2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)

Detecting Malicious Domains with Behavioral Modeling and Graph Embedding

Kai Lei[†,‡], Qiuai Fu[†,‡], Jiake Ni[†], Feiyang Wang[†], Min Yang[¶], Kuai Xu[§,*]
[†]ICNLAB, School of Electronics and Computer Engineering (SECE), Peking University, Shenzhen, China
[‡]PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China
[¶]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[§]School of Mathematical and Natural Sciences, Arizona State University
[†]leik@pkusz.edu.cn, [†]{fuqiuai, jiake.ni, wangfy16}@pku.edu.cn, [¶]min.yang@siat.ac.cn, [§]kuai.xu@asu.edu
[*]Corresponding Author

***Abstract*—The last decade has witnessed the explosive growth of malicious Internet domains which serve as the fundamental infrastructure for establishing advanced persistent threat command and control communication channels or hosting phishing Web sites. Given the big data nature of Internet traffic data and the ability of algorithmically generating domains and acquiring and registering the domains in a near-automated fashion, detecting malicious domains in real-time is a daunting task for security analysts and network operators. In this paper, we introduce bipartite graphs to capture the interactions between end hosts and domains, identify associated IP addresses of domains, and characterize time-series patterns of DNS queries for domains, and explore one-mode projections of these bipartite graphs for modeling the behavioral, IP-structural, and temporal similarities between domains. We employ graph embedding technique to automatically learn dynamic and discriminative feature representations for over 10,000 labeled domains, and develop an SVM-based classification algorithm for predicting malicious or benign domains. Our model makes the progress towards adapting to the changing and evolving strategies of malicious domains. The experimental results have shown that our proposed algorithm achieves an area under the curve (AUC) of 0.94 based on k-fold cross-validation. To the best of our knowledge, this is the first effort to apply the combination of behavioral modeling and graph embedding for effectively and accurately detecting malicious domains.**

1. Introduction

A wide spectrum of cyber attacks such as phishing, botnets, and advanced persistent attacks (APT) share one fundamental infrastructure, i.e., malicious domains for establishing control and command (C&C) communication channels or hosting fake Web sites for click frauds or credentials collection [1]. Although there exist a rich body of research efforts in the literature, detecting malicious domains remains a daunting and challenging task due to the big data nature of Internet traffic and the ability of algorithmically generating domains and acquiring them in a near-automated fashion [2], [3].

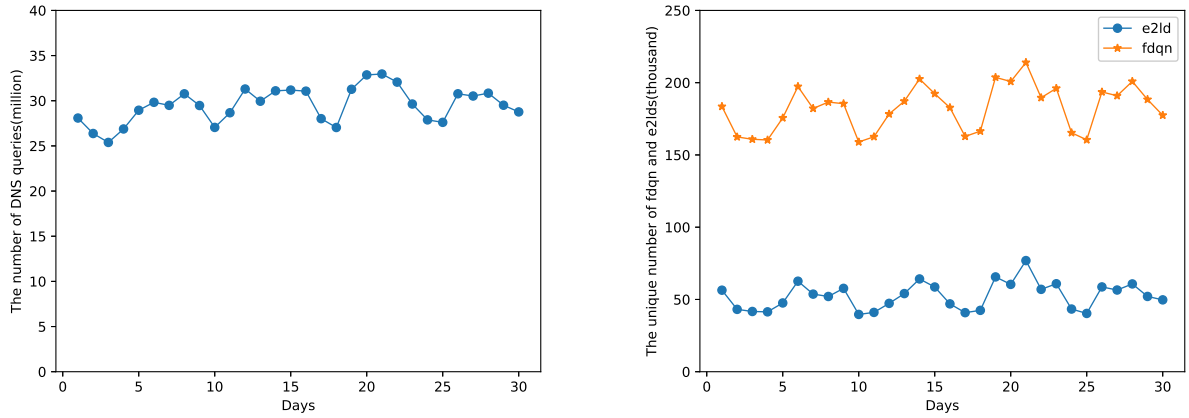
In this paper, we explore bipartite graphs to model the interactions of host-domain query behaviors, domain-IP resolving structure, and the temporal patterns of domains, and build one-mode projections of three different bipartite graphs for capturing the similarity of domains on end-host querying, IP resolving, and temporal patterns. Comparing with features manually extracted from domain knowledge, network traffic or lexical expressions, these behavioral features are more robust and stable since the underlying behavior of malicious domains tend to exhibit high consistency over. Prior studies have shown that domain knowledge varies across networks, network features such as TTL values set by malicious domains shift over time for avoiding detection [4], and cyber attacks can easily emulate lexical features of benign domains, e.g., using similar number of characters or pronounceable words in the domain names.

The performance of machine learning based malicious domain detection methods is heavily dependent on the selection of feature representation in feature engineering, which is important but labor-intensive. However, the major weakness of conventional malicious domain detection methods based on such methods lies in their inability to extract and organize the discriminative and dynamic features from the data.

In this paper, we employ LINE [5], which is one of the best performers in graph embedding, to automatically learn the dynamic and discriminative latent feature representations from the constructed domain graph. This innovative process of extracting the dynamic features and ease of applicability of machine learning makes the malicious domain detection methods less dependent on feature engineering, and more importantly, improves the flexibility of these methods against the changing and evolving attacking strategies.

The feature vectors generated from graph embedding represent the domain behavior in a high dimension space, thus we develop an SVM-based binary classifier with over 10,000 labeled domains for predicting the new domains as malicious or benign. Our extensive experimental results have shown that the proposed algorithm achieves an AUC of 0.94 based on k-fold cross-validation, which outperforms the strong competitors by an improvement of 6.8% on the AUC metric.

2575-8411/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDCS.2019.00066



(a) DNS query volume

(b) Unique numbers of FQDNs and e2LDs

Figure 1: The DNS query volumes and the unique numbers of observed FQDNs and e2LDs over time in the large campus network over 1-month time period.

The contributions of this paper are three-fold:

- This paper introduces bipartite graphs and one-mode projections for modeling domains behaviors on host interactions, resolving IP addresses, and temporal patterns.
- This paper explores graph embedding for learning feature representations based on the similarities among domains, which makes our model be independent of the labor-intensive feature engineering and captures the evolving attacking patterns.
- The extensive experimental results have shown that our proposed algorithm achieves significant malicious domain classification and clustering results based on DNS traffic logs collected from a large campus network.

The remainder of this paper is organized as follows. Section 2 describes the background of the research problem and the data-sets used in this paper. Section 3 presents an overview of our method. In Section 4, we explore bipartite graphs and one-mode projections to model the domain behaviors on the interactions with end hosts, resolved IP addresses and temporal patterns, while Section 5 introduces graph embedding to build feature vectors based on the similarity matrix of the domains. Section 6 develops an SVM-based classification algorithm with labeled training data-sets for predicting new domains as malicious or benign, while Section 7 identifies strong associations among malicious domains. In Section 8, we evaluate the performance of our proposed algorithm. Section 9 discusses related work in detecting malicious domains, and Section 10 concludes this paper and outlines our future work.

2. Background and Data Sets

The last decade has witnessed the wild growth of malicious domains for hosting malware, phishing and scam content and for facilitating command and control (C&C) communication channels for botnets. As a key Internet infrastructure for resolving Internet hostnames to IPv4 or IPv6 addresses and vice versa, DNS is triggered and observed for all Internet applications which involve hosts and domains. Detecting malicious domains in DNS traffic originating from end hosts in real-time is a crucial step for preventing these vulnerable hosts from being compromised by a wide spectrum of cyber attacks.

On the other hand, cyber attackers have devised intelligent mechanisms such as DNS based domain fluxing [6] to algorithmically generate and register a large number of short-lived domain names for operating C&C servers and hosting malicious sites. In addition, cyber attackers often rapidly change the IP addresses of the C&C servers or malicious sites under their control with DNS-based FastFlux tools [7] to avoid the simple IP blocking. Thus there is a pressing need to develop effective and accurate algorithms to characterize, detect and filter malicious domains during the very early stage of their operations.

Towards this end, we first collect IP data packets of DNS traffic originating from or destined to all the campus DNS servers via the edge routers in the campus network which supports thousands of desktops, laptops, servers, smart phones, tablets, and Internet-of-things[1] since June 2017. For each DNS query packet, we collect timestamp, identification number, source IP address, queried name, and query type, e.g., A, NS, CNAME, or MX. For each DNS reply packet, we extract timestamp, identification number, destination IP address, and the response value for the corresponding type. In addition to the DNS logs, we also collect DHCP logs in parallel for mapping the MAC addresses to the assigned IP addresses. The DHCP data ensures our ability of pinpointing to the same compromised physical

1. For simplicity, we refer to these Internet-capable devices as end hosts throughout this paper.

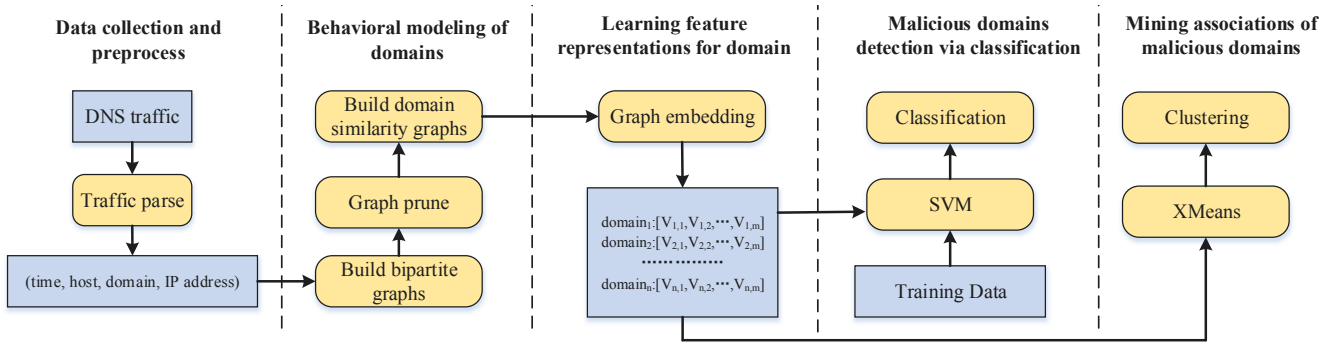


Figure 2: An overview of our proposed end-to-end system for detecting malicious domains from a large campus network.

devices in the campus network which could have different IP addresses due to device mobility on the same campus or DHCP lease timeout.

Figure 1(a)(b) illustrate the DNS query volumes and the unique numbers of fully qualified domain names (FQDNs) and effective second level domains (e2LDs) over time in the large campus network during March 2018. As seen in Figure 1, the big data nature of DNS traffic creates unprecedented challenges for characterizing and detecting malicious domains on the fly.

3. System Architecture

In this paper, we develop an end-to-end prototype system to detect malicious domains from a large campus network. Figure 2 illustrates a schematic overview of the system architecture which consists of five key system components: data collection and pre-processing, behavioral modeling of domains, learning domain features via graph embedding, supervised learning algorithms for classifying malicious or benign domains, and unsupervised learning algorithms for clustering malicious domains.

The first system component is to automatically collect DNS traffic logs from edge routers in the campus networks, and pre-process the logs to extract the records of DNS queries and the corresponding responses. The second system component is centered on the behavioral modeling of domains via bipartite graphs and one-mode projections for capturing the behaviors of domains on end-host querying, IP addressing resolving and temporal patterns, while the third component explores graph embedding techniques for learning feature representations of domains. The fourth component develops an SVM-based classification algorithm with over 10,000 labeled domain data-set for predicting new domains observed in the same campus networks as malicious or benign. The last component uses a clustering algorithm to discover malware families and other applications.

4. Behavioral Modeling of Domains via DNS Traffic

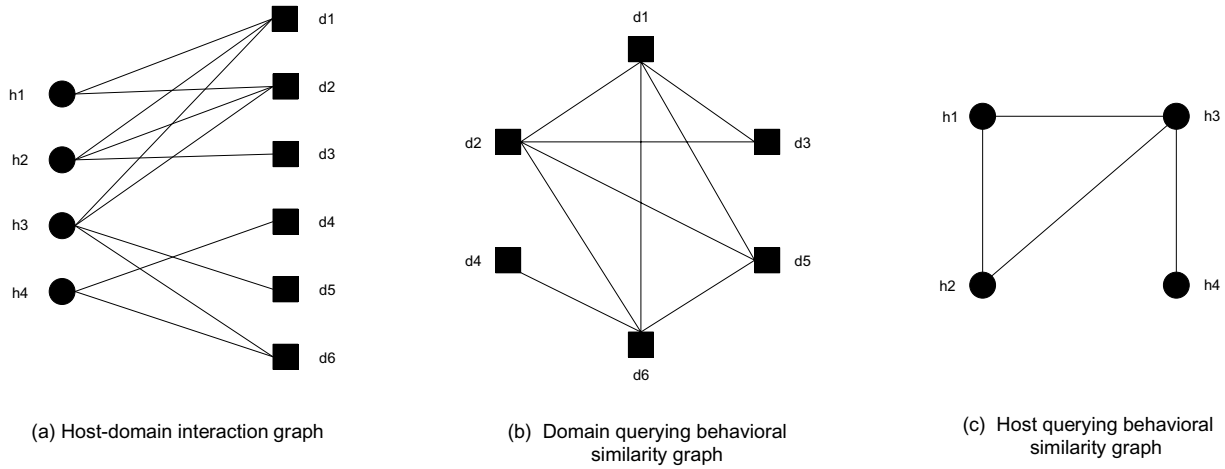
In this section, we first introduce bipartite graphs for modeling the DNS behaviors of benign and malicious domains. Specifically, we explore three types of bipartite graphs for capturing the interactions between end hosts on the campus networks and domains, identifying associated IP addresses of domains, and characterizing time-series patterns of DNS queries for domains, respectively. Subsequently, we rely on one-mode projections of these bipartite graphs for understanding the behavioral, IP-structural, and temporal similarities of domains.

4.1. Modeling Domain Behaviors with Bipartite Graphs

4.1.1. Host-Domain Bipartite Graph. The first type of bipartite graphs is host-domain interaction graph $HDBG = (\mathcal{H}, \mathcal{D}, \mathcal{E})$, where the vertex sets $\mathcal{H}$ and $\mathcal{D}$ represent all end hosts in the campus network and all the Internet domains queried by these hosts during a given time window t. The edge set $\mathcal{E}$ denotes all interactions between hosts in $\mathcal{H}$ and domains in $\mathcal{D}$. For example, if a host $h_i \in \mathcal{H}$ sends a DNS query for the domain $d_j \in \mathcal{D}$, an edge $e_{h_i,d_j} \in \mathcal{E}$ will be created between h_i and d_j. As shown in Figure 3(a), the host-domain bipartite graph essentially captures the query behaviors of domains as well as the interactions between end hosts in the campus networks and these domains.

4.1.2. Domain-IP Bipartite Graph. The second type of bipartite graphs is domain-IP mapping graph $DIBG = (\mathcal{D}, \mathcal{I}, \mathcal{E})$, where the vertex set $\mathcal{D}$ consists of all the domains observed in DNS traffic logs, while the vertex set $\mathcal{I}$ represents all the IP addresses resolved for the hostnames from the domains in $\mathcal{D}$. A hostname from domain $d_i \in \mathcal{D}$ resolving to an IP address $i_j \in \mathcal{I}$ will lead to an edge $e_{d_i,i_j} \in \mathcal{E}$ between d_i and i_j. Thus the domain-IP bipartite graph effectively illustrate the resolved IP addresses for all the hostnames in the same domains.

4.1.3. Domain-Time Bipartite Graph. The third type of bipartite graphs for modeling the DNS behaviors of benign and malicious domain is domain-time association graph $DTBG = (\mathcal{D}, \mathcal{T}, \mathcal{E})$. The vertex sets $\mathcal{D}$ and $\mathcal{T}$ represent all the domains and the distinctive time units. If a given domain $d_i \in \mathcal{D}$ is observed in at least one DNS query



(a) Host-domain interaction graph

(b) Domain querying behavioral similarity graph

(c) Host querying behavioral similarity graph

Figure 3: Discovering domain behavioral similarities via bipartite graphs and one-mode projection graph.

queried during the time window $t_j \in \mathcal{T}$, there will exist an edge $e_{d_i,t_j} \in \mathcal{E}$ between d_i and t_j. In this paper, the duration of the time window t_j is one minute. Therefore the domain-time bipartite graph represents the temporal behavior of domains which are being queried by end hosts in the campus network.

These three types of bipartite graphs carry a wide spectrum of behavioral features for domains which interact with end hosts on the campus network. In this study, we construct each graph based on DNS traffic logs for a month.

As we collect DNS traffic logs from a large campus network, the constructed bipartite graphs contain a sheer volume of domains, resolved IP addresses, and end hosts on the campus networks. For improving the running time of our proposed algorithms, we prune each bipartite graph without sacrificing the coverage and accuracy of malicious domains detections with the three simple rules. Firstly, we prune the well-known domains, e.g., google.com, which interact with over 50% of end hosts in the campus networks for providing valuable and legitimate Internet services for the students and faculty. Secondly, we remove all domains which are requested only by a single host. Although this step has a potential risk of pruning malicious domains during its early stage. However, our empirical experience has shown that our proposed algorithm is able to identify such domains in a later stage with the accumulated knowledge on their behavioral, temporal, and IP-structural patterns. Thirdly, we use the only effective second-level domain (e2LDs) in the graphs since e2LDs often tell the domain ownerships and reflect the actual organizations behind the domains. For example, the e2LDs for the hostnames maps.google.com and www.bbc.uk.co are google.com and bbc.uk.co, respectively. Moreover, e2LDs are the widely used domain aggregation unit for detecting malicious domains in the literature.

4.2. Discovering Domain Behavioral Similarities via One-Mode Projection Graphs

Projecting bipartite graphs on each of two vertex sets leads to two one-mode projection graphs, which are widely used for capturing the similarities among nodes in the same vertex set [8], [9]. Section 4.1 models domain behaviors with three distinct types of bipartite graphs, thus we could naturally build one-mode projection graphs on the domain vertex set for capturing the querying behavioral similarity, IP resolving similarity, and temporal behavioral similarity. For example, Figure 3(b) builds the one-mode projection graph for the domain vertex set from the host-domain bipartite graph. Similarly, projecting the bipartite graph on the host vertex leads to another one-mode projection graph, as shown in Figure 3(c), which essentially captures the shared domain interests for different end hosts.

4.2.1. Domain Querying Behavioral Similarity Graph. Our intuition of exploring domain querying behavioral similarity lies in the principle that if two domains are queried by the same set hosts on the campus networks during a certain time window, these two domains are highly correlated and strongly associated. Previous studies have revealed that compromised hosts infected with the same malware family tend to query a similar set of malware-control domains [10]. In general, normal hosts are unlikely to query domains that are created for proving malware-only functionalities such as serving as command and control centers.

To construct the domain querying behavioral similarity graph, we first define how to measure the similarity between two domains observed in DNS traffic logs. Let d_i represent a random domain in $\mathcal{D}$. The set of end hosts $\mathcal{H}_{d_i}$ denotes all hosts that queries the IP addresses of at least one hostname in the domain d_i. Similarly, the set $\mathcal{H}_{d_j}$ denotes all hosts the queries the hostnames in the domain d_j. The Jaccard index, s_{d_i,d_j}, for measuring the query behavioral similarity between the domains d_i and d_j is then defined as follows:

$$qs_{d_i,d_j} = \frac{|\mathcal{H}_{d_i} \cap \mathcal{H}_{d_j}|}{|\mathcal{H}_{d_i} \cup \mathcal{H}_{d_j}|}, \quad (1)$$

where $|\mathcal{H}_{d_i} \cap \mathcal{H}_{d_j}|$ represents the number of the hosts that query both domains d_i and d_j while $|\mathcal{H}_{d_i} \cup \mathcal{H}_{d_j}|$ represents the number of the hosts which query either the domain d_i or the domain d_j. A similarity qs_{d_i,d_j} of 1 suggests that d_i and d_j are queried by the same set of end hosts in the campus network. Thus, the higher the similarity qs_{d_i,d_j}, the stronger query correlations between the domains d_i and d_j. Based on this similarity measure capturing the domain query behavior correlations, we could construct the full similarity graphs for all domains observed in DNS traffic logs at a given time window.

4.2.2. Domain IP Resolving Similarity Graph. Domains controlled by the same organizations often share IP blocks or addresses via shared Web hosting or other techniques, thus it is not surprising to observe different domains share IP addresses [11]. Therefore if the hostnames from two different malicious domains are resolved to the same IP addresses, these two domains are likely associated with the same cyber criminals. Intuitively, the more IP addresses shared by two domains, the more stronger association between them.

Let $\mathcal{I}_{d_i}$ and $\mathcal{I}_{d_j}$ denote the set of IP addresses resolved from the hostnames of domains d_i and d_j respectively. Then based on the Jaccard similarity coefficient we define the domain IP resolving similarity is_{d_i,d_j} between domains d_i and d_j as:

$$is_{d_i,d_j} = \frac{|\mathcal{I}_{d_i} \cap \mathcal{I}_{d_j}|}{|\mathcal{I}_{d_i} \cup \mathcal{I}_{d_j}|}. \quad (2)$$

Similar to qs_{d_i,d_j}, the higher IP resolving similarity is_{d_i,d_j} between two domains, the stronger their association.

4.2.3. Domain Temporal Similarity Graph. Our empirical analysis on DNS traffic logs shows the strong temporal correlations between many domains. The major reasons behind such correlation are client-side or server-side Web redirections and embedded hyper-links in HTML content. For example, when an Internet browser requests a Web page, the local host first sends a DNS request to resolve IP addresses of the Web site[2] for TCP connections. Once the browser starts to render the Web page, it will generate additional DNS queries for the domain names which are embedded in the page, e.g., Google's sponsored links on CNN front page. For client-side or server-side redirection, the browser will have to resolve the hostname for each URL in the redirection chain before the Web page is retrieved from the last URL. As shown in [12], drive-by exploits typically involve a long redirection chain before the actual exploit activity occurs.

Let $\mathcal{T}_{d_i}$ and $\mathcal{T}_{d_j}$ denote the minute sets we observe domains d_i and d_j respectively in DNS traffic logs. Then the temporal similarity of domains d_i and d_j is defined as

$$ts_{d_i,d_j} = \frac{|\mathcal{T}_{d_i} \cap \mathcal{T}_{d_j}|}{|\mathcal{T}_{d_i} \cup \mathcal{T}_{d_j}|}. \quad (3)$$

Clearly the temporal correlation between two domains measures how frequently these domains are queried by end hosts on the campus networks at the same time.

5. Learning Feature Representations for Domains

The performance of machine learning based malicious domain detection methods is heavily dependent on the selection of data representation (or features) on which they are applied. Such feature engineering is important but labor-intensive and highlights the weakness of the conventional malicious domain detection methods: their inability to extract and organize the discriminative and dynamic features from the data. In order to extract the dynamic features and ease of applicability of machine learning, it would be highly desirable to make the malicious domain detection methods less dependent on feature engineering, so that novel applications could be constructed faster, and more importantly, to make progress towards adapting to the changing and evolving attacking strategies. In this paper, we employ LINE [5], which is one of the best performers in graph embedding, to automatically learn the useful and meaningful (latent) feature representations from the constructed domain graph. In the rest of this section, we elaborate on the LINE algorithm in detail.

5.1. Problem Definition

LINE [5] learns the low-dimensional embeddings of large-scale information networks. It is designed for homogeneous graphs, i.e., the networks with the same types of nodes. The main idea behind LINE is to make use of the first-order proximity and the second-order proximity between vertices, which assumes vertices with similar neighbors are similar to each other and thus should be represented closely in a low dimensional space. In particular, LINE preserves the local structures that are represented by the observed links in the networks, capturing the first-order proximity between the vertices. In addition, LINE explores the second-order proximity between the vertices, which is not determined through the observed tie strength but through the shared neighborhood structures of the vertices.

We formally define the problem of homogeneous graph embedding. In this study, we have three kinds of domain similarity graphs, and each graph is defined as $G = (D, E, W)$, where D is the set of vertices, each of which represents a domain; E is the set of edges between the domains, each of which represents a relationship between two domains; W is the set of weights, each of which represents the similarity between two domains. Each edge $e_{ij} \in E$ is an unordered domain pair $e_{ij} = (d_i, d_j)$ and is associated with a weight $w_{ij} > 0$. Note that our domain similarity graphs are undirected, thus we have $(d_i, d_j) \equiv (d_j, d_i)$.

2. Here we assume the IP address of the Web site is not cached on the host or expired.

First-order Proximity. Following [5], the local network structures of the similarity graphs can be defined as the *first-order proximity* between the vertices. For each pair of domain (d_i, d_j) linked by an edge e_{ij}, the weight on that edge (i.e. w_{ij}) indicates the first-order proximity between domain d_i and domain d_j. If no edge is observed between d_i and d_j, their first-order proximity is 0. Because of the importance of the first-order proximity in practice that implies the similarity of two nodes, the majority of graph embedding algorithms aims to preserve the first-order proximity.

Second-order Proximity. The vertices that share similar neighbors are likely to be similar to each other, even though there is no link between them. LINE explores second-order proximity which complements the first-order proximity and preserves the graph structure. Formally, assume that $s_i = (w_{i,1}, ..., w_{i,|V|})$ denote the first-order proximity of d_i with all the other vertices, then the *second-order proximity* between d_i and d_j is determined by the similarity between s_i and s_j. If no vertex is linked from/to both d_i and d_j, the second-order proximity between d_i and d_j is 0.

Given the domain similarity graph $G = (D, E, W)$, LINE aims to learn low-dimensional vertex representations, preserving both the first-order proximity and the second-order proximity between the vertices.

5.2. Graph Embedding Algorithm

The training objective of LINE is to find vertex representations that maximize the occurrence probability among the directly connected vertices (the first-order proximity) and the vertices sharing many connections to other vertices (the second-order proximity). Given the domain similarity graph $G = (D, E, W)$, we first define the conditional probability of vertex $d_i \in D$ generated by $d_j \in D$ as:

$$p(d_i|d_j) = \frac{\exp(\overrightarrow{u}_i^T \cdot \overrightarrow{u}_j)}{\sum_{i' \in |V|} \exp(\overrightarrow{u}_{i'}^T \cdot \overrightarrow{u}_{i'}^T)} \tag{4}$$

where $\overrightarrow{u}_i$ is the embedding vector of vertex d_i, and $\overrightarrow{u}_j$ is the embedding vector of vertex d_j, $|V|$ is the number of vertices or "contexts". For each vertex $d_j \in D$, Eq. (4) defines the conditional distribution $p(\cdot|d_j)$ over all the vertices in the domain set. For each pair of vertices d_j and $d_{j'}$, the second-order proximity can be calculated by their conditional distributions $p(\cdot|v_j), p(\cdot|v_{j'})$. Thus, we can make the conditional distribution $p(\cdot|v_j)$ close to its empirical distribution $\hat{p}(\cdot|v_j)$ by minimizing the following objective function:

$$O = \sum_{d_j \in D} \sum_{d_i \in D} w_{ij} KL(p(\cdot|d_j), \hat{p}(\cdot|d_j)) \tag{5}$$

$$= - \sum_{e_{ij} \in E} w_{ij} \log p(d_j|d_i) \tag{6}$$

where $KL(\cdot, \cdot)$ is the KL-divergence between two distributions; $\sum_{d_i \in D} w_{ij}$ indicates the importance of vertex d_j.

Finally, the objective function Eq. (6) can be optimized with stochastic gradient descent through the techniques of edge sampling [5] and negative sampling [13]. Concretely, in each step, a binary edge e_{ij} is sampled with the probability proportional to its weight w_{ij}, and meanwhile multiple negative edges are sampled from a noise distribution. The reader can refer to [5] for the implementation details of LINE.

6. Malicious Domains Detection via SVM Classification

In this section we will first discuss how we obtain labeled data-sets on the benign or malicious domains for supervised learning, and subsequently introduce an SVM-based classification algorithms for effectively and accurately detecting malicious domains.

6.1. Labeled Data-Sets for Supervised Learning

We obtain a blacklist of malicious domains and a whitelist of benign domains from a large Internet security company. In order to reduce false positives of malicious domains, we validate each malicious e2lds domain with the public VirusTotal API [14] which queries over 60 global blacklists from different credible sources. We will only include an e2lds domain as a labeled malicious domain if and only if the domain is confirmed by the VirusTotal API, and appears at least two of the 60 global blacklists. The final labeled data-set consists over 10,000 domains. Among these domains, 30% are confirmed malicious domains while the remaining 70% are benign domains.

For each domain in the labeled data-set, we could build three feature vectors via learning domain querying behavioral similarity graph, domain IP resolving similarity graph, and domain temporal similarity graph, respectively. Given an embedding size of k, we could use $[V_1, V_2, ..., V_k]$, $[V_{k+1}, V_{k+2}, ..., V_{2k}]$, and $[V_{2k+1}, V_{2k+2}, ..., V_{3k}]$ to represent these three feature vectors. Thus the final feature vector is in the form of $x \in R^{3k}$, where x_j denotes its jth component. The experimental results show that combining three feature vector together achieves the best detection quality. In Section 8.1, we will study the marginal contribution of each feature vector and discover the unique contribution of each vector. In addition, we use $y \in 0, 1$ to denote the class label of a domain with $y = 1$ indicating a malicious domains and $y = 0$ indicating a benign domain.

6.2. SVM Classification Algorithms

In this study, we choose SVM classification algorithm for building the binary classifier for predicting malicious or benign domains due to its wide popularity and efficiency on high-dimensional data-sets [15]. As the SVM classifier is built for maximizing the margin of correct classification, the established decision boundary is robust or insensitive to slight changes on the feature vectors, thus minimizing the over-fitting effect.

The decision rule generated by SVM classifier is expressed via a kernel function, i.e., $K(x, x')$ which computes

the similarity between two feature vectors and non-negative coefficients $\{a_i\}_{i=1}^{n}$ where n is the size of training dataset. In this article, we exploit radial basis function(rbf) to be the kernel function. The penalty parameter is set to 0.09 and the kernel coefficient for rbf is set to 0.06. The SVM classifier predicts the class label of the new domains, e.g., x, via computing their distance $d(x)$to the decision boundary as follows:

$$d(x) = \sum_{i=1}^{n} a_i(2y_i - 1)K(x_i, x), \tag{7}$$

where the sum is aggregated over all the objects in the training set. The sign of the derived distance measure determines the side of the decision boundary on which the new domain x lies. In practice, we could set a threshold value for $d(x)$ to predict the binary class label, i.e., malicious or benign, for the new domains.

7. Mining Associations of Malicious Domains

In this section, we first discuss how to uncover the strong association among malicious or benign domains via cluster analysis, and subsequently visualize the feature vectors of domains.

7.1. Clustering Analysis of Domain Clusters

Malicious or benign domains are often associated as they are from the same malware family or the same business owners. The ability of our proposed SVM classifier in distinguishing malicious or benign domains allow us to further group associated domains into the same clusters for in-depth analysis.

In this paper, we apply a simple yet effective partitioning-based clustering algorithm, i.e., XMeans [16] due to its simplicity and automated selection and optimization on the number of clusters. The distance measure between any two domains is measured by the Euclidean distance on their feature vectors.

The discovery of domain clusters shows very interesting and strong association among malicious or benign domains. For example, most of 61 domains in one cluster are reported as spam or phishing domains by ThreatBook [17]. Table 1 shows a subset of these spam domains in this cluster. Similarly, one cluster includes 131 domains, and most of them are reported as Conficker DGA domains by ThreatBook, and unsurprisingly these domains share the same IP addresses and are queried by the same of end hosts on the campus networks. Table 2 shows a subset of these DGA-generated domains in the cluster.

brvegnholster.bid	turputch.bid
fattylivercur.bid	curafane.bid
bstwoodprofit.bid	ankletol.bid
concldpermitt.bid	nanoclen.bid
undetectdrger.bid	cooknice.bid
turmericuses.bid	bellydit.bid
lrn2plymuzic.bid	easycanvvas.bid
simflightgam.bid	solaramrica.bid

TABLE 1: An example of spam domains cluster

oorfapjflmp.ws	nalenyjvncb.ws	oceudjhxmts.ws
lhkiqoedwng.ws	kkdgqwjedyj.ws	qxenrcbugvr.ws
gqzmihbsjrs.ws	nhqaldggstz.ws	tylyqdbtzud.ws
vrstvbcvkpm.ws	hcamlehhrpc.ws	emkpjrvaxxq.ws
dyyrtlmnsyf.ws	nseotiosjvj.ws	sakssuuaxex.ws
zumxvivnqma.ws	urfdekqrffe.ws	nomgyfygkub.ws

TABLE 2: An example of DGA-generated domains cluster

7.2. Applications

7.2.1. Acquiring Additional Labeled Malicious Domains for Model Training. The discovery of malicious or benign domain clusters can reciprocally improve malicious domain detection via SVM classifications by acquiring additional labeled domains for model training. In other words, we could potentially identify a large number of suspicious domains from a few known and confirmed malicious domains.

For each unknown domain in the malicious domain clusters from our experiments, we query the public VirusTotal API [14] to confirm if it is indeed a known malicious domain. We consider the confirmed domains as *true* malicious domains and the unconfirmed ones as *suspicious* domains. Figure 4 shows the number of newly discovered true and suspicious malicious domains with varying seed sizes of malicious domains. As we increase the seed size of malicious domains from 0 to 200, we can discover nearly 2,000 true malicious domains and 500 suspicious domains. These true malicious domains can in turn substantially contribute to the quality and improvement of our SVM-based classification algorithm for detecting malicious domains.

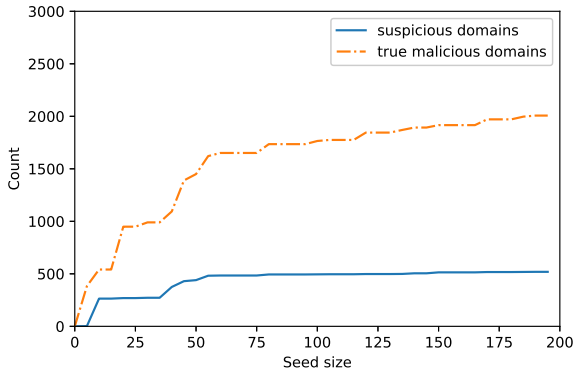


Figure 4: The number of newly discovered malicious domains with a small seed of malicious domains.

7.2.2. Discovering traffic patterns of malware. Our experimental results have shown that malicious domains in the same clusters often belong to the same malware or malware family, share similar IP addresses and ranges, and communicate with the same set of compromised end hosts on the campus networks. From the netflow traffic collected

from edge routers of the campus network, we find that the domains in the same clusters exhibit similar traffic patterns. Such traffic patterns reveal deep insight on the attack behaviors and tactics of malware.

For example, 12 domains in the same malicious cluster are all reported as spam domains by Threatbook. These domains share one single IP address, and communicate with 518 end hosts in the campus network on three destination ports: 80, 1337, and 2710. Similarly, 32 malicious domains in one cluster are all reported as C&C server domains by Threatbook. These domains share three IP addresses, and talk to 8 hosts in the campus network on the destination port 80. Our in-depth investigation reveals that these 8 compromised hosts are indeed controlled by the same botnet. Such interesting behaviors will be difficult to uncover if we simply focus on traffic patterns of the single malicious domain without correlating with multiple domains in the same cluster.

7.3. Domain Cluster Visualization

The graph embedding method can capture the characteristics of the local structure and node features of the graph. It thus learns a latent representation for each domain so that similar domains have similar domain embeddings. In this experiment, we randomly selected five domains clusters and use t-SNE [18] to reduce the dimension of the domain embeddings to a 2-dimensional space and then visualize them in Figure 5. As shown in Figure 5, the domains with strong association are close in low-dimensional space, illustrating the ability of our model to automatically learn the implicit relationships between domains.

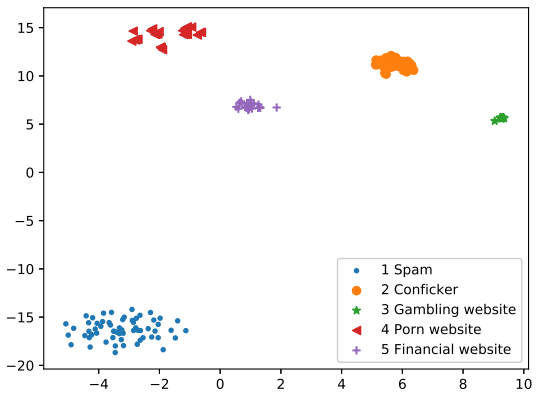


Figure 5: Domains visualization.

8. Experimental Evaluations

In this section, we first measure the accuracy of our proposed algorithm with the combined feature vector as well as each individual feature vector for detecting malicious with k-fold cross-validation. subsequently, we compare our approach with other systems.

8.1. Measuring Detection Accuracy with AUC

To understand the quality of detecting malicious domains, we adopt the widely used k-fold cross-validation method to evaluate the SVM-based classification algorithm. Firstly, we randomly shuffle the labeled domain data-set, and split the data into k groups. Subsequently, for each of these k groups, we consider it as a hold-out group as testing data-set, and use the remaining $k-1$ groups as the training data-set for developing SVM classification model. Finally, we evaluate the quality of the trained model with the hold-out group for evaluations.

After repeating the same process of each group, we calculate the average detection quality from the k trained SVM models. In our experiment, we choose $k = 10$, a common value in machine learning model evaluations, for measuring the detection quality. Figure 6 illustrates the false positive rate versus the true positive rate, also referred as the receiver operating characteristic (ROC) curve. The area under the curve (AUC) is 0.94, suggesting that our proposed system is able to effectively and accurately detect malicious domains.

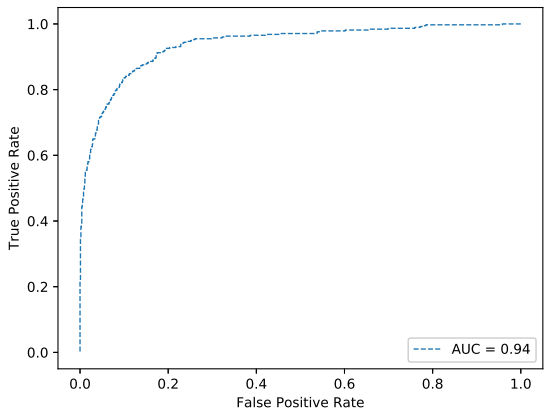


Figure 6: Measuring detection accuracy via AUC based on 10-fold cross-validation.

To qualify the contributions of each feature vector learned from graph embedding, we evaluate the contribution of each individual feature vector for detecting malicious domains. Specifically, we train three different SVM-based binary classifiers based on three feature vectors via learning domain querying behavioral similarity graph, domain IP resolving similarity graph, and domain temporal similarity graph, respectively. As shown in Figure 7, the AUCs for these SVM classifiers are 0.89, 0.83, and 0.65. The SVM classifier built on top of the feature vector learned from via learning domain querying behavioral similarity graph achieves the highest detection accuracy, while the SVM

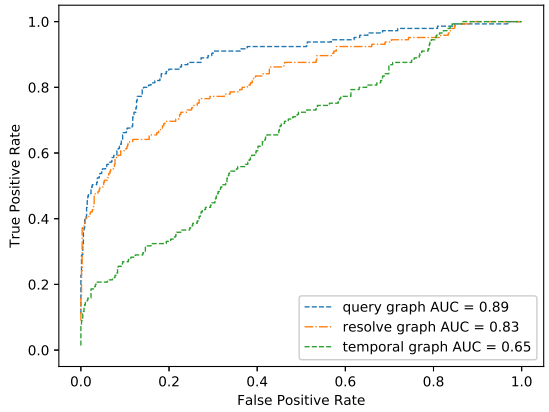


Figure 7: Qualifying the contributions of three feature vectors learned from graph embedding.

classifier based on domain temporal similarity graph has the lowest accuracy. However as illustrated in Figure 6, combining all these feature vectors together to train the classification algorithm lead to an AUC of 0.94, thus substantially improving the overall detection accuracy.

8.2. Performance Comparisons

To demonstrate the performance of our proposed algorithm in detecting malicious domains, we implement and evaluate the state-of-the-art algorithm, i.e., Exposure [19] which detects malicious domains with a similar method of passive DNS analysis. In particularly, Exposure relies on four groups of statistical features extracted from passive DNS traffic: time-based features, DNS response-based features, TTL-based features, and domain name lexical features, and develops a J48-based decision tree classifier with labeled training data.

Based on the same training data we used in this study, we extract the same four groups of features as in [19], and trains J48 binary classification classifier. The experimental result shows that the Exposure algorithm achieves an AUC of 0.88. In other words, comparing with the algorithm in [19], our proposed SVM-based classification algorithm with feature vectors learned with graph embedding improves the AUC by 6.8%.

Our in-depth analysis on the Exposure algorithm shows that statistical features of DNS traffic change over time and cross different networks, thus lowering the detection quality of Exposure in classifying malicious domains. For example, in *TTL-based features group*, Exposure uses lower TTL values to detect malicious domains. However, recent studies [4] show the TTL values of malicious domains has increased, while the average TTL value of all Internet domains is decreasing due to the wide deployment of content distribution networks (CDNs) and similar services [12]. Similarly, in *domain name lexical features group*, the percentage of numerical characters and the length of the longest meaningful substring (LMS) can not effectively detect malicious domains for non-English speaking countries where many domain names do not contain valid English words or prefer the numerical values with special meanings in local languages.

9. Related Work

The existing studies on malicious domain detection can be broadly classified into three categories based on the underlying techniques: classification-based solutions, clustering-based solutions, and graph-based techniques. The classification-based solutions first rely on domain knowledge from networking and security experts to extract relevant statistical features of benign and malicious domains. Based on such features and labeled data-sets with known malicious and benign domains, these solutions employ supervised classification algorithms such as support vector machines (SVMs) and decision trees to build binary classifiers for distinguishing benign and malicious domains [10], [19]–[23].

For example, the early study in [20] first identifies a broad range of network and zone features of domains for characterizing capacity planning, usage patterns and management of domains, then explores automated classification algorithms to model DNS behavior of known benign and malicious domains, and finally applies the classification model to compute reputation or malicious scores for new domains. Similarly, [19] extracts four groups of features from passive DNS traffic: time-based features, DNS answer-based features, TTL-based features, and domain name lexical features for training a J48 decision tree classifier with labeled training data-set. In addition to features extracted from local DNS traffic, a number of recent studies also exploit features from network traffic at the upper DNS hierarchy [21], APT malware C&C server domain features [22], host behavior features, domain activity features and IP abuse features [10], and temporal dynamics and behavior profiles of domains [23]. Differing from these studies, our proposed algorithm requires little expert knowledge for extracting domain statistical features which often exhibit different behaviors across different networks or over time. In addition, our empirical analysis shows an increasing trend of TTL values from malicious domains.

The clustering-based solutions [12], [24], [25] extract the feature similarity between domains, and utilize the similarity measures to group domains into distinct clusters. For example, [24] calculates the similarity between non-existent domains (NXDomains) at the authoritative level, and then adopts hierarchical clustering to identify the domain groups produced by domain generation algorithms (DGA). Similarly, [12] examines the temporal correlation among DNS queries, and then applies XMeans clustering algorithms with a few seed malicious domains to detect a large number of malicious domains groups, while [25] clusters the end hosts compromised by the same DGA algorithm based on their query behaviors observed in DNS traffic logs.

The graph-based solutions [26], [27] first model the behavior of domains via a variety of graphs, and then use graph mining approaches to detect malicious domains. For example, [26] builds DNS failure graphs to model the interactions between end hosts and failed domain names, and successfully extracts coherent co-clusters from DNS failure graphs with a statistical tri-nonnegative matrix factorization technique, while [27] constructs host-domain association graph for detecting malicious domains via the belief propagation (BP) inference algorithm.

Our proposed algorithm essentially combines the advantages of all three existing types of malicious domain detections since we i) rely on bipartite graphs and one-mode projections for capturing behavioral similarity of domains and explore graph embedding to learn the feature representations of domains, ii) develop a binary classifier based on SVMs for distinguishing malicious domains, and iii) run XMeans clustering for mining and discovering malicious domain clusters.

10. Conclusions and Future Work

In light of the limitations of existing approaches in detecting malicious domains, this paper introduces an end-to-end system for effectively and accurately detecting malicious domains via behavioral modeling and graph embedding which does not rely on expert knowledge, traffic or lexical features. Specifically this paper models the behavioral patterns of domains via bipartite graphs, and explores one-mode projections to capture the behavioral similarity of domains. Relying on graph embedding techniques, we successfully represent domain behaviors via feature vectors, and develop an SVM-based classification algorithm with labeled domain data-sets for predicting new domains observed in a large campus network as malicious or benign. Our extensive experimental results have shown that our proposed algorithms outperform existing studies and achieves an AUC of 0.94 for detecting malicious domains from a real and large campus network. One of our future work is to deploy our proposed system in distributed campus networks or enterprise networks and analyze the correlations of malicious domains for mining large-scale attack campaigns and detecting new and evolving botnets.

Acknowledgment

This work is supported by Shenzhen Fundamental Research Project (No. JCYJ20170412151008290) and Guangdong Natural Science Fund Project (Grant No. 2018A030313017). In addition, Kuai Xu is supported in part by the National Science Foundation under the grant CNS #1816995.

References

[1] J. Szurdi, B. Kocso, G. Cseh, J. Spring, M. Felegyhazi, and C. Kanich, "The long "taile" of typosquatting domain names." in *Proceedings of USENIX Conference on Security Symposium (USENIX Security)*, August 2014.

[2] S. Hao, M. Thomas, V. Paxson, N. Feamster, C. Kreibich, C. Grier, and S. Hollenbeck, "Understanding the domain registration behavior of spammers," in *Proceedings of ACM Internet Measurement Conference (IMC)*, October 2013.

[3] P. Kintis, N. Miramirkhani, C. Lever, Y. Chen, R. Romero-Gomez, N. Pitropakis, N. Nikiforakis, and M. Antonakakis, "Hiding in plain sight: a longitudinal study of combosquatting abuse," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, October 2017.

[4] W. Xu, X. Wang, and H. Xie, "New trends in fastflux networks," in *Proceedings of the 16th BlackHat USA*, July 2013.

[5] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of International Conference on World Wide Web (WWW)*, May 2015.

[6] S. Yadav, A. K. K. Reddy, A. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proceedings of ACM Internet Measurement Conference (IMC)*, November 2010.

[7] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, February 2008.

[8] K. Xu, F. Wang, and L. Gu, "Behavior analysis of internet traffic via bipartite graphs and one-mode projections," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 931–942, June 2014.

[9] A. Jakalan, J. Gong, Q. Su, X. Hu, and A. M. Abdelgder, "Social relationship discovery of ip addresses in the managed ip networks by observing traffic at network boundary," *Computer Networks*, vol. 100, pp. 12–27, May 2016.

[10] B. Rahbarinia, R. Perdisci, and M. Antonakakis, "Efficient and accurate behavior-based tracking of malware-control domains in large isp networks," *ACM Transactions on Privacy and Security*, vol. 19, no. 2, pp. 4:1–4:31, September 2016.

[11] I. Khalil, T. Yu, and B. Guan, "Discovering malicious domains through passive dns data graph analysis," in *Proceedings of ACM Asia Conference on Computer and Communications Security (ASIACCS)*, June 2016.

[12] H. Gao, V. Yegneswaran, J. Jiang, Y. Chen, P. Porras, S. Ghosh, and H. Duan, "Reexamining dns from a global recursive resolver perspective," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 43–57, February 2016.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, December 2013.

[14] VirusTotal, https://www.virustotal.com.

[15] B. Schölkopf, A. J. Smola *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[16] D. Pelleg, A. W. Moore *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters." in *Proceedings of International Conference on Machine Learning (ICML)*, June 2000.

[17] Threatbook, https://x.threatbook.cn.

[18] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, November 2008.

[19] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive dns analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security*, vol. 16, no. 4, pp. 14:1–14:28, April 2014.

[20] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for dns." in *Proceedings of USENIX Conference on Security Symposium (USENIX Security)*, August 2010.

[21] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, "Detecting malware domains at the upper dns hierarchy." in *Proceedings of USENIX Conference on Security Symposium (USENIX Security)*, August 2011.

[22] G. Zhao, K. Xu, L. Xu, and B. Wu, "Detecting apt malware infections based on malicious dns and traffic analysis," *IEEE Access*, vol. 3, pp. 1132–1142, July 2015.

[23] D. Chiba, T. Yagi, M. Akiyama, T. Shibahara, T. Yada, T. Mori, and S. Goto, "Domainprofiler: Discovering domain names abused in future," in *Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2016.

[24] M. Thomas and A. Mohaisen, "Kindred domains: detecting and clustering botnet domains using dns traffic," in *Proceedings of International Conference on World Wide Web (WWW)*, April 2014.

[25] T. S. Wang, H. T. Lin, W. T. Cheng, and C. Y. Chen, "Dbod: Clustering and detecting dga-based botnets using dns traffic analysis," *Computers & Security*, vol. 64, pp. 1–15, January 2017.

[26] N. Jiang, J. Cao, Y. Jin, L. E. Li, and Z.-L. Zhang, "Identifying suspicious activities through dns failure graph analysis," in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, October 2010.

[27] P. K. Manadhata, S. Yadav, P. Rao, and W. Horne, "Detecting malicious domains via graph inference," in *Proceedings of European Symposium on Research in Computer Security (ESORICS)*, September 2014.