, Y.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; Li, H. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 3848–3858. [CrossRef]