mathematics

MDPI

Article

AGCN-Domain: Detecting Malicious Domains with Graph Convolutional Network and Attention Mechanism

Xi Luo [1,†], Yixin Li [2,†], Hongyuan Cheng [1,†] and Lihua Yin [1,*]

[1] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China; luoxi0930@gzhu.edu.cn (X.L.); hongyuancheng866@gmail.com (H.C.)
[2] Big Data Center of State Grid Corporation of China, Xicheng District, Beijing 100052, China; yixin__li@163.com
* Correspondence: yinlh@gzhu.edu.cn
† These authors contributed equally to this work.

Abstract: Domain Name System (DNS) plays an infrastructure role in providing the directory service for mapping domains to IPs on the Internet. Considering the foundation and openness of DNS, it is not surprising that adversaries register massive domains to enable multiple malicious activities, such as spam, command and control (C&C), malware distribution, click fraud, etc. Therefore, detecting malicious domains is a significant topic in security research. Although a substantial quantity of research has been conducted, previous work has failed to fuse multiple relationship features to uncover the deep underlying relationships between domains, thus largely limiting their level of performance. In this paper, we proposed ***AGCN-Domain*** to detect malicious domains by combining various relations. The core concept behind our work is to analyze relations between domains according to their behaviors in multiple perspectives and fuse them intelligently. The ***AGCN-Domain*** model utilizes three relationships (client relation, resolution relation, and cname relation) to construct three relationship feature graphs to extract features and intelligently fuse the features extracted from the graphs through an attention mechanism. After the relationship features are extracted from the domain names, they are put into the trained classifier to be processed. Through our experiments, we have demonstrated the performance of our proposed ***AGCN-Domain*** model. With 10% initialized labels in the dataset, our ***AGCN-Domain*** model achieved an accuracy of **94.27%** and the F1 score of **87.93%**, significantly outperforming other methods in the comparative experiments.

Keywords: malicious domain; graph convolutional network; attention mechanism; domain relations

MSC: 68T07

check for updates

Citation: Luo, X.; Li, Y.; Cheng, H.; Yin, L. AGCN-Domain: Detecting Malicious Domains with Graph Convolutional Network and Attention Mechanism. *Mathematics* **2024**, *12*, 640. https://doi.org/10.3390/math12050640

Academic Editor: Antanas Cenys

Received: 3 February 2024
Revised: 16 February 2024
Accepted: 19 February 2024
Published: 22 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The Domain Name Resolution System (DNS) is one of the key infrastructures of today's Internet, which provides domain name resolution services for Internet users and enables the mapping of domain names to IPs. However, due to the openness of DNS services, cyber attackers also rely on DNS services to carry out attacks by registering a large number of domain names. These malicious domains can lead to web pages that the attacker has carefully designed to conduct crimes such as stealing accounts, fraud, and infecting computers with viruses (ransomware, Trojans, etc.). These types of web pages are almost identical to normal web pages, thus reducing the vigilance of ordinary users and providing attackers with opportunity. In addition, some malicious software, such as Trojans and backdoor viruses, connect to C&C servers through domain names (C&C servers act as bridges for communication between attackers and infected hosts. Attackers send instructions to compromised hosts through C&C servers, enabling lateral movement to infect other internal hosts or steal sensitive data). These forms of malicious software use fast-flux technology to evade detection, making it increasingly difficult to detect malicious domains. Therefore, the detection of malicious domains has attracted the attention of many researchers.

Researchers have proposed a large number of works to discover malicious domains. Traditional approaches can be classified as *feature-based methods* [1–12]. They usually extract features provided by security experts for each domain from DNS data [1–7]. Typical features include lexical features, whois features, resolution features, etc. They recognize malicious domains with trained classifiers. However, such methods can be evaded easily since most of them are based on features of an individual domain. In addition, they are limited in expert observation and knowledge with manual prejudice.

In the face of these problems, some researchers have turned to the use of relational features between domain names for malicious domain name detection. They built various graphs to describe relations like *client relations* [13,14], *resolution relations* [15], *whois relations* [16], etc. And then they discovered malicious domains based on the intuition that domains strongly related to known malicious domains are more likely to be malicious. However, in some studies they focused on only one type of relationship between domains, which left a great deal of information unexplored and failed to reveal the richer potential relationships between domains, which allowed experienced attackers to evade detection by this type of system. Some studies utilized multiple relationships [17–19] to calculate domain name similarity through relationships [17] or to discover domain names strongly associated with malicious domains through multiple mete-paths [18]. But these works ignored the fine-grained distinctions between relationship features and ceased to address the different effects of different relationship features on domain name judgements with implications for the accuracy of the system.

To address the problems with the above two main ways of detecting malicious domain names, it is necessary to develop a more comprehensive malicious domain name detection system. Graph Convolutional Neural Network is a good solution in solving graph-based correlation problems, and it also has good performance in collaborative classification, machine account detection, etc.

In this paper, we made a basic observation about the relational characteristics of domain names that domain names with strong associations are more likely to belong to the same category [15]. This is because the attacker's attack resources are limited due to cost, making the adversary reuse their attack resources, leading to a strong correlation of malicious domains in the relationship [18]. Based on the above basic observation, we propose *AGCN-Domain*, a system that detects malicious domains by leveraging various relations and balancing their influence with the attention mechanism to solve the above limitation. In the *AGCN-Domain* model, firstly, we used three homogeneous subgraphs: **client relation graph**, **resolution relation graph**, and **cname relation graph** to represent the relationship features between domains and pre-process the subgraphs to optimize the subgraph structure. Graph convolution neural networks were implemented to aggregate features in each of the three subgraphs. Second, to balance the impact of different relationship features on domain features, we introduced an attention mechanism to make the detection results more accurate. We utilized a soft attention mechanism to intelligently assign weight coefficients to the relational features of the domain converged on the three subgraphs, thus obtaining a comprehensive relational feature vector of the domain. Finally, a fully connected layer was used to output the malicious/benign probability for the domain.

The contributions can be summarized as follows:

- We proposed *AGCN-Domain*, a system for detecting malicious domain names using multiple relationship patterns: Client relation, resolution relation, and cname relation, which could extract and fuse the relationship features of domain names. The malicious domain names detection problem was transformed into a node binary prediction problem.
- We designed a mechanism to detect malicious domains with high accuracy. We introduced a graph convolutional neural network, which performed well in graph correlation tasks to detect malicious domains. We integrated the graph convolutional

neural network with an attention mechanism to intelligently blend the effects of different relationship features on the classification results of domains for different types of domains.

- We made a comprehensive evaluation of our work with real-world data collected from an educational network. The results demonstrate that it has good performance in detecting malicious domains.

Organization. The rest of this paper is organized as follows. Section 2 reviews related work and its limitations. Section 3 illustrates preliminaries of our system, and Section 4 introduces the system structure and major components. Then, we make a comprehensive evaluation in Section 5. Finally, Section 6 concludes this paper.

2. Related Work

With many important breakthroughs in the application of deep neural network models to graphs, graph-based methods have attracted more and more attention from scholars in many research areas. For example, to better analyze large-volume and fast-changing streaming graphs, Gu and Jiang, et al., proposed a method to efficiently find the graph stream summary, PGSS-MDC [20], to address the problem of providing highly accurate and fast answers to graph queries within a certain time interval. Meanwhile, the use of graph-based methods to detect network intrusions have also become an important research direction; for example, to to detect the existence of multi-step attacks in a large number of security alarms in smart city application scenarios, Jia, et al., proposed the ACAM network security analysis framework [21], which is a network security analysis framework based on the Knowledge Representation Model (Multi-dimensional Data Association and Intelligent Analysis Model [22]). The model constructs a large-scale multidimensional association knowledge graph (MDATA graph) through existing security analysis reports and databases, then converts the detected network attack rules into MDATA subgraphs and generates real-time alarm MDATA graphs by extracting information from each alarm in the alarm flow, and finally detects multi-step attacks through a subgraph matching algorithm.

Detecting malicious domain names is one of the important research directions in network security, and researchers have proposed many approaches to recognize malicious domains. These works can be divided into two categories: feature-based methods and relation-based methods.

2.1. Feature-Based Methods

The typical approach taken by this research is to observe the characteristics of domains from multiple perspectives and extract related features according to security experts' observations and knowledge. Then, a classifier, usually a supervised one, is used to recognize malicious domains with these features.

Earlier works usually used manual features from many angles. Antonakakis, et al. [1] proposed a system named Notos for dynamically assigning reputation scores to unknown domains, which extracts relevant statistical features to calculate the reputation scores of unknown domains, including Network-based features(The number of IPs associated with domain x, the number of autonomous systems in which the IPs to which domain x refers, etc.), Zone-based features(Average length of domains associated with domain x, number of different TLDs, etc.) and Evidence-based features(The number of malware linked to domain x, etc.) Similar to previous works, Bilge et al. [2,3] presented a system named Exposure to identify domains that are involved in malicious network campaigns like phishing and malware spam, etc. They extracted 15 time-based, answer-based, TTL-based, and name-based features from passive DNS data and classified malicious domains with the decision tree algorithm. Based on the work of Antonakakis, et al. [4], Chinba, et al. [5] proposed the concept of "temporal change patterns" (TVPs) from the perspective of domain name behavior and proposed the DomainProfiler model. The "time-varying model" defines the changes in the black/white list of a domain name in a certain time window as four pattern features (Null, Fall, Rise, Stable), which are combined with the Network-based features and Zone-based features proposed in the Notos system to form feature vector of the target domain name. Finally, the decision tree algorithm was used to predict the classification of the target domain name.

Most of the aforementioned malicious domain detection systems are passive in detection, and abnormal domains are only detected after malicious activity has begun, therefore delaying the detection of anomalies. By the time the malicious domain is detected, it has already conducted malicious activities for a long time. To solve this problem, Hao et al. [6] proposed the PREDATOR active reputation system, which identifies malicious domains at registration. In addition to utilising basic domain information features, the PREDATOR system also incorporates domain registration history features and Batch Correlation features, verifying the registering agent for re-registered domains, the dormant time of re-registered domains, the different lifecycle ratios of domains registered in the same batch, and the possibility of registering a large number of domains in batches. Finally, the Convex Polytope Machine supervised learning algorithm was used for classification.

In addition to using basic information about the domain name and the IPs associated with the domain name as features, there are several studies that have focused on detecting the lexical distribution of algorithm-generated domains. They generally analyzed domain strings and extracted lexical features from multiple perspectives such as structure, linguistics, pronunciation, and statistic [7–9] to detect DGA domains. Recently, some researchers have leveraged a deep learning model to extract lexical features automatically. Woodbridge, et al. [10] proposed a malicious domain detection system with LSTM (long short-term memory) network. The system takes domain name string sequence as input and extracts character features with the LSTM model intelligently. On the basis of this, Tran, et al. [11] presented the LSTM-MI model, which combines both binary and multiclass classification models to deal with the multiclass imbalance problem in LSTM. Xu, et al. [12] combined an N-gram and a convolutional neural network to extract lexical features. The model they proposed takes the domain itself as input and automatically estimates whether the domain is generated by DGA malware.

2.2. Graph-Based Methods

Approaches in this category usually leverage graphs to describe diverse relations among domains and then infer malicious domains with labeled nodes and relationships.

Early research discovered malicious domains with only one relation. Manadhata, et al. [14] depicted client relations among domains with a client-domain bipartite graph. Then, malicious domains were detected using the belief propagation algorithm. Khalil, et al. [15] came up with resolution relations between domains with the domain–domain graph, and then calculated the malicious score for each unknown node according to their relations with known malicious domains. Building on the work of Khalil, et al. [15], Lee, et al. [23] analyzed the sequential correlation of domains and further generated a domain–travel graph to detect malicious domains. Unlike the above work focusing on the resolution relationships between domain names, Peng, et al. [24] focused on cname records. They constructed a domain–domain graph based on their cname relations and inferred whether an unknown node was malicious or not with the belief propagation algorithm.

As research progressed, researchers found that combining multiple relationships revealed more underlying semantics and deeper relationships, making the detection of malicious domain names more effective. Zou, et al. [19] constructed a client–domain–IP graph and leveraged a propagation algorithm to discover malicious nodes. Najafi, et al., instead proposed the MalRank system [25], which uses information collected by an enterprise's internal SIEM, derived from, for example, proxy, DNS, and DNCP logs, to construct a knowledge graph that identifies malicious entities by calculating and iterating a malicious score over the entities in the knowledge graph. Lei, et al. [17] built three domain relation graphs and leveraged the graph embedding technique to obtain features. For Sun, et al., to derive more information and discover deeper semantics, a DNS scenario was recreated by constructing a heterogeneous information network, and six meta-paths were used to combine the heterogeneous information network to identify malicious domain nodes [16,18].

2.3. Discussion

Feature-based works have mainly depended on hand-crafted features that are manually extracted by experts' observation. Researchers have collected DNS data (e.g., DNS traffic, DNS logs, passive DNS information, etc.) and other supplemental data (e.g., whois data, geographic information for IP, etc.) and analyzed malicious network activities. Summarized features can effectively distinguish malicious domains. However, these works are inevitably limited by experts' observation and knowledge. In addition, they can be evaded easily by attackers since features are usually for one individual domain. For instance, nearly all lexical features may lose efficacy if attackers use new technology like GAN to avoid detection [26–28]. Graph-based works leverage multiple relations among domains based on the basic observation that attackers reuse attack resources when limited. Recent works have tended to combine multiple relations to obtain a more accurate result. However, these works treat various relations equally and ignore their different influences. For example, resolution relations are usually more important than client resolution in malicious domain detection.

In this paper, we solved the above problems with a graph convolution network (GCN) and attention mechanism. Specifically, we derived domain features intelligently with the GCN model to avoid manual prejudice and combined multiple relations with the attention mechanism to balance the influence of different relations. Our work can generate a more comprehensive analysis using this method.

3. Preliminaries

In this section, we first introduce the relations we used in this work and the reason why they are effective in detecting malicious domains. Then, we summarize the techniques used in our work.

3.1. Domain Relations

We investigated previous works detecting malicious domains utilizing domain relations and summarized three typical relations that are widely used: **client relation**, **resolution relation**, and **cname relation**.

Client Relation [13,14]. The client relation indicates that a domain queried by a client that has requested malicious domains is more likely to be malicious. There are two basic observations of the client relation. (1) Compromised clients are unlikely to query only one malicious domain since they commonly query a set of domains to achieve one malicious activity, and they are usually involved in more than one attack. (2) Malicious domains have no reason to be accessed by benign clients since they are dedicated to malicious services. As shown in Figure 1, domain *d6* can be inferred to be malicious because it is requested by suspicious clients c1, c3, c5 and clearly far away from the benign network: no benign clients have queried it.

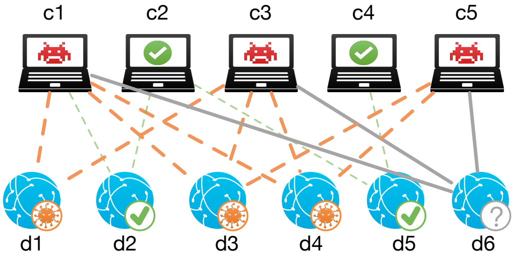


Figure 1. Client Relations.

Resolution Relation [15]. The resolution relation means that domains with the same resolutions (hosted on the same IP) tend to have the same labels. This relation is strongly based on the rarity and expensiveness of malicious servers (IPs) where attackers can host domains. In fact, attackers widely reuse resources (especially malicious IPs) in malicious activities, such as domain-flux and fast-flux. As shown in Figure 2, domain *d5* is considered to be malicious, because it is hosted on suspicious servers *IP1*, *IP2*, which host three known malicious domains *d1*, *d2*, *d3* and no benign domains.

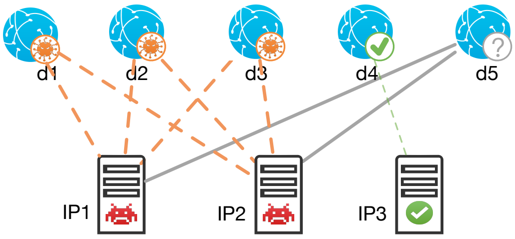
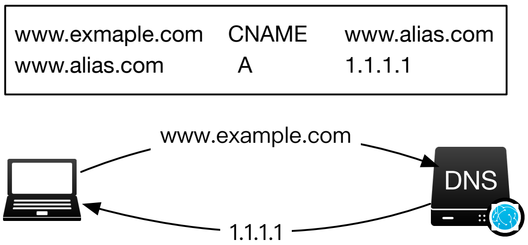


Figure 2. Resolution Relations.

Cname Relation [24]. The cname relation in this paper indicates that domains that appear in one cname record have a homophilic state. In the DNS system, the cname record is used to set the alias name of domains. We illustrate an example of a cname record in Figure 3. When setting a cname record, the domain and its alias name finally point to the same destination: the client first queries *www.example.com* and obtains its alias name *www.exmaple-alias.com*, then the client queries *www.exmaple-alias.com* and obtains the final destination. It is obvious that domains connected to malicious domains through a cname relation are more likely to be malicious since they map to the equal resource.



www.exmaple.com	CNAME	www.alias.com
www.alias.com	A	1.1.1.1

Figure 3. Cname Relations.

3.2. Graph Convolutional Network (GCN)

Nowadays, graph convolutional networks (GCN) have yielded unusually brilliant results in multiple areas, such as document classification, recommender systems, traffic prediction, etc. GCN was first proposed by Kipf and Welling [29] in 2017. The GCN model takes both node attributes and graph structures as input and combines them to obtain the deep semantics of nodes. The propagation rule of multi-layer GCN can be expressed as:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{1}$$

where A is the adjacency matrix of graph, σ is a nonlinear active function, $W^{(l)}$ is the trainable weight matrix of lth layer, and $H^{(l)}$ is the activations matrix in lth hidden layer.

4. Method

4.1. Overview

As shown in Figure 4, *AGCN-Domain* is an intelligent malicious domain detection system with three main components: data preprocessor, relation graph constructor, and attention-GCN classifier. First, in the data preprocessor, we extracted four basic fields from raw DNS logs: domain, IP, client, and timestamp to represent domain behaviors in subsequent steps. Then, in the relation graph constructor, we constructed three types of relation graph of each domain, respectively: *client relation graph*, *resolution relation graph* and *cname relation graph*. Among them, *client relation graph* and *resolution relation graph* are mapped from two bipartite graphs while *cname relation graph* is generated from type cname records. Finally, we designed an attention-GCN classifier to mine deep features of domains by fusing information from multiple relation graphs to recognize malicious domains.

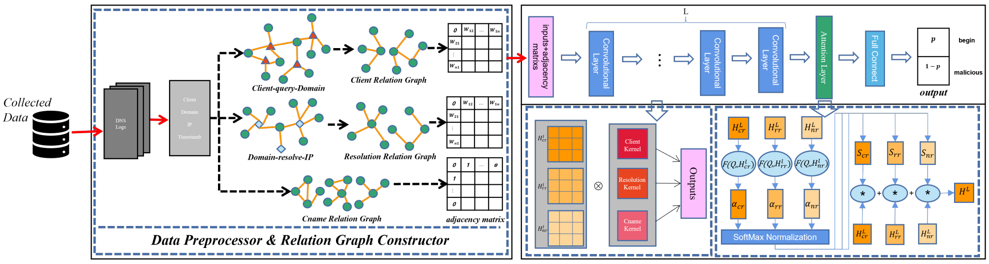


Figure 4. Architecture of *AGCN-Domain*.

It should be noted that *AGCN-Domain* is a universal framework that is not limited to the three types of relations mentioned in this paper and can be extended to far more relations among domains.

4.2. Data Structure

The DNS record dataset we collected includes the following fields: **timestamp string**, **source IP**, domain name server IP, protocol name, **requested domain name**, request record type, **answers**, and TTLs rejected.

4.3. Data Preprocessor

The main purpose of this step is to formalize dirty raw data and improve system efficacy. We took DNS logs as input and focused only on cname and A records. We filtered out records that meet filtering rules and extracted four fundamental fields (requested domain name, domain-resolve-IP, source IP, timestamp string) from the remaining data. Specific filtering rules are as follows:

- Corrupted records. There are some corrupted records from transmission errors in collected raw data, such as an incomplete record missing some fields.
- Irregular domains. There are some irregular domains in the original data, which can be divided into two categories. One is that domains do not comply with domain naming rules, which is probably due to mistyping or misconfiguration, for example, containing commas in strings like *youtube,com*. The other is that the domains whose TLD (Top Level Domain) are not registered in IANA, which means that they are invalid Internet domains.

4.4. Relation Graph Constructor

In this step, we constructed three relation graphs, client relation graph, resolution relation graph, and cname relation graph, to describe relationships among domains.

4.4.1. Client Relation Graph

Client relation actually indicates that two domains share the same clients within a time window. In order to model such a relation, we first built a bipartite graph among clients and domains, and further generated the client relation graph.

The client–query–domain bipartite graph can be expressed as $G_{cd} = (C, D_t, E)$, where C and D_t represent all clients in the network and all domains (with timestamps) they queried, respectively, E is a set of edges between domain and client. For instance, if client c_i requests domain d_j at time t, then an edge $< c_i, d_j(t) > \in E$ exists to describe this behavior in the client–query–domain graph.

For depicting client relation among domains, we transformed the above client–query–domain graph into a client–relation graph $G_cr = (D, E, W)$, where D and E are the set of domains and edges, and W represents weights of edges. If domain d_i and d_j share at least one client within one time window, there is an edge $< d_i, d_j > \in E$. The weight of $< d_i, d_j >$ can be calculated based on Jaccard Similarity:

$$W(d_i, d_j) = \frac{|\bigcup_{t=1}^{n}(\mathbb{C}_i^t \cap \mathbb{C}_j^t)|}{|\mathbf{C}_i \cup \mathbf{C}_j|} \tag{2}$$

In the above formula, $\mathbb{C}_i^t$ and $\mathbb{C}_j^t$ are sets of clients that have queried domain d_i and d_j within time window t, respectively. $\mathbf{C}_i$ and $\mathbf{C}_j$ are sets of clients that have queried domain d_i and d_j in the network data, respectively.

4.4.2. Resolution Relation Graph

Resolution relation here indicates that two domains share the same IPs in the whole network data. To model such a relation, we first built a bipartite graph among domains and resolved IPs and further generated the resolution relation.

The domain–resolve–IP bipartite graph can be expressed as $G_{di} = (D, P, E)$, where D and P represent all domains in the network and all IPs where they once hosted respectively, and E is a set of edges between domain and IP. For instance, if domain d_i once hosts on IP P_j, then there is an edge $< d_i, p_j) > \in E$ to describe this behavior in domain–resolve–IP graph.

For depicting resolution relation among domains, we transformed the above domain-resolve-IP graph into a resolution relation graph $G_{rr} = (D, E, W)$, where D and E are the set of domains and edges, W represents weights of edges. If domain d_i and d_j share at least one resolved IP, then there is an edge $< d_i, d_j > \in E$. The weight of $< d_i, d_j >$ can be calculated based on Jaccard Similarity as follows:

$$W(d_i, d_j) = \frac{|\mathbf{P}_i \cap \mathbf{P}_j|}{|\mathbf{P}_i \cup \mathbf{P}_j|} \tag{3}$$

In the above formula, $\mathbf{P}_i$ and $\mathbf{P}_j$ are sets of IPs hosting domain d_i and d_j, respectively.

4.4.3. Cname Relation Graph

Cname relation represents that domains belong to one cname record. Unlike the above two relation graphs, the *Cname Relation Graph* can be generated from raw DNS data directly.

The cname relation graph $G_{nr} = (D, E)$ is an unweighted graph, where D and E are the set of domains and edges. Edge $< d_i, d_j >$ represents that domain d_i and d_j were once shown in one cname record.

4.4.4. Graph Pruner

Since local network data are noisy with massive domains, IPs, and clients, we deleted some nodes that cannot provide useful information from client–query–domain graph, domain–resolve–ip graph, and cname relation graph to increase *AGCN-Domain*'s performance and efficacy. We investigated former works [13,18] and set pruning rules as follows:

- Popular domains. The basic intuition is that domains that have been queried by more clients are more likely to be legitimate. The typical example is that famous domains, such as google.com, can be queried by nearly all clients in the monitored local network. Processing such famous popular domains will take a lot of resources; thus, we pruned them to increase the efficacy of the system. A popular domain was defined as requested by more than 25% of clients.
- Hyperactive clients. In our data, there are some very active clients that can query domains even 1,000,000 times one day. We analyzed them and found that they are proxies or forwarders: there may be hundreds of clients behind source IP. Such clients cannot provide valid client relation for domains, so we deleted them. We set the top 0.1% clients as hyperactive clients and removed them.

- Inactive clients. There are some clients that query only a few domains. Such clients also cannot offer much information; thus, we set a threshold of N_{ic} and removed clients querying fewer domains than this. The N_{ic} was set to 2 in our experiment.
- Inactive IPs. The same as inactive clients, we erased IPs that host only one domain in our network data.
- Exceptions. Similar to previous work [18], we kept malicious domains and their related information even when they complied with the above rules, considering that malicious domains usually are inactive to avoid detection.

4.5. Attention-GCN Classifier

Model. In this step, we took three graphs that described relations between domains from different views, as input and output are deep features for each domain. Considering that diverse relations have different influences for detecting malicious domains, we applied GCN with an attention mechanism in our model. The hidden layer is designed as follows:

$$H_{cr}^{l+1} = \sigma(\mathcal{L}_{cr} H_{cr}^{l} W_{cr}) \tag{4}$$

$$H_{rr}^{l+1} = \sigma(\mathcal{L}_{cr} H_{rr}^{l} W_{rr}) \tag{5}$$

$$H_{nr}^{l+1} = \sigma(\mathcal{L}_{cr} H_{nr}^{l} W_{nr}) \tag{6}$$

$$H^L = \sum_{i \in (cr,rr,nr)} softmax(\alpha_i) \odot H_i^L \tag{7}$$

$$\mathcal{L} = D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}} \tag{8}$$

Formulas (4)–(6) extract features of relation graphs, and final hidden features can be calculated by combining them with attention mechanism as shown in Formula (7), where H_i^l denotes the feature convergence process of the lth convolutional layer on the relation graph i, $H_i^l \in \mathbb{R}^{n_v^i \times d_v}$, n_v^i denotes the number of nodes in relation graph i, $i \in \{cr, rr, nr\}$, $H_i^0 = (X)$, and X denotes the initial embedding vector of the node, $X \in \mathbb{R}^{d_v}$, $\mathcal{L}_{cr}$, $\mathcal{L}_{rr}$, $\mathcal{L}_{nr}$ denote the normalized Laplacian matrices of the three graphs, respectively, A and D represent the adjacency matrix and degree matrix of the subgraph, respectively, I is the identity matrix, $\mathcal{L}_{cr}$, $\mathcal{L}_{rr}$, $\mathcal{L}_{nr} \in \mathbb{R}^{n_v^i \times n_v^i}$, and W_{rr}, W_{cr}, W_{nr} denote the trainable weight matrices on three relational graphs, W_{rr}, W_{cr}, $W_{nr} \in \mathbb{R}^{d_v \times d_v}$, and $\sigma(\cdot)$ represents ReLU activation function.

α_i is used as the attention score of the H_i^L, $i \in \{cr, rr, nr\}$, which is determined by the similarity between the hidden features of domain nodes and the $\bar{h}_i$ vector, which is the mean of the H_i^L of the domains. α_i is calculated as follows:

$$\alpha_i = \begin{bmatrix} \alpha_{i1} \\ \alpha_{i2} \\ \vdots \\ \alpha_{in} \end{bmatrix} \tag{9}$$

$$\alpha_{ik} = -Similarity\left(h_{ik}^L, \bar{h}_i\right) = -\frac{h_{ik}^L \cdot \bar{h}_i^t}{\|\bar{h}_i\| \cdot \|h_{ik}^L\|} \tag{10}$$

$$\bar{h}_i = \frac{\sum_{v \in S_i} h_i^L(v)}{|S_i|} \tag{11}$$

where S_i is the set of nodes of relation graph i.

Isolated points. In each relation graph, it is inevitable to have isolated nodes that have no edges with any other nodes, because it is impossible for all domains to have three relations with others in network data. Such nodes will be fairly negative to our model. Thus, we set one node's related hidden vectors to zero if it was an isolated point in a graph to minimize the negative. For instance, the hidden vector $h^{l}_{cr}(d)$ of domain d in layer l, which was an isolated node in client relation, was set to zero (Algorithm 1).

Algorithm 1 Attention-GCN Classifier

```
Input: client relation graph $G_{cr}$; resolution relation graph $G_{ar}$; cname relation graph $G_{nr}$;
    Parameters: embedding size $k$; depth of layer $L$;
Output: feature vectors $V$of nodes (domains)
Initialize $H^{(0)}$
$\mathcal{L}_{cr} \leftarrow Normalized\ AdjacentMatrix(G_{cr})$
$\mathcal{L}_{rr} \leftarrow Normalized\ AdjacentMatrix(G_{rr})$
$\mathcal{L}_{nr} \leftarrow Normalized\ AdjacentMatrix(G_{nr})$
for $l$ in range $(0, L-1)$ do
    $H^{l+1}_{cr} = \mathcal{L}_{cr} H^{l}_{cr} W_{cr}$
    $H^{l+1}_{rr} = \mathcal{L}_{rr} H^{l}_{rr} W_{rr}$
    $H^{l+1}_{nr} = \mathcal{L}_{nr} H^{l}_{nr} W_{nr}$
end for
$H^{L} = \sum_{i \in (cr,rr,nr)} softmax(\alpha_i) H^{L}_{i}$
$V = FC(H^{L})$
return $V$;
```

Loss Function For each domain, we obtained a $k-length$ vector, which represented its deep features after the above processes. Then, distinguishing malicious domains could be regarded as a binary classification problem. Thus, a fully connected layer was set to judge whether a domain was malicious or not. Naturally, the training objective was to minimize the gap between our predicted results and labels of known nodes. Therefore, the loss could be measured by cross-entropy:

$$\begin{aligned} loss &= \sum_{d \in DL} Q(y'_d, y_d) \\ &= -\sum_{d \in DL} (y_d \log y'_d + (1-y_d)\log(1-y'_d)) \end{aligned} \tag{12}$$

where DL is the set of all labeled domains, Q is the cross-entropy function, y'_d is the predicted result of domain d, and y_d is its real label.

$$\frac{\partial loss}{\partial y'_d} = -(\frac{y_d}{y'_d} - \frac{1-y_d}{1-y'_d}) \tag{13}$$

$$\begin{aligned} y'_d &= softmax\Big[\Big(H^{L} \cdot W_f + b_f\Big) W_b + b_b\Big] \\ &= softmax\Big(V \cdot W_b + b_b\Big) \end{aligned} \tag{14}$$

$$params = params - \eta * \frac{\partial loss}{\partial params} \tag{15}$$

where W_b and W_f are the trainable weight matrices for the output layer and the fully connected layer respectively, $params$ is the set of trainable parameters, η is the learning rate.

5. Evaluation

In this section, we implement a prototype of our system and evaluate it with multiple experiments from various perspectives.

5.1. Setup

To evaluate our system, we captured one week of DNS logs from a local education network in 2018. In our experiment, we only concentrated on successful A and cname records to obtain information. As described in Section 4, we collected four fields (domain, resolve IP, timestamp, and client) for A records to depict client relation and resolution relation, and two fields (domain and cname result) for cname records to obtain cname relation.

For labeling DNS data, we took ground truth from various sources: (1) Private blacklists/whitelists. The first and foremost ground truth we obtained was the blacklist and whitelist from a large Internet company. (2) Public blacklists/whitelists. We collected Alexa Top 10 K sites [30] for two months in 2018. The domain whose sTLD appeared every day in Alexa was marked as benign. Also, we collected public domain blacklists like 360 DGAs [31], malwaredomainslist.com [32], and Malc0de.com [33]. (3) Security Engine. We further leveraged VirusTotal [34], a popular security engine that has been used in many previous works as ground truth, to check our labels. Finally, we obtained over 10,000 labeled domains, including 2.6 K malicious domains and 8 K benign domains.

We implemented *AGCN-Domain* in Python 3.6 with PyTorch [35] to build the classifier and Networkx [36] to process the graph. In the following experiments, if there was no special explanation, we leveraged the 5-fold cross-validation technique to obtain final results, and the time window to capture client relation was set to one hour. Considering that it is impossible to infer the state of one domain with no relation to labeling nodes, we labeled at least one node for a component. In addition, we present the metrics used in the following experiment in Table 1.

Table 1. Confusion Matrix Calculation Table.

Metrics Name	Instruction
TP	The number of malicious domains predicted as malicious
FP	The number of benign domains predicted as malicious
TN	The number of benign domains predicted as benign
FN	The number of malicious domains predicted as benign
Accuracy	(TP + TN)/(TP + FP + TN + FN)
Precision	TP/(TP + FP)
Recall	TP/(TP + FN)
F1	2 × (Precision × Recall)/(Precision + Recall)

5.2. Features

In the following experiments, we evaluated the effectiveness of our features by comparing fused domain relation features with individual relation features.

The results are shown in Table 2, where C-R represents features extracted from only client relation graph, R-R is for resolution relation, and N-R is for cname relation. In these experiments, parameters embedding size k was set to 10, layer $L = 1$, and the initial labels fraction was 70%. The coverage ratio (CovRatio) in the table represents the domain ratio that the feature can cover. In other words, there were some domains that could not be predicted for one feature since they are isolated in the related graph. Such isolated domains were not calculated in the results for the individual feature. It can be seen that cname relation achieves the best result, but they can only cover a few domains, while our fused features can obtain good results with coverage of far more domains than the cname relation.

Table 2. Effectiveness evaluation of malicious domain classification under different relationship models and the comprehensive assessment of the three relationship models.

Relation Pattern	Accuracy	Precision	Recall	F1	CovRatio
C-R	0.9421	0.9857	0.7991	0.8827	95.04%
R-R	0.9744	0.9650	0.8932	0.9277	66.96%
N-R	0.9955	1.000	0.8333	0.9091	11.48%
Fused	0.9693	0.9814	0.8915	0.9343	100.0%

5.3. Initial Label Fraction

To further estimate the effectiveness of our system, we changed the initial label fraction of domains from 10% to 90% and observed the results, which are shown in Table 3. It is obvious that as the number of initially labeled domains increases, our system obtains better performance: the F1 score goes up from 0.87 to 0.94. It should be noted our work can detect malicious domains effectively even with limited labels: it obtains 94% accuracy and 98% precision even with only 10% initial labeled domains.

Table 3. Result with different fraction of labels.

Initial Labels	Accuracy	Precision	Recall	F1
10%	0.9427	0.9811	0.7967	0.8793
30%	0.9564	0.9770	0.8487	0.9083
50%	0.9643	0.9823	0.8731	0.9245
70%	0.9693	0.9814	0.8915	0.9343
90%	0.9720	0.9867	0.9024	0.9427

5.4. Sensitive Parameters

In this set of experiments, we analyzed the influence of parameters: embedding size k and hidden layer L.

We first set the initial label fraction to 10%, $L = 2$, and experimented with k from 10 to 200. The results are shown in Figure 5. It can be seen from the figure that the embedding size k has little effect on the result: the accuracy and F1 score are almost unchanged as k increases from 10 to 200.

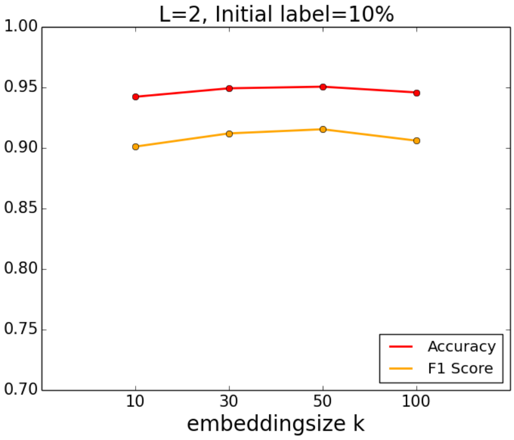


Figure 5. Impact of embedding size k.

Then we set $k = 10$, initial label fraction to 10%, and experimented with L from 1 to 7. The results are shown in Figure 6. Different from embedding size k, it can be seen that L has more impact on the result: with the increase of L, the performance slightly increases at first, but as it continually increases, the performance decreases rapidly.

The reason behind such a phenomenon is that in the beginning, nodes can obtain information from more neighbors, so they can contain deeper information on features. Once L exceeds a value, nodes obtain too much fusing information from far nodes and obtain invalid information on features.

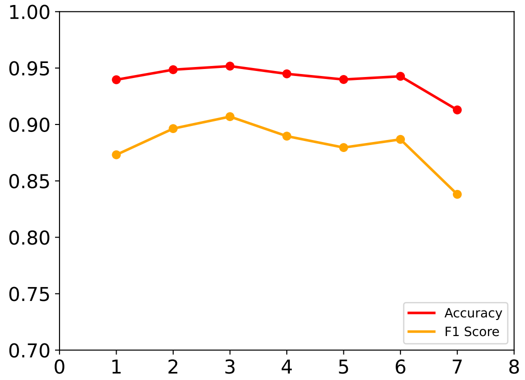


Figure 6. Impact of layer *L*.

5.4.1. Comparison with Other Models

We first compare our work with the other two models that are usually used for node classification tasks in the graph, DeepWalk and Node2Vec.

Then, we compare the *AGCN-Domain* model with the *Basic GCN* model to highlight the effectiveness of incorporating attention mechanisms in combining three relationship patterns for the detection of malicious domains.

- DeepWalk [37]. DeepWalk learns representations of graph nodes from truncated random walks. For this experiment, we took the DeepWalk model to each relation graph and added them to obtain final embedding vectors for each node. Then, we leveraged a fully connected layer to distinguish malicious domains.
- Node2Vec [38]. Node2Vec aims to learn the scalable features of nodes in the graph. Similar to DeepWalk, we took Node2Vec to each relation graph and obtained representations for each node, then added them to obtain final node features. Then, a fully connected layer was applied to predict malicious domains.
- Basic GCN [29]. GCN is a famous model and has shown great performance in many areas. In this experiment, we took a basic GCN model to each graph and combined different relation features without an attention mechanism.

Table 4 shows the result. It can be seen that our work *AGCN-Domain* can obtain better performance than other models. The attention mechanism can effectively and intelligently combine different relation features with the consideration of their various significance.

Table 4. Comparison with other models with 10% initial labels.

Model	Accuracy	Precision	Recall	F1
DeepWalk	0.9171	0.9332	0.7365	0.8233
Node2vec	0.9212	0.9323	0.7539	0.8337
BasicGCN	0.9341	0.9786	0.7640	0.8581
AGCN-Domain	0.9427	0.9811	0.7967	0.8793

5.4.2. Comparison with Other Malicious Domain Detection Systems

To further illustrate the performance of our system, we compared *AGCN-Domain* with three similar works [14,15,17] that detect malicious domains based on relations.

- Manadhata, et al. [14] constructed a client-querying-domain bipartite graph to depict *who is querying what*. Then, the researchers labeled domains with ground truth and leveraged belief propagation algorithm to predict unknown domains' states.
- Khalil, et al. [15] generated a domain resolution relation graph to represent whether two domains are sharing common resolutions. Then malicious scores for nodes can be calculated based on their distance from all known malicious domains.

- Lei, et al. [17] modeled domain behavior with three domain similarity graphs. The researchers derived domain features with graph embedding techniques from three graphs and detected malicious domains by concatenating these features.

To make the comparison, we implemented the above research locally in accordance with our understanding of their papers and experimented using the same local DNS data and labels. Table 5 shows the result. It can be seen that *AGCN-Domain* even covers more domains than other works with better performance (their limited relations make some domains untapped). It is probable that our system leverages more relations, and our model can obtain deeper and more essential features.

Table 5. Comparison with other detection systems with 10% initial labels.

Method	Accuracy	Precision	Recall	F1
Manadhata [14]	0.8994	0.7998	0.8736	0.8351
Khail [15]	0.9404	0.8869	0.8107	0.8471
Lei [17]	0.9217	0.7810	0.8676	0.8221
AGCN-Domain	0.9427	0.9811	0.7967	0.8793

6. Conclusions

In this paper, we proposed a novel system named ***AGCN-Domain***, which can distinguish malicious domains intelligently by fusing multiple domain relations considering their influences. Specifically, ***AGCN-Domain*** first analyzes domain behaviors with *client-query-domain* and *domain-resolve-IP* graphs. It describes three relations, client relation, resolution relation, and cname relation, with a relation graph of domains. Finally, a model composed by GCN and attention mechanism is applied to obtain and combine deep features of domains which are extracted from various relation graphs. We set multiple experiments from different angles to evaluate ***AGCN-Domain*** using 7 days of real-world data. We compared our ***AGCN-Domain*** model with classical graph representation algorithms, including Deepwalk, Node2vec, and BasicGCN. These algorithms simply sum up the node vector representations learned from the three relationship subgraphs without specifically focusing on the feature representation of any particular relationship subgraph or considering the combined representation of node vectors from the three relationship subgraphs. Consequently, their performance in experiments is inferior to our proposed ***AGCN-Domain*** model, demonstrating the effectiveness of our proposed attention mechanism. In comparison to the three malicious domain detection models proposed by Manadhata, Khalil, and Lei, which only consider one type of relationship between domains, the ***AGCN-Domain*** model comprehensively integrates the representations of three types of relationships through attention mechanisms, thereby highlighting the feature representations of malicious domains and improving the accuracy and F1 score of malicious domain detection. In the case of a 10% label initialization rate, the ***AGCN-Domain*** model exhibits excellent performance compared to other malicious domain detection systems, with an accuracy of 94.27% and an F1 score of 87.93%. Tables 4 and 5 show that although the ***AGCN-Domain*** model performs well in terms of accuracy and precision, **its recall rate is slightly lower than that of other malicious domain detection systems**. We speculate that this is because some malicious domain relationship patterns are similar to benign domains, requiring further extraction of deeper relationship features. In summary, **our proposed AGCN-Domain model intelligently integrates domain representation features from three relationship patterns through attention mechanisms**. Its comprehensive performance surpasses that of the compared malicious domain detection systems, balancing precision and recall, reducing the number of false malicious domain alarms, and decreasing the workload of security personnel. This demonstrates the superiority of our proposed ***AGCN-Domain*** model. In future work, we plan to propose a method that can process massive data to improve the efficiency of the detection system and explore a real-time method of detecting malicious domains.

Author Contributions: Conceptualization, supervision, X.L.; methodology, Y.L. and H.C.; formal analysis, X.L. and Y.L.; investigation, original draft preparation Y.L. and H.C.; review and editing, X.L. and H.C.; project administration, X.L. and L.Y.; funding acquisition, L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is in part supported by the National Key R&D Program of China (No. 2022YFB3104100), National Science Foundation of China (No. 62102109), and the Major Key Project of PCL (No. PCL2021A09, PCL2021A02, PCL2022A03).

Data Availability Statement: The data supporting all the findings in the article are accessible, and no additional source data are required.

Conflicts of Interest: Yixin Li was employed by Big Data Center of State Grid Corporation of China. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. Big Data Center of State Grid Corporation of China 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.

Abbreviations

The following abbreviations are used in this manuscript:

AGCN	Graph convolutional network with attention mechanism
GCN	Graph convolutional network
CR	Client relation
RR	Resolution relation
NR	Cname relation

References

1. Antonakakis, M.; Perdisci, R.; Dagon, D.; Lee, W.; Feamster, N. Building a dynamic reputation system for dns. In Proceedings of the 19th USENIX Security Symposium (USENIX Security 10), Washington, DC, USA, 11–13 August 2010; USENIX Association: Berkeley, CA, USA, 2010; p. 18.
2. Bilge, L.; Sen, S.; Balzarotti, D.; Kirda, E.; Kruegel, C. Exposure: A passive dns analysis service to detect and report malicious domains. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2014**, *16*, 1–28. [CrossRef]
3. Bilge, L.; Kirda, E.; Kruegel, C.; Balduzzi, M. Exposure: Finding malicious domains using passive dns analysis. In Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS2011), San Diego, CA, USA, 6–9 February 2011; pp. 1–17.
4. Antonakakis, M.; Perdisci, R.; Lee, W.; Vasiloglou, N.; Dagon, D. Detecting malware domains at the upper dns hierarchy. In Proceedings of the 20th USENIX Conference on Security (USENIX Security 11), San Francisco, CA, USA, 8–12 August 2011; USENIX Association: Berkeley, CA, USA, 2011; p. 27.
5. Chiba, D.; Yagi, T.; Akiyama, M.; Shibahara, T.; Yada, T.; Mori, T.; Goto, S. Domainprofiler: Discovering domain names abused in future. In Proceedings of the 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Toulouse, France, 28 June–1 July 2016; pp. 491–502.
6. Hao, S.; Kantchelian, A.; Miller, B.; Paxson, V.; Feamster, N. Predator: proactive recognition and elimination of domain abuse at time-of-registration. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1568–1579.
7. Schüppen, S.; Teubert, D.; Herrmann, P.; Meyer, U. {FANCI}: Feature-based automated nxdomain classification and intelligence. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 1165–1181.
8. Yadav, S.; Reddy, A.K.K.; Reddy, A.; Ranjan, S. Detecting algorithmically generated malicious domain names. In Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, Melbourne, Australia, 1–3 November 2010; pp. 48–61.
9. Schiavoni, S.; Maggi, F.; Cavallaro, L.; Zanero, S. Phoenix: Dga-based botnet tracking and intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 192–211.
10. Woodbridge, J.; Anderson, H.S.; Ahuja, A.; Grant, D. Predicting domain generation algorithms with long short-term memory networks. *arXiv* **2016**, arXiv:1611.00791.
11. Tran, D.; Mac, H.; Tong, V.; Tran, H.A.; Nguyen, L.G. A lstm based framework for handling multiclass imbalance in dga botnet detection. *Neurocomputing* **2018**, *275*, 2401–2413. [CrossRef]
12. Xu, C.; Shen, J.; Du, X. Detection method of domain names generated by dgas based on semantic representation and deep neural network. *Comput. Secur.* **2019**, *85*, 77–88. [CrossRef]

13. Rahbarinia, B.; Perdisci, R.; Antonakakis, M. Segugio: Efficient behavior-based tracking of malware-control domains in large isp networks. In Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Rio de Janeiro, Brazil, 22–25 June 2015; pp. 403–414.
14. Manadhata, P.K.; Yadav, S.; Rao, P.; Horne, W. Detecting malicious domains via graph inference. In *European Symposium on Research in Computer Security*; Springer International Publishing: Cham, Switzerland, 2014; pp. 1–18.
15. Khalil, I.; Yu, T.; Guan, B. Discovering malicious domains through passive dns data graph analysis. In Proceedings of the ASIA CCS '16: 11th ACM on Asia Conference on Computer and Communications Security, Xi'an, China, 30 May–3 June 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 663–674.
16. Sun, X.; Wang, Z.; Yang, J.; Liu, X. Deepdom: Malicious domain detection with scalable and heterogeneous graph convolutional networks. *Comput. Secur.* **2020**, *99*, 102057. [CrossRef]
17. Lei, K.; Fu, Q.; Ni, J.; Wang, F.; Yang, M.; Xu, K. Detecting malicious domains with behavioral modeling and graph embedding. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 601–611.
18. Sun, X.; Tong, M.; Yang, J.; Xinran, L.; Heng, L. Hindom: A robust malicious domain detection system based on heterogeneous information network with transductive classification. In Proceedings of the 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019), Beijing, China, 23–25 September 2019; USENIX Association: Berkeley, CA, USA, 2019; pp. 399–412.
19. Zou, F.; Zhang, S.; Rao, W.; Yi, P. Detecting malware based on dns graph mining. *Int. J. Distrib. Sens. Netw.* **2015**, *11*, 102687. [CrossRef]
20. Jia, Y.; Gu, Z.; Jiang, Z.; Gao, C.; Yang, J. Persistent graph stream summarization for real-time graph analytics. *World Wide Web* **2023**, *26*, 2647–2667. [CrossRef]
21. Jia, Y.; Gu, Z.; Du, L.; Long, Y.; Wang, Y.; Li, J.; Zhang, Y. Artificial intelligence enabled cyber security defense for smart cities: A novel attack detection framework based on the MDATA model. *Knowl.-Based Syst.* **2023**, *276*, 110781. [CrossRef]
22. Jia, Y.; Gu, Z.; Li, A. *MDATA: A New Knowledge Representation Model: Theory, Methods and Applications*; Springer Nature: New York, NY, USA, 2021; Volume 12647, pp. 1–255.
23. Lee, J.; Lee, H. Gmad: Graph-based malware activity detection by dns traffic analysis. *Comput. Commun.* **2014**, *49*, 33–47. [CrossRef]
24. Peng, C.; Yun, X.; Zhang, Y.; Li, S.; Xiao, J. Discovering malicious domains through alias-canonical graph. In Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS, Sydney, Australia, 1–4 August 2017; pp. 225–232.
25. Najafi, P.; Mühle, A.; Pünter, W.; Cheng, F.; Meinel, C. Malrank: A measure of maliciousness in siem-based knowledge graphs. In Proceedings of the 35th Annual Computer Security Applications Conference, San Juan, PR, USA, 9–13 December 2019; pp. 417–429.
26. Anderson, H.S.; Woodbridge, J.; Filar, B. Deepdga: Adversarially-tuned domain generation and detection. In Proceedings of the AISec '16: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, Vienna, Austria, 28 October 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 13–21.
27. Fu, Y.; Yu, L.; Hambolu, O.; Ozcelik, I.; Husain, B.; Sun, J.; Sapra, K.; Du, D.; Beasley, C.T.; Brooks, R.R. Stealthy domain generation algorithms. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1430–1443. [CrossRef]
28. Yun, X.; Huang, J.; Wang, Y.; Zang, T.; Zhou, Y.; Zhang, Y. Khaos: An adversarial neural network dga with high anti-detection ability. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 2225–2240. [CrossRef]
29. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017; Conference Track Proceedings OpenReview.net 2017.
30. Alexa. Available online: https://www.alexa.com (accessed on 15 September 2022).
31. 360DGAs. Available online: https://data.netlab.360.com/dga/ (accessed on 15 September 2022).
32. MalwareDomainList. Available online: https://www.malwaredomainlist.com (accessed on 15 September 2022).
33. Malc0de.com. Available online: https://malc0de.com/bl/ZONES (accessed on 15 September 2022).
34. VirusTotal. Available online: https://www.virustotal.com (accessed on 15 September 2022).
35. Pytorch. Available online: https://pytorch.org (accessed on 10 September 2022).
36. Networkx. Available online: https://networkx.org (accessed on 10 September 2022).
37. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.
38. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.