Ringer: Systematic Mining of Malicious Domains by Dynamic Graph Convolutional Network

Zhicheng Liu[1,2], Shuhao Li[1,2,4], Yongzheng Zhang[1,2,4], Xiaochun Yun[3], and Chengwei Peng[3]

[1] Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
[3] National Computer Network Emergency Response Technical Team/Coordination Center of China
[4] Key Laboratory of Network Assessment Technology, Chinese Academy of Sciences
{liuzhicheng,lishuhao,zhangyongzheng}@iie.ac.cn, {yunxiaochun, pengchengwei}@cert.org.cn

Abstract. Malicious domains are critical resources in network security, behind which attackers hide malware to launch the malicious attacks. Therefore, blocking malicious domains is the most effective and practical way to combat and reduce hostile activities. There are three limitations in previous methods over domain classification: (1) solely based on local domain features which tend to be not robust enough; (2) lack of a large number of ground truth for model-training to get high accuracy; (3) statically learning on graph which is not scalable. In this paper, we present Ringer, a scalable method to detect malicious domains by dynamic Graph Convolutional Network (GCN). Ringer first uses querying behaviors or domain-IP resolutions to construct domain graphs, on which the dynamic GCN is leveraged to learn the node representations that integrate both information about node features and graph structure. And then, these high-quality representations are further fed to the full-connected neural network for domain classification. Notably, instead of global statically learning, we adopt time-based hash to cut graphs to small ones and inductively learn the embedding of nodes according to selectively sampling neighbors. We construct a series of experiments on a large ISP over two days and compare it with state of the arts. The results demonstrate that Ringer achieves excellent performance with a high accuracy of 96.8% on average. Additionally, we find thousands of potential malicious domains by semi-supervised learning.

Keywords: Graph Convolutional Network · Malicious Domain Mining · Malware Activities · Deep Learning · Time-based hash

1 Introduction

Malicious domains are important platforms of launching malicious attacks in network security, such as spamming, phishing, botnet command and control (C2) infrastructure and so on. The cost of dollars is rising as a result of the growing prevalence of financial fraud based on domains. For instance, some ransomwares encrypt personal files through domain dissemination to extort money from individuals, which even cause more emotional and professional damage. Moreover, personal privacy and intellectual property rights caused by malicious domains are also arguably serious issues. Therefore, effective detection of malicious domains bears the utmost importance in fighting malwares.

Since blocking malicious domains can immediately lead to reduction in malwares, a wealth of techniques have been proposed to discover malicious domains. Those efforts can be divided into two categories, including feature-based and behavior-based methods. Traditional feature-based methods [9–11, 26] basically use the network or domain name features to distinguish

This work is supported by the National Key Research and Development Program of China (No.2016YFB0801502), and the National Natural Science Foundation of China (Grant No.U1736218). The corresponding author of this paper is Shuhao Li.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

malicious domains from benign ones due to its efficiency and effectiveness. However, attackers can circumvent the detections through some simple manipulation on these features such as making the pronounceable feature conform to the normal domains' form. Advancements in machine learning make it possible to achieve superior performance on domain classification over the passive DNS statistical features. However, it hinders adversarial crafting data generated by the Generative Adversarial Network (GAN) [8] and needs a lot of labeled data with time-consuming manual inspections.

Behavior-based methods [17, 19, 21] make use of the querying relationship of the hosts or the parsing relationship of the domains to build the association graphs in which domains or hosts represent nodes. Some inference algorithms (eg., Belief Propagation [31]) are leveraged to infer the reputation of unlabeled domains on the given labeled domains. However, they achieve poor precision when the number of ground truth is small and do nothing to the isolated nodes that have no relationship with the ground truth. Moreover, some methods [20, 25] based on analyzing host behaviors are prone to evasion techniques such as fake-querying and sub-grouping.

In order to solve the limitations of the existing methods, we propose Ringer, a novel and effective system to classify malicious domains. Our insights are built on three observations: (1) Malicious domains often serve malicious users whose behaviors deviate from that of benign domains usually registered for benign services. Hence, it is very likely that multiple malicious domains are accessed by the overlapping client set; (2) Multiple malicious domains are commonly hosted on the joint server hosts due to the fact that malicious IPs are important and scarce resources which are generally reused. For example, attackers would typically place a number of rogue softwares on the same server hosts that they control to reduce the cost. Once identified, the malwares will be migrated to another host in chunks. The reuse mechanism and common querying behaviors of these malicious resources reflect the strong correlation among malicious domains; (3) Graph convolutional network (GCN) [18] can aggregate information of both graph structure and static features to generate the underlying representation of nodes for classification tasks. In turn, we use these two strong associations, combined with advanced graph deep learning technique, to discover malicious domains.

Ringer is designed to model the association among domains into graphs, on which dynamic GCN algorithm is applied to learn the high-quality node representations that are amenable to efficient domain classification. Instead of analyzing single domain in isolation, we study the relevance of multiple domains involved in malicious activities. The robustness of the system is enhanced owing to the unchangeable behavior relationship and global correlation. Both of them will not be eliminated or altered by the will of the human being. More specifically, the detection of malicious domains by Ringer consists of three main steps. Firstly, Ringer constructs the domain association graph according to the querying behavior and resolution relation. And then, Ringer takes advantage of dynamic GCN to quickly aggregate attribute information associated with each vertex through the neighborhood defined by the graph structure in sampling fashion, thus transferring the graph-represented domains to vectors. Finally, the node representations that embed high-dimensional structural information, as well as attributes, are fed to neural network for malicious domain mining. There are two challenges that need to be addressed: (1) the existing GCN operates on the full static graphs, which cannot be directly applied to large-scale graphs (especially our internet-scale domain data); (2) it is time-consuming to learn the entire graph every time as the graph is constantly evolving. Therefore, we introduce two innovations: (1) hashing the large graphs to form small ones according to time; (2) sampling the neighborhoods of each node to construct the graph dynamically and making the training of the model dependent only on the node and its sampled neighbors, but the entire input graphs. To improve performance, we simultaneously aggregate all intermediate representations using layer-wise attention encoder.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

Our contributions are summarized as follows:

- We propose a robust malicious domain classification method named Ringer, which transforms the research problem from malicious domain mining to graph learning. We use time-based hash technology to split the graphs into small ones, which makes Ringer more scalable.
- Dynamic GCN is developed to automatically capture the structure and characteristics information of the sampled neighbors, thus generating the underlying representations for further domain classification.
- Our method has strong generalizability, which is independent of the input graph. Once the model is trained, it can be directly applied to the new domain association graphs or the newly-added nodes.
- We implement the prototype of Ringer and perform experiments on large-scale DNS traces from ISP to evaluate the effectiveness. The results show that our system has superior scalability and accuracy than state-of-the-art system, and a large number of potential malicious domains are found.

The remaining sections of this paper are organized as following. In Section 2, we review the related work. In section 3, we describe the association among malicious domains and the original GCN. We provide a systematic overview of Ringer and detail each module in Section 4. The collection of the datasets and ground truth used in this paper is elaborated in Section 5. We highlight the experimental results and discuss the limitations of our systems in Section 6, while we conclude the whole study in Section 7.

2 Related work

Malicious domain detection has been a hot issue and a great deal of strides have been made to discover malicious domains over the past years. The detection on DNS can dig out the malicious activities earlier due to the nature of DNS flow prior to attack traffic. DNS-based detection is also lighter than that based on all traffic and has no need to take into account the traffic encryption. Here, we present some representative works which can be divided into two categories: feature-based and behavior-based methods.

Feature-based methods adopt advanced machine learning combined with domain statistical features to detect malicious domains. Antonakakis *et al.* [9] put forward Notos, a system uses different characteristics of a domain to compute a reputation that indicates the domain malicious or legitimate. Bilge *et al.* [11] propose EXPOSURE, which uses fewer training datasets and less time to detect malicious domains. Moreover, the system is able to detect the malicious domains of the new categories without updating the data feeds. Babak *et al.* [25] present the Segugio, a system which tracks the co-occurrence of malicious domains by constructing a machine-domain bipartite graph to detect new malware-related domains. Antonakakis *et al.* [10] use the similarity of Non-Existent Domain responses generated by Domain Generation Algorithm (DGA) and querying behavior to identify DGA-related domains. Jehyun *et al.* [19] build domain travel graphs that represent DNS query sequences to detect infected clients and malicious domains which is robust with respect to space and time. Thomas *et al.* [28] analyze the NXDomains querying pattern on several TLD authoritative name servers to identify the strong connected groups of malware related domains.

Behavior-based approaches are to detect malicious domains by analyzing host behavior or network behavior. Pratyusa *et al.* [21] firstly use proxy logs to construct host-domain bipartite graphs on which belief propagation algorithm is used to evaluate the malicious probability of domains. Issa *et al.* [17] construct domain-resolution graph to find new malicious domains by using strong correlation between domain resolution values. Acar *et al.* [27] introduce a system AESOP, which uses a locality-sensitive hashing to measure the strength of the relationship

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

between these files to construct a graph that is used to propagate information from a tagged file to an untagged file. Liu *et al.* [20] propose CCGA which first clusters NXdomains according to the hosts infected family DGA generate the same domain name set, and then extracts the cooperative behavior relationship of domains (including time, space and character similarity) to realize the classification of NXdomain cluster. Peng *et al.* [23] design a system MalShoot, which construct domain-resolution graph to learn the relationship among domains by using graph embedding technology. They use the node embeddings as features to identify malicious domains and defend illegal activities.

3 Background

In this section, we first describe malicious domain correlations used in malicious domain mining. Then we discuss the technique of interest GCN.

3.1 Malicious domain correlation

Our method is to use the relationship among domains combined with statistical characteristics for domain classification. It is suitable for detecting multi-domain malicious activities, which complements the detection of single-domain malicious activities. The associations among malicious domains mainly fall into two categories: client similarity and resolution similarity.

Client similarity. Multiple malicious domains are accessed by a collection of overlapping clients, which is the client similarity. This similarity is determined by infected malware for the fact that the hosts infected with the same malware may have the same list of querying domains or seeds. In addition, they have the same attack strategy: get instructions from one domain, then download or update malware from another domain. Normal and malicious domains generally have no intersection on the client side. However, there are some exceptions. For example, some malwares now use fake-querying technique to query some benign domains deliberately which interfere with the detections. Fortunately, these benign domains have obvious recognizable features, such as being accessed by a large number of hosts. In that case, they can be easily removed.

Resolution similarity. A plurality of malicious domains are hosted on the same malicious IPs which constitutes the resolution similarity. Resolution sharing reveals the essential association among domains which is not changed by the will of people. This is due to the limitation of malicious IP resources and the flexibility of multi-domain malware. Malicious IPs, as pivotal resources available for attackers, are very small in quantity. While, many malwares need to use multiple domains to evade detection, improve the usability, attacking ability or survivability. Given these points, multiple malicious domains are hosted on the same set of IPs controlled or maintained by attackers. For example, once a domain is blacklisted, malwares based on DGA will generate other domains which are also resolved to the same IP.

3.2 Graph convolutional network

The convolution in Convolutional Neural Network (CNN) essentially uses a shared parameter filter (kernel) to extract spatial features by calculating the weighted sum of central points and adjacent points. GCN is a kind of deep learning technology which applies CNN technology to graph. Given an attributed un-directed graph $G = (V, E)$, where V is a set of nodes with the size of M and E is a set of edges. It is also assumed that $X \in R^{M \times N}$ is the feature matrix with each node $v \in V$ endued N-dimensional attributes, and $A \in R^{M \times M}$ is the adjacency matrix in which $A_{i,j} = a_{i,j}$ if there is an edge $e =< i, j >$ with the corresponding weight $a_{i,j}$, otherwise $A_{i,j} = 0$. Kipf *et al.* [18] give the layer-wise propagation rule of multi-layer GCN:

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

Where $\tilde{L} = D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}$ is the symmetric normalized laplacian with $\tilde{A} = A + I$, I is M-demensional identity matrix, D is the diagonal degree matrix. $W^{(l)}$ is the weight matrix of

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

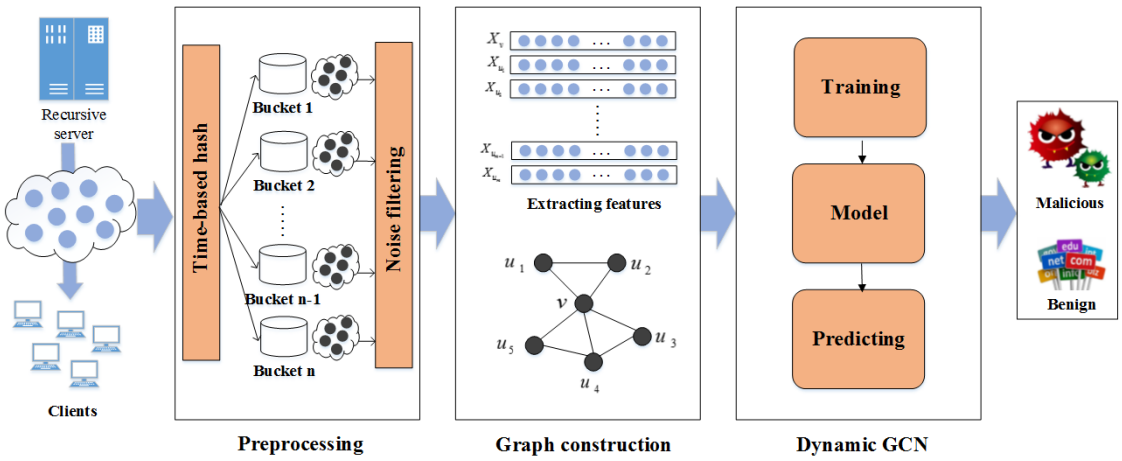


Fig. 1: The process flow diagram of malicious domain mining by using Ringer.

Table 1: The symbols used in the paper

Symbols	Meanings
TP	True positive; a malicious domain correctly identified as illicit domain
FP	False positive; a benign domain incorrectly identified as illicit domain
TN	True negative; a benign domain correctly identified as legitimate domain
FN	False negative; a malicious domain incorrectly identified as legitimate domain

layer l and $H^{(l)}$ is input activation of layer l with $H^{(0)} = X$. σ denotes activation function (eg., $ReLU(.) = max(0, .)$).

When we only take into account 2-layer GCN, the forward model becomes the following form:

$$Z = softmax\left(\tilde{L}ReLU(\tilde{L}XW^{(0)})W^{(1)}\right) \tag{2}$$

Here, $W^{(0)} \in R^{M\times H}$ is an input-to-hidden weight matrix with H feature maps in hiddn layer. $W^{(1)} \in R^{H\times F}$ is a hidden-to-output weight matrix assuming that the output has the F categories. Softmax activation function ranges each element of a vector x to $[0, 1]$ and the sum of all elements to 1 with definition $softmax(x_i) = \frac{exp(z_i)}{\sum_i exp(x_i)}$. We evaluate the cross-entropy loss function over the labeled samples as:

$$loss = -\sum_{l\in y_L}\sum_{f=1}^{F} Y_{lf} ln Z_{lf} \tag{3}$$

Where y_L is the indices of nodes that have labels in semi-supervised multi-class classification. The weight matrix $W^{(0)}$ and $W^{(1)}$ are trained by using gradient descent method to minimize the loss function (3).

GCN simultaneously learns the structure and attribute information, which have achieved great success in clustering and classification tasks [30, 32]. However, from formula (1), it can be seen that the output of each node in the current layer needs the attribute support of its neighbors from the previous layer, and with the increase of the number of layers, the number of support required by each node increases in exponential explosion. Obviously, it is only suitable for working on relatively small graphs, not for large-scale domain data such as ours. Some works [12, 15] provide a perspective on the effectiveness of sampling neighbors.

4 System design

The goal of Ringer is to discover domains involved in malicious activities by analyzing passive DNS traffic (traces). As shown in the figure 1, the system architecture of Ringer consists of three modules: preprocessing, graph construction and dynamic GCN. In order to better describe our research, we introduce some notations listed in table 1.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

4.1 Preprocessing

The preprocessing module consists of two parts: time-based hash and noise filtering.

Time-based hash. The goal of time-based hash is to divide the big graphs into small ones according to the granularity of the time. Typically, domain-based malicious behaviors are generally relatively continuous and have short duration, such as domains generated by DGA. Intuitively, if the shared similarity between two domains happens within a short period of time, then their relevance will be strong. Relatively, if a period of time is very long, it has high probablity to arise shared similarity for various reasons (such as server migration), which would introduce weak correlations. For example, the relevance intensity of two domains that have shared hosts within an hour and distinct weeks is different. Therefore, the length of the graph composition time is going to affect TPs and FPs. If the time is too short, the association between domains will not show up. If the time is too long, a large number of weak correlations will be introduced. Due to the page constraint, we will discuss the impact of time on performance in future studies. In this paper, we refer to [10] and empirically select the timespans as one hour . The time-based hash step is shown in the figure 1. Firstly, all records are hash to different time buckets according to the timestamp. Then, the noise filtering is used to remove some noises for each bucket. Finally, we construct the graphs in each hash bucket for the succedent operation.

Noise filtering. Noise filtering is designed to remove the noises introduced by some normal users as well as reduce the FPs. We adopt two strategies to filter DNS traffic. Firstly, we remove domains based on the client degree. We define the client degree of a domain as the number of clients that query the domain at a given epoch. The more clients the domain is queried by, the more popular the domain is and the less likely it is to be malicious. We remove the domains queried by more than N clients. We will discuss the selection of the threshold N later. Similarly, we can discuss the resolution degree, remove the domains with more than M resolution values. In this way, we can remove some public domains, such as content delivery networks (CDN), cloud and other domains for public services. And then, we further get rid of domains generated by some normal applications such as BitTorrent due to the fact that they have obvious characteristics, for example, ending with 'tracker' or 'DHCP'.

4.2 Graph construction

Ringer uses graph representation to express domains as the nodes and the relationship between two domains as an edge. If two domains have shared clients (resolutions) within a given period of time (in one hash bucket), then an edge exists between the corresponding nodes in the graph. The weight of the edge is the number of shared clients (resolutions) of the two domains. The graph construction algorithm is summarized as algorithm 1. In order to save space, we store the attributes of nodes and edges as files. We can get them from the file system whenever we need. As a result, the graph construction process outputs of the correlation graph of domains and the graph-related property files. The construction of the graph aggregates the DNS query information on the graph in a cumulative manner, which makes the whole system more robust and scalable.

4.3 Dynamic GCN

Dynamic GCN is the key part of our method, which takes graph structure and attributes associated nodes as input to generate high-quality embedding for each node. These embeddings are then fed to full-connected neural network for domain classification. In this section, we discuss the technical details of Dynamic GCN.

Dynamic GCN uses localized graph convolution to generate embeddings from selectively sampled neighbors. The specific forward propagation procedure is shown as Algorithm 2. More specifically, in order to generate the embedding of node v, we only need to consider the input features and the graph neighbors $N_v = \{u|(u, v) \in E\}$. We first selectionly sample

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

Algorithm 1 Algorithm to construct domain graph through passive DNS.

Input: DNS data D with each record r in the form of {*sip*, *domain*, *type*, *response*}.
Output: Domain graph G.

Initializing domain graph $G \leftarrow (V, E)$, $V \leftarrow \varnothing$ and $E \leftarrow \varnothing$, domain-clients$\leftarrow$ *dict*(), client-domains $\leftarrow$ *dict*().
for each record $r \in D$ **do**
    domain-clients[domain].update(sip)
    client-domains[sip].update(domain)
end for
for each client in client-domains **do**
    if there is only one domain d in client-domains[client] **then**
        $V = V \cup \{d\}$
        continue
    end if
    for each pair $(d_1, d_2) \in$ client-domains[client] **do**
        $V = V \cup \{d_1, d_2\}$
        if edge $e(d_1, d_2) \in E$ **then**
            weight(e) = getEdgeWeight(e) + 1
        else
            $E = E \cup \{e(d_1, d_2)\}$
            weight(e) = 1
        end if
    end for
end for
return G

Algorithm 2 Algorithm to construct domain node representation.

Input: Current embedding z_v^k of node v at layer k, set of neighbor embeddings $\{z_u | u \in N(v)\}$, preceding embeddings $(z_v^{(0)}, z_v^{(1)}, ..., z_v^{(k)})$ of node v, AGG_{node} and AGG_{layer}.
Output: New vector representations $z_v^{(k+1)}$ at layer $k+1$.

$h_v^{(k+1)} = AGG_{node}(z_v^{(k)}, \left\{z_u^{(k)}\right\}_{u \in N_v})$
$z_v^{(k+1)} = \sigma\left(W * CONCAT(z_v^{(k)}, h_v^{(k+1)})) + b\right)$
$z_v^{(k+1)} = z_v^{(k+1)} / \left\|z_v^{(k+1)}\right\|_2$
$z_v^{(k+1)} = AGG_{layer}(z_v^{(0)}, z_v^{(1)}, ..., z_v^{(k)}, z_v^{(k+1)})$

neighbors depending on the weight of the edges. Then, we introduce node-wise attention aggregator to aggregate neighbors in attention fashion. Next, we concate the aggregation from the neighbors with the current representation of node to form the intermediate representation that will be encoded into a new node embedding; The output of the algorithm is the node representation that incorporates both information itself and the neighbors, on which we use the $L2$ normalization to prevent the model from over-fitting and make the training more stable. Finally, layer-wise attention module is implemented to encode embeddings generated by different layers for enhancement of representation learning.

Selectively sample neighbors. Instead of random sampling, we sample neighbors according to the weight of the edges. We deem to that the larger the weight of the edge is, the stronger the correlation between the two points is. As a consequence, we sample the top m nodes for each node with the largest weights as neighbors. It is important to set neighbor parameters for later computational models when we take into account the fixed number of nodes. Moreover, we can control the use of memory footprint during training with the certain nodes selected to aggregate.

Node-wise attention aggregator. With neighbor context, we can use GCN to aggregate neighbor features. However, neighbors have different influence on the target node due to the different relationship strength between nodes or some noises. Therefore, we propose node-wise attention aggregator which uses a weighted approach to aggregate neighbor nodes. As shown in figure 2 (left), we use the normalized weights as the attention weights which is in

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

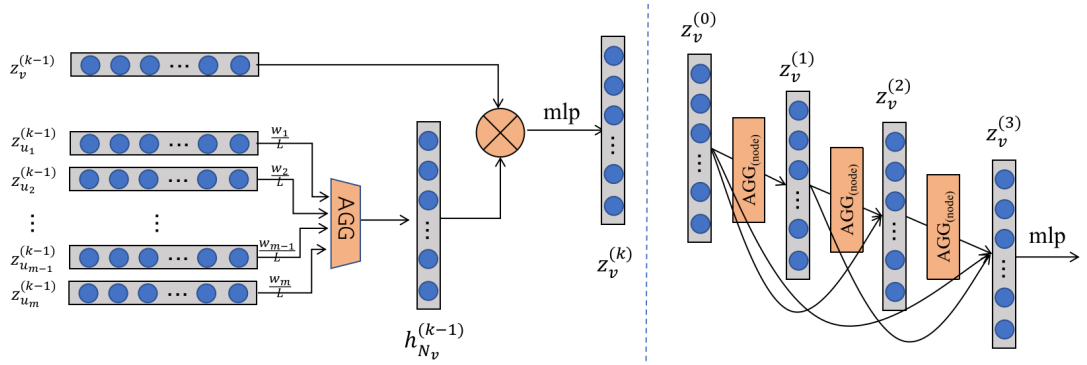


Fig. 2: Node-wise attention aggregator (left) and layer-wise attention encoder (right).

accordance with their relationship intensities. That is, for neighbors $N_v = \{u_1, u_2, ..., u_m\}$ with the corresponding weights $\{w_1, w_2, ..., w_m\}$, we can compute attention weights $\{\frac{w_1}{L}, \frac{w_2}{L}, ..., \frac{w_m}{L}\}$ and $L = |w_1| + |w_2| + ... + |w_m|$ is $L1$ normalization of weights. Compared with the original GCN, our specific node-wise attention aggregator formula is shown as follows:

$$h_v^{(k)} = AGG_{node}(z_v^{k-1}, \{z_u^{k-1}\}_{u \in N_v}) = \delta\left(W * (h_v^{k-1} + \sum_{i=1}^{m} \frac{w_i}{L} z_i^{k-1}) + b\right) \tag{4}$$

Here, k denotes the layer index k; w and b are trainable parameters of weights and bias respectively; z_v^0 is initialized by x_v.

Layer-wise attention encoder. Encouraged by residual network, many works [32] adopt skip-connection to stack convolutions for high performance. Except for this, we also propose the layer-wise attention encoder to aggregate all intermediate representations, which is similar to DenseNet [16]. Figure 2 (right) schematically illustrates the layout of the layer-wise attention encoder which introduces direct connections from any layer to all subsequent layers. The node embedding z_v^k at kth layer receives all node feature maps $z_v^0, z_v^1, ..., z_v^{k-1}$ from previous layers as input. Formally, We use the following formula with $[z_v^{(0)} : z_v^{(1)} : ... : z_v^{(k-1)}]$ referring to the concatenation operation.

$$z_v^{(k)} = AGG_{layer}(z_v^{(0)}, z_v^{(1)}, ..., z_v^{(k-1)}) = \delta\left(W * [z_v^{(0)} : z_v^{(1)} : ... : z_v^{(k-1)}] + b\right) \tag{5}$$

Training details. We train Ringer in a semi-supervised way. We define loss functions such as formula (3). Our goal is to optimize the parameters so that the output labels produced by model is close to the ground truthes in labeled dataset. we use mini-batch gradient descent to train for each iteration, which make it fit in memory.

5 Data collection

ISP dataset. To verify the effectiveness of our scheme, we collect DNS traces in an ISP recursive server within two days from November 4th to November 5th, 2017. Many steps have been taken by our data provider to eliminate privacy risks for the network users. Due to the limitation of storage space and computing resources, we uniformly sample the data with the scale of 1/10. Previous work [22] has proved that uniform sampling makes it effective to evaluate and extrapolate the key attributes of the original dataset from the samples. This billion-level data contains 1.1 billion pieces of data per hour, which contains various types of DNS records. Our experiment is based only on the A records, thus ensuring the existence of domain-to-IP resolution except Non-Existent Domains (NXdomains). After time-based hash operation, our data is cuted into 48 portions according to the time series. To better illustrate our data, we set forth the overview of dataset (shown as figure 3) in one bucket from 1:00 clock to 2:00 on November 4th, 2017, which only retains domains on A record. We can see

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

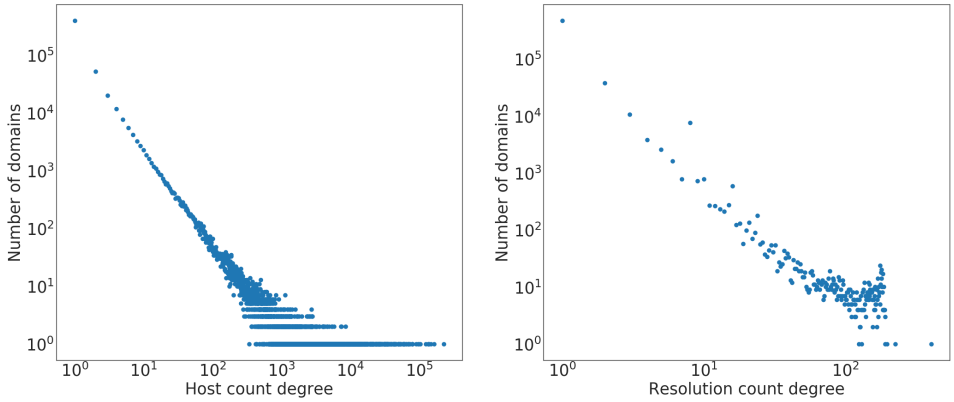


Fig. 3: Distribution of the number of domains with host count degree (left) and resolution count degree (right) in bucket 2 (within one hour from 1 o'clock to 2).

Table 2: Features extracted in this paper for learning

Type	**Feature Description**
Structural	Domain name length, Number of subdomains, Subdomain length mean, Has www prefix, Has valid TLD, Contains single-character subdomain, Is exclusive prefix repetition, Contains TLD as subdomain, Ratio of digit-exclusive subdomains, Ratio of hexadecimal-exclusive subdomains, Underscore ratio, Contains IP address
Linguistic	Contains digits, Vowel ratio, Digit ratio, Alphabet cardinality, Ratio of repeated characters, Ratio of consecutive consonants, Ratio of consecutive digits, Ratio of meaningful words
Statistical	N-gram (N=1,2,3) frequency distribution (mean, standard deviation, median, max, min, the lower quartile, the upper quartile), Entropy

that the distribution of querying host and resolution are heavy-tailed, with a few domains being accessed by a large number of hosts or resolved to a large number resolution IPs.

Ground truth. We have collected 21558 distinct second-level domains from a number of authoritative blacklists, including malwaredomains.com [2], Zeustracker [7], malwaredomainlist.com [5], malc0de.com [4], bambenek consulting [1]. We also use all DGA data from DGArchive [24] until December 31, 2018, totaling 87 DGA families or variants, 88614672 distinct domain samples. To obtain benign domains, we use the domains that appear for one year (2017) in Alexa Top 1 Million Global Sites (https://www.alexa.com/topsites) as benign domains. The owners of these sites have always maintained these sites well, so they have good reputation. Motivated by this theory, we have collected benign domains with a number of 404,536. We further drop the malicious domains out of benign domain list. Although there are some FPs and FNs in the ground truth, it is relatively effective to evaluate our method.

Features for learning. In the experiment, we extracted 42 statistical features to represent domain names with reference to FANCI [26], which are listed in table 2. These features can be divided into three categories, including structural features, linguistic features and statistical features.

6 Experiments

In this section, we discuss the selection of parameters on real-world data and evaluate the effectiveness of our approach, as well as tracking ability.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

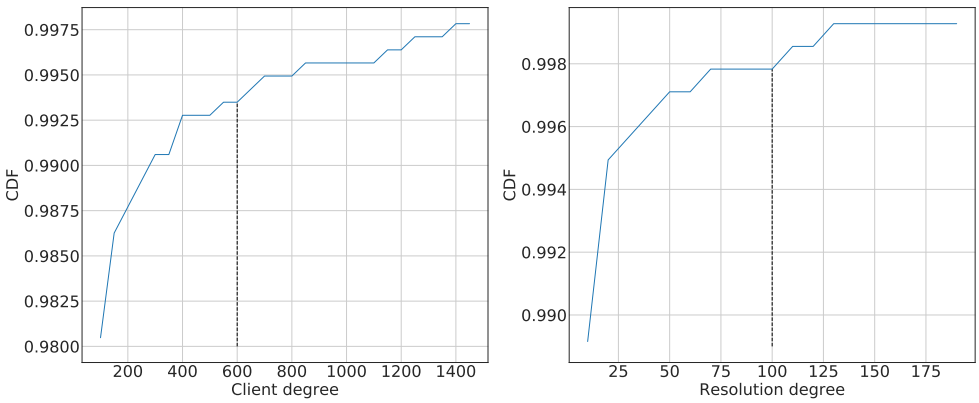


Fig. 4: The CDF distribution of the domain number over client degree (left) and resolution degree (right) in ground truth.

6.1 Selection of threshold N and M

In order to select appropriate threshold N (client degree) and M (resolution degree), we count the client and resolution degree of each domain in ground truth. Figure 4 shows the Cumulative Distribution Function (CDF) of the domain number on client degree and resolution degree in one bucket. We find that more than 99% of malicious domains have client degree less than 600. We manually check these domains with relatively large client degree and find that they are some FPs, such as the CDN or Dynamic DNS. For example, the domain "ec2-52-207-234-89.compute-1.amazonaws.com" is used for Cerber ransomware reported by malwaredomainlist.com, however, it is unreasonable for us to take the second-level domain "amazonaws.com" that is in alexa top list as a malicious domain. In order to cover more malicious domains as well as reduce the FPs, we empirically choose a threshold $N = 600$. Through this threshold, we retain 99% of the malicious domains. Similarly, we have determined that the resolution degree is $M = 100$. We manually checked the domains we removed, such as "herokuapp.com", "igexin.com" and "cloudapp.net". All of them are domains used to provide public services, which are less likely to malicious domains. Therefore, it makes sense to remove those domains.

6.2 Detection accuracy

We use three metrics to measure the performance of our approach, namely True Positive Rate (TPR), False Positive Rate (FPR) and accuracy (ACC) respectively. $TPR = \frac{|TPs|}{|TPs|+|FNs|}$, $FPR = \frac{|FPs|}{|TNs|+|FPs|}$ and $ACC = \frac{|TPs|+|TNs|}{|TPs|+|TNs|+|FPs|+|FNs|}$. For each of our buckets, there are around 3000 malicious domains and around 200,000 benign domains in ground truth. Total 136,827 malicious domains are labeled in two days (there are duplicate domains because they come from different buckets, but they have different neighbors). we randomly select the same number of benign domains (136,827) with malicious domains which are fed to Ringer to adopt K-fold cross validation (in this paper, $K = 5$). The results are shown in table 3. Our system implements high TPR (0.957 on average) and low FPR (0.020 on average) in domain classification.

Baselines for comparison. We make the comparison with the following state-of-art baselines including FANCI [26], node2vec [14] combined with features. FANCI is an open-source system that extracts 21 domain features and uses machine learning (such as Support Vector Machine (SVM)) to classify domains into benign or malicious instances. Node2vec maps the nodes into the feature vectors of the low-dimensional space by unsupervised learning, which preserves the network neighborhood relation of nodes. Node2vec is also open source and publicly available. Node2vec learns the node embeddings that only encodes the information of

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

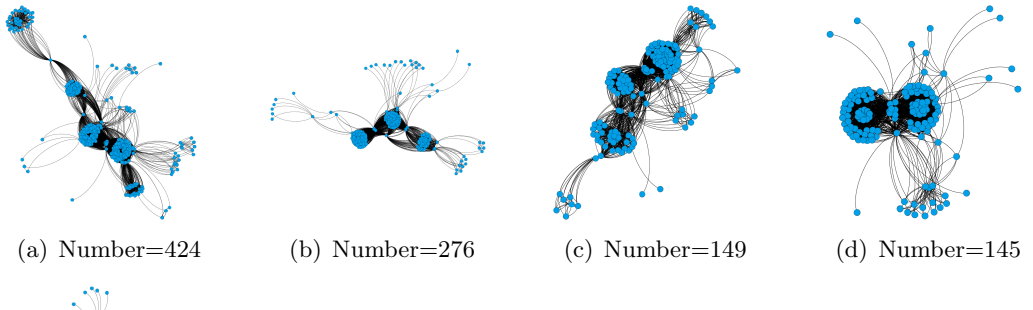
Table 3: Comparison of detection performance over Ringer, FANCI and Node2vec for domain classification using K-fold (for K=5)

Systems	Metrics	1	2	3	4	5	Average
Ringer	TPR	0.956	0.955	0.958	0.958	0.956	**0.957**
	FPR	0.018	0.020	0.024	0.020	0.020	**0.020**
	ACC	0.972	0.965	0.970	0.968	0.966	**0.968**
FANCI	TPR	0.897	0.900	0.912	0.899	0.912	0.904
	FPR	0.027	0.027	0.019	0.023	0.019	0.023
	ACC	0.939	0.936	0.937	0.938	0.947	0.939
Node2vec	TPR	0.913	0.907	0.921	0.917	0.917	0.915
	FPR	0.027	0.029	0.025	0.038	0.026	0.030
	ACC	0.941	0.938	0.947	0.939	0.944	0.942

network structure. For this reason, we first concate the node embeddings with the extracted features, and then use SVM to classify the domains.

We applied these two methods on our dataset and the results are shown in the table 3. Both of systems achieve promising results. However, FANCI ignores the structure information, and these relations embodied in time and space are of great significance to the classification of malicious domains. Node2vec only considers the structure relationship among the domains, and we concate the learned node embeddings with the static features, which has some improvement in classification performance. Yet, node2vec can only learn associated node embeddings by using unsupervised learning, and there is nothing to do with isolated nodes. To learn the node embedding of newly added nodes, all nodes need to learn in global fashion, which is time-consuming and labor-intensive. It is obvious that the performance of our system was significantly outperform the other two systems. In order to achieve optimal results, Ringer is capable of simultaneously learning statistical and structural features, and combining intermediate representations to make full use of all multi-order information of domains.

Domains detected by Ringer can be viewed as subgraphs. For example, Ringer detects 152 malicious domain subgraphs in bucket 2. In order to express the results of our method more intuitively, we select the top 8 subgraphs according to the size of connected components. The results drawed by software gephi [3] are showed as figure 5. The images vividly elaborate how



(a) Number=424 (b) Number=276 (c) Number=149 (d) Number=145

(e) Number=86 (f) Number=67 (g) Number=63 (h) Number=51

Fig. 5: The 8 samples of malicious domain subgraphs detected from bucket 2.

attackers to organize and deploy malicious domains, which is more conducive to the future study of malicious resources.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

Table 4: The results of LSTM and FANCI in prolongation

Systems	Prolongation	TPs	FNs	Percent
LSTM	6109	5192	917	0.85
FANCI	6109	5387	722	0.88

6.3 Prolongation

One significant function of Ringer is to detect potential malicious domains. As we all know, the redundancy of multi-domain names provides better flexibility for malware. Therefore, detection of newly generated (zero-day) and rarely used malicious domains is an important metric of the system. We apply the model directly on domains that are not in ground truth, and we find 6109 potentially suspicious malicious domains.

For the thousands of potential malicious domains detected by our method, we use two heuristic methods to verify their wickedness conservatively. Firstly, we use two excellent systems, FANCI [26] and LSTM [29], both of which are open source and public available. FANCI adopts machine learning on the domain features and LSTM uses deep learning technology to distinguish between good and malicious domains. The results of the classification of new suspicious domains by the two systems are as shown as table 4. Through two systems, there are still 530 unique domains that cannot be confirmed. Secondly, we try to find some historical snapshots of these domains or the IPs parsed from them by VirusTotal (www.virustotal.com). Although there are many public blacklists, they are not very comprehensive, some of which contain only malicious domains for the given day, while VirusTotal collects 66 authoritative blacklists. We deem to the domains that appear at least one blacklist as malicious domains by using the public API [6]. At a result, 108 new malicious domains we have found are exposed in the form of domains or IPs. We observed that there are still 422 domain names without qualitative judgment. Through analysis, a total of 5687 malicious domain names are found, and we have 93% confidence to believe that our system have ability to respond against new threats.

6.4 Scalability

Ringer is scalable to large-scale DNS data, such as DNS traces from ISP. To further illustrate the scalability of our system, we analyze the complexity of Ringer. Suppose that we have N records and the number of unique domain with return value on A record is V, the complexity of Ringer is shown as follows. During the graph construction, all DNS records need to be scanned which takes $O(N)$. The graph convolution is our main computational workhorse. Previously spectral convolutions defined on the graph is multiplication using a filter $g\theta(L) = diag(\theta)$ in Fourier domain with the given signal x that is a scalar for every node:

$$g\theta(L)x = g\theta(U\Lambda U^T)x = Ug\theta(\Lambda)U^T x \tag{6}$$

Where $\theta \in R^V$, U is the matrix of eigenvectors of the normalized graph Laplacian matrix. $L = I_V - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U\Lambda U^T$ with a diagonal matrix of its eigenvalues Λ. The computational complexity of formula (6) is $O(V^2)$. And for large graphs, the eigendecomposition of L is pohibitively expensive. In order to alleviate the computaional cost, we adopt K-order truncated Chebyshev polynomials as [13] approximate $g\theta(\Lambda)$: $g\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k\left(\tilde{\Lambda}\right)$. Where $\tilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_V$, and λ_{max} is the largest eigenvalue of L. The Chebyshev ploynomials is recursively defined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0 = 1$ and $T_1 = x$. Therefore, we have the approximations:

$$g\theta(L)x \approx \sum_{k=0}^{K-1} \theta_k T_k\left(\tilde{L}\right)x \tag{7}$$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

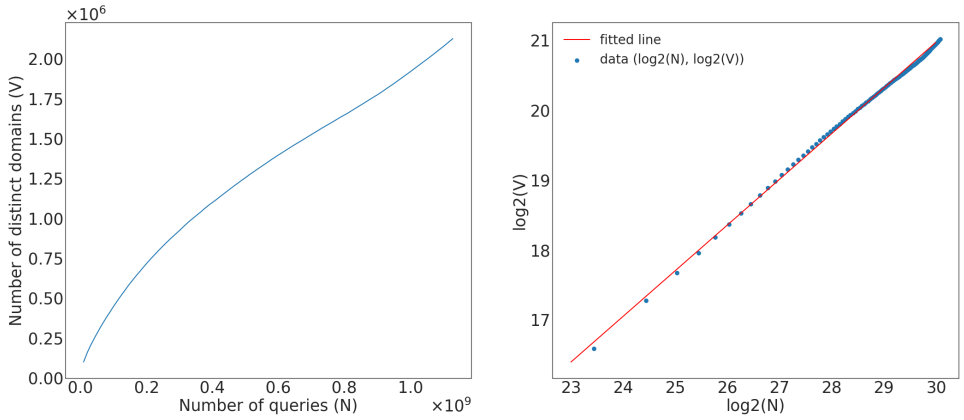


Fig. 6: The distribution of the number of distinct domains over the number of DNS queries (left) and the straight line fitted with the Least Squares approximation on points $(log2(N), log2(V))$ (right).

Where θ_k is a vector of Chebyshev coefficients and $\tilde{L} = \frac{2}{\lambda_{max}} L - I_V$ is the scaled Laplacian matrix. The complexity of formula (7) is $O(E)$, linear in the number of edges. Through the above analysis, the complexity is mainly related to the number of unique edges and the $E \leq m * V$ where the m is the number of sampled neighbors (constant).

Figure 6 (left) shows the distribution of the number of distinct domains to be analyzed changing with the number of DNS queries in real-world data. As an illustration, the size of unique domains does not increase linearly with the size of queries, but follows Heaps' law $V(N) = \alpha N^{\beta}$. In order to find the appropriate parameters α and β, we take the log2 on both sides of the equation. Then, we get $log_2(V) = log_2(\alpha) + \beta log_2(N)$. Figure 6 (right) shows the scattered distribution of points $(log2(N), log2(V))$ (blue), and the straight line (red) fitted with the Least Squares approximation. Finally, in our dataset, the parameters $\alpha = 2.577$, $\beta = 0.6536$. To sum up, the compution overhead of whole system is $O(N)$ linear in the number of input records, which proves scalable.

6.5 Limitation

We discuss about two drawbacks that need to be considered. One potential limitation is that Ringer cannot distinguish specific service categories that the malicious domains is used for, such as fishing, spamming, C2 and so on. We lack more detailed knowledge base or ground truth to cover and label them, thus obtaining the distribution of different malicious categories. Another limitation is that for unassociated or weakly associated domains, our approach is equivalent to using only its static statistical features without neighbors relevance.

7 Conclusion

The intelligence of attackers using malicious domains makes it more resilient for existing detection methods. In this paper, we propose a malicious domain detection mechanism Ringer, which uses graphs to represent the strong correlation among domains including client similarity and resolution similarity. Dynamic GCN is used to learn the node representations that combines structural information and statistical feature information inductively. The dynamic learning depending on neighbors and itself enables great gains in both effectiveness and efficiency. Exposing the relevance of malicious domains by graph enhances the robustness of the system irrespective of some evading techniques. We use DNS data from ISP to evaluate our system, the results show that our system perform higher precision and scalability. Our approach, as a promising effort, helps to prevent illegal domain-based activities more effectively in practice.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28

References

1. Bambenek Consulting. https://www.bambenekconsulting.com/ (2019)
2. DNS-BH. http://www.malwaredomains.com/ (2019)
3. Gephi: The Open Graph Viz Platform. https://gephi.org/ (2019)
4. Malc0de.com. https://malc0de.com/bl/ZONES (2019)
5. Malware Domain List. http://www.malwaredomainlist.com/ (2019)
6. VirusTotal public API. https://github.com/clairmont32/VirusTotal-Tools (2019)
7. Zeustracker. https://zeustracker.abuse.ch/blocklist.php (2019)
8. Anderson, H.S., Woodbridge, J., Filar, B.: Deepdga: Adversarially-tuned domain generation and detection. In: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security. pp. 13–21. ACM (2016)
9. Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., Feamster, N.: Building a dynamic reputation system for dns. In: USENIX security symposium. pp. 273–290 (2010)
10. Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., Dagon, D.: From throw-away traffic to bots: detecting the rise of DGA-based malware. In: Presented as part of the 21st USENIX Security Symposium (USENIX Security 12). pp. 491–506 (2012)
11. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: Finding malicious domains using passive dns analysis. In: Ndss. pp. 1–17 (2011)
12. Chen, J., Ma, T., Xiao, C.: Fastgcn: fast learning with graph convolutional networks via importance sampling. arXiv preprint arXiv:1801.10247 (2018)
13. Duvenaud, D.K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. In: Advances in neural information processing systems. pp. 2224–2232 (2015)
14. 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 (2016)
15. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems. pp. 1024–1034 (2017)
16. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
17. Khalil, I., Yu, T., Guan, B.: Discovering malicious domains through passive dns data graph analysis. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. pp. 663–674. ACM (2016)
18. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
19. Lee, J., Lee, H.: GMAD: Graph-based Malware Activity Detection by DNS traffic analysis. Computer Communications **49**, 33–47 (2014)
20. Liu, Z., Yun, X., Zhang, Y., Wang, Y.: Ccga: Clustering and capturing group activities for dga-based botnets detection. In: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 136–143. IEEE (2019)
21. Manadhata, P.K., Yadav, S., Rao, P., Horne, W.: Detecting malicious domains via graph inference. In: European Symposium on Research in Computer Security. pp. 1–18. Springer (2014)
22. Papalexakis, E.E., Dumitras, T., Chau, D.H., Prakash, B.A., Faloutsos, C.: Spatio-temporal mining of software adoption & penetration. In: 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013). pp. 878–885. IEEE (2013)
23. Peng, C., Yun, X., Zhang, Y., Li, S.: Malshoot: Shooting malicious domains through graph embedding on passive dns data. In: International Conference on Collaborative Computing: Networking, Applications and Worksharing. pp. 488–503. Springer (2018)
24. Plohmann, D., Yakdan, K., Klatt, M., Bader, J., Gerhards-Padilla, E.: A Comprehensive Measurement Study of Domain Generating Malware. In: USENIX Security Symposium. pp. 263–278 (2016)
25. Rahbarinia, B., Perdisci, R., Antonakakis, M.: Segugio: Efficient behavior-based tracking of malware-control domains in large ISP networks. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. pp. 403–414. IEEE (2015)
26. Schüppen, S., Teubert, D., Herrmann, P., Meyer, U.: FANCI: Feature-based automated nxdomain classification and intelligence. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 1165–1181 (2018)
27. Tamersoy, A., Roundy, K., Chau, D.H.: Guilt by association: large scale malware detection by mining file-relation graphs. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1524–1533. ACM (2014)
28. Thomas, M., Mohaisen, A.: Kindred domains: detecting and clustering botnet domains using dns traffic. In: Proceedings of the 23rd International Conference on World Wide Web. pp. 707–712. ACM (2014)
29. Woodbridge, J., Anderson, H.S., Ahuja, A., Grant, D.: Predicting domain generation algorithms with long short-term memory networks. arXiv preprint arXiv:1611.00791 (2016)
30. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596 (2019)
31. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. Exploring artificial intelligence in the new millennium **8**, 236–239 (2003)
32. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 974–983. ACM (2018)

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_28