Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 102687, 12 pages
http://dx.doi.org/10.1155/2015/102687

*Research Article*

# Detecting Malware Based on DNS Graph Mining

**Futai Zou,[1] Siyu Zhang,[2] Weixiong Rao,[3] and Ping Yi[1]**

[1]*School of Information Security Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*
[2]*Network and Information Center, Shanghai Jiao Tong University, Shanghai 200240, China*
[3]*School of Software Engineering, Tongji University, Shanghai 201804, China*

Correspondence should be addressed to Futai Zou; zoufutai@sjtu.edu.cn

Received 24 August 2014; Accepted 17 April 2015

Academic Editor: Qingquan Zhang



Malware remains a major threat to nowadays Internet. In this paper, we propose a DNS graph mining-based malware detection approach. A DNS graph is composed of DNS nodes, which represent server IPs, client IPs, and queried domain names in the process of DNS resolution. After the graph construction, we next transform the problem of malware detection to the graph mining task of inferring graph nodes' reputation scores using the belief propagation algorithm. The nodes with lower reputation scores are inferred as those infected by malwares with higher probability. For demonstration, we evaluate the proposed malware detection approach with real-world dataset. Our real-world dataset is collected from campus DNS servers for three months and we built a DNS graph consisting of 19,340,820 vertices and 24,277,564 edges. On the graph, we achieve a true positive rate 80.63% with a false positive rate 0.023%. With a false positive of 1.20%, the true positive rate was improved to 95.66%. We detected 88,592 hosts infected by malware or C&C servers, accounting for the percentage of 5.47% among all hosts. Meanwhile, 117,971 domains are considered to be related to malicious activities, accounting for 1.5% among all domains. The results indicate that our method is efficient and effective in detecting malwares.

## 1. Introduction

Malwares such as Trojans, worms, spyware, and botnets are rampant and have seriously threatened user privacy, social economy, and national security. As the key part of Internet infrastructure, DNS has been widely exploited by malicious software. The well-known IP-Flux, Domain-Flux, and DNS tunnel are those tightly related to DNS. It has become an exploding research to detect malware based on DNS technology for these years, and we highlight the reason as follows. First of all, independent of C&C protocols, DNS-based detection is very useful to detect various kinds of malwares. Second, DNS-based malware detection systems can be easily deployed onto large-scale networks because the detection cost associated with DNS is trivial compared with those based on the analysis of the entire Internet traffic. Finally, DNS-based detection is also applicable to detect malwares that encrypt their control communications.

Among those malwares that leveraged DNS, Fast-Flux has been used by botnets to build malicious content distribution networks by rapidly changing IP address [1, 2]. Domain-Flux malware can generate a large number of pseudorandom domain names in order to escape domain blacklists [3, 4]. Dietrich et al. [5] discovered and analyzed a botnet C&C based on DNS tunnel, a special DNS exploit but with a very limited number of detectable types of malwares. Overall, all of the above malware detections are dedicated to specific kinds of malwares and thus with the limitations of detecting only such kind of malwares alone.

There do exist related works utilizing machine learning methods to design a generic malware detection system. Notos [6] designed a dynamic domain reputation system based on these three features: network addresses, DNS zones, and malicious clues. EXPOSURE [7] recognizes malicious domains such as malware, phishing site, by a classification algorithm based on DNS traffic features (including query time, response content, TTL, and domain). Kopis [8] took advantage of the information of upper authoritative domain server to mine malicious domain based on the distribution

of the requesters, the properties of the requesters, and the reputation of the resolvers.

In this paper, we propose a graph mining-based malware detection algorithm. Specifically, we first build the DNS graph by the relationship of domains and IPs. Next, we compute the reputation of those domains and IPs by using the belief propagation algorithm on the DNS graphs. Finally, we use the computed reputation scores to identify those malicious domains, control servers, and victim hosts infected by malware. During the above three steps, we use client IPs, domain names, and the result of domain resolution as the data source to construct DNS graph, exploit the inner relationship between IP and domain to find the homogenization relationship on DNS graph, and finally infer reputation of nodes by applying the belief propagation algorithm.

Compared with the aforementioned approaches [1–8], the malware detection approach proposed in this paper makes the following contributions. (1) The DNS graph in this paper includes both servers and clients, and thus our approach can comprehensively detect malware servers and the associated victims. (2) The DNS graph can cover all domains, and our approach has the ability to identify the overall architecture of the detected malwares.

The rest of this paper is organized as follows. In Section 2, we describe the DNS graph model. Section 3 presents the DNS graph mining algorithm. Section 4 presents the dataset, evaluation method, and the experimental results. Section 5 discusses related works. Finally, Section 6 presents our conclusion.

## 2. DNS Graph Model

As the base of our graph mining algorithm to detect malwares, we first describe the DNS graph model. Given a DNS graph $G = (V, E)$, the vertex set $V$ consists of hosts and domain names (we use IP addresses to uniquely identify the hosts), and the edge set $E \subseteq V \times V$ represents the connections between domain names. We will define two DNS graphs: DNS query response graph (DQRG) and passive DNS graph (PDG).

*2.1. DNS Query Response Graph (DQRG).* By analyzing DNS traffic between clients and recursive DNS servers, we build a DNS query response graph $G_{\mathrm{QR}} = (V, E)$ as follows. First, if a client cli queries the domain name $d$, we add an edge $e_q = (v_{\mathrm{cli}}, v_d) \in E$ between cli and $d$. Next, as the domain name $d$ is resolved to a set of IP addresses $\{\mathrm{addr}_1, \mathrm{addr}_2, \ldots, \mathrm{addr}_N\}$, we then connect the domain name $d$ with all of the IP addresses by the edges $\{e_{r_1}, e_{r_2}, \ldots, e_{r_N}\} \subseteq E$, where $e_{r_i} = (v_d, v_{\mathrm{addr}_i})$, $i = 1, \ldots, N$, represents the DNS record relating $d$ and $\mathrm{addr}_i$.

In the graph $G_{\mathrm{QR}}$, we ignore the CNAME chain in domain name resolution and connect the query domain QNAME directly to the final resolved IP address. In this way, the edges $e_{r_i}$ directly indicate the relationship between a domain name and server IP address. Therefore, DQRG is a bipartite graph. Vertex set $V$ can be divided into disjoint subsets $H$ and $D$, where $H$ are host nodes and $D$ are domain nodes. An edge $e = (u, v)$ in $E$ connects two vertices $u$ and $v$ which is in set $H$ and $D$, respectively.

*2.2. Passive DNS Graph.* Passive DNS graph (PDG) can be constructed using resource records of DNS response traffic or from a passive DNS database. Passive DNS [9] collects data or resource records in DNS system by monitoring DNS traffic. It is widely used in security researches. ISC (Internet Systems Consortium) and other organizations have implemented and deployed passive DNS systems. The passive DNS data do not contain client information, which may leak privacy info of users.

PDG $G_{\mathrm{PASV}} = (V, E)$ uses A and CNAME records. For the name $d$ of A records and IP address addr in RDATA, we map them to two vertices $v_d \in V$ and $v_{\mathrm{addr}} \in V$, connected with an edge $e_{\mathrm{addr}} = (v_d, v_{\mathrm{addr}}) \in E$. CNAME records in $G_{\mathrm{PASV}}$ represent relationships among domain names, with an edge $e_{\mathrm{cname}} = (v_d, v_{\mathrm{cname}}) \in E$ from the name $d$ to its canonical name cname.

In order to illustrate the construction of DQRG (referred as $G_{\mathrm{QR}}$) and PDG (referred as $G_{\mathrm{PASV}}$) more clearly, we demonstrate the process of building such two graphs through an example. If a client with the IP address 1.2.3.4 queries the domain name http://www.bing.com/ to a DNS server and obtains the resolution result (as shown in Figure 1(a)), we then build $G_{\mathrm{QR}}$ in Figure 1(b) and $G_{\mathrm{PASV}}$ in Figure 1(c).

In $G_{\mathrm{QR}}$, when client 1.2.3.4 queries domain name http://www.bing.com/, we add an edge between these two vertices. And as http://www.bing.com/ resolves into two IP addresses 124.40.41.15 and 124.40.41.17, it also forms two edges. CNAME as an intermediate result is ignored during the construction of $G_{\mathrm{QR}}$.

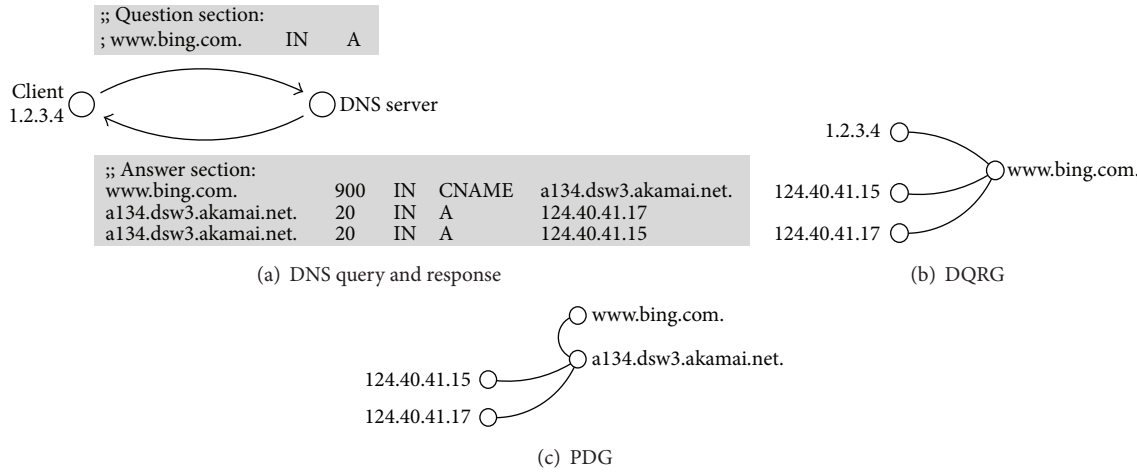
$G_{\mathrm{PASV}}$ does not simply cut out client node from $G_{\mathrm{QR}}$. The construction of $G_{\mathrm{PASV}}$ is entirely based on the resource records in the DNS answer. The three resource records in Figure 1(a) is one to one correspondent to three edges of $G_{\mathrm{PASV}}$. Currently, $G_{\mathrm{PASV}}$ only uses A and CNAME records. In future expansion of PDG, MX, SRV, NS, PTR, and other records that may also produce edges, AAAA records and IPv6 addresses should be supported.

## 3. DNS Graph Mining Algorithm

In this section we first introduce Markov Random Field model, with the extended DNS graph marking the reputation of each node (host or domain name). On the basis of the Markov Random Field, we use the belief propagation algorithm to explore the relationship between malware host and domain name, thus conducting reputation inference of DNS graph nodes.

*3.1. Markov Random Field.* Markov Random Field (MRF) is a set of random variables with Markov properties described by an undirected graph. A MRF contains an undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, N\}$ is the collection of vertices. Each vertex corresponds to a random variable $x_i$, $i = 1, \ldots, N$. For a given vertex $i$, we denote the set of its neighbor nodes to be $N_i$ with $(i, j) \in E$ and $j \in N_i$. MRF conform to the local property:

$$\Pr\left(x_i \mid \{x_j\}_{j \in V \setminus i}\right) = \Pr\left(x_i \mid \{x_j\}_{j \in N_i}\right). \quad (1)$$




(a) DNS query and response

(b) DQRG

(c) PDG

Figure 1: Demonstration of the construction of a DNS graph.

That is, the random variable $x_i$ of vertex $i$ only depends on its neighbor nodes, and $x_i$ has no relationship with random variable of nonneighbor nodes [10].

Let us apply the MRF model to DNS graph. Each vertex of DNS graph corresponds to a random variable, which represents the category of host or domain name. For malware detection, we define the label set $L = \{l_{\text{good}}, l_{\text{mal}}\}$, where $x_i = l_{\text{good}}$ indicates that the host or domain name represented by this vertex is benign, and $x_i = l_{\text{mal}}$ otherwise indicates that the host is infected with malware or is a C&C server or malware related domain. Vertex's reputation $\text{rep}_i = \Pr(x_i = l_{\text{good}})$ is a number between 0 and 1, representing the probability that the node is normal. The goal of the mining algorithm on DNS graph is to calculate each vertex's reputation $\text{rep}_i$, with part of vertex labels, and relationships between vertices are previously known.

*3.2. Belief Propagation Algorithm.* Calculating the marginal probability or vertex labels described in Section 3.1 is NP-hard. The time required to calculate the reputations is exponential to the number of vertices. By heuristic method, belief propagation [11] (BP) algorithm approximately calculates marginal probability by iterative message passing, and its execution time only linearly increases with the number of vertices [12]. The BP algorithm has been widely used in artificial intelligence, image processing, and coding.

The basic idea of the message passing algorithm is that every vertex in the graph told its neighbor what it thinks its neighbors' labels are. Or, more precisely, it told its neighbors the probability of obtaining each label. Suppose that each vertex is associated with $|L|$ possibilities; there are $|L|$ messages to be transferred from a vertex $i$ to its neighbor $j$.

Firstly, according to prior knowledge, the probability of each label $x_i$ of node $i$ is marked as function $\phi_i(x_i)$, $x_i \in L$, which is called the evidence of vertex $i$. The inference between neighbor nodes is shown as function $\varphi_{ij}(x_i, x_j) = \Pr(x_i \mid x_j)$, that is, the conditional probability for node $i$ to have label $x_i$ when node $j$ has label $x_j$. Since MRF is undirected, this function is symmetric for neighboring nodes $i$, $j$; that is, $\varphi_{ij}(x_i, x_j) = \varphi_{ji}(x_j, x_i)$. The goal of the algorithm is to calculate the marginal probability for nodes to have each label, and we call the approximate marginal probability calculated by this algorithm as "belief," denoted as $b_i(x_i)$.

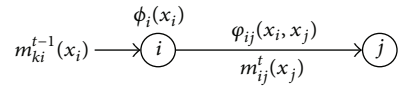



Figure 2: Message passing.

The principle of BP algorithm is based on message passing. The message is essentially the viewpoint of vertex $i$ on the possibility of its neighbor vertex $j$ holding a label $x_j$. The message passed from $i$ to $j$ marks as $m_{ij}(x_j)$, as shown in Figure 2. The calculation is as follows. For $\forall e_{ij} \in E$ and every possible label, calculate $m_{ij}(x_j)$ and $m_{ji}(x_i)$. All messages are being passed once in each iteration and the order of message delivery can be arbitrary. The message passed from $i$ to $j$ is generated by calculation from other neighbor messages of $i$. It can be denoted as follow:

$$m_{ij}^{t}\left(x_j\right) \longleftarrow \sum_{x_i \in L}\left(\phi_i\left(x_i\right)\varphi_{ij}\left(x_i, x_j\right)\prod_{k \in N_i \setminus j} m_{ki}^{t-1}\left(x_i\right)\right); \tag{2}$$

$N_i$ is the set of neighbor vertices of $i$, and $m_{ki}^{t-1}$ shows the message passed in last iteration. Also, above formula is called "Sum-Product" algorithm due to its form.

With this method, message delivery can be carried out in parallel on all edges. In the beginning of calculation, for all $(u, v) \in E$, $x_v \in L$, message $m_{uv}^{0}(x_v)$ must be initialized first.

In practice, messages need to be normalized. Otherwise, after many message delivery iterations, the value of the message will overflow or underflow (become smaller gradually and turn to "0" under the floating-point precision range).

Message normalization is conducted at the vertex receiving messages:

$$\sum_{x_j \in L} m_{ij}\left(x_j\right) = 1. \tag{3}$$

In each iteration, the belief calculation of vertex is based on the messages received from neighbors, specifically:

$$b_i\left(x_i\right) = \frac{1}{Z_i}\phi_i\left(x_i\right)\prod_{j \in N_i} m_{ji}\left(x_i\right); \tag{4}$$

$Z_i$ is the normalization constant as the sum of beliefs on each label for vertex $i$ should be 1.

BP algorithm starts with the initial belief sets which are given by the prior knowledge for some vertices; then messages are continuously iterated until the belief of the vertex is convergent (variation is less than a certain threshold) or algorithm running iterative time reach the limit.

*3.3. Definition of Inferred Relationship.* DNS graph provides the relation of host querying domain names and the domain name resolving to some IP addresses. By understanding the relationship among hosts infected by malware, malware domain, and C&C server, intuition tells us that the connected edge between IP address and domain name in DNS graph represents the potential homogenization laws:

(1) Host infected by malware will proactively query malware domain, and, in general, the malware domain is only queried by the infected host. Thus, we call it relevance between infected host and malware domain.

(2) It is rare that legitimate domain will resolve to malicious IP addresses. We can just see that the IP address of malware domain is related to C&C server. We call it relevance between malware domain and C&C server.

(3) A nonmalicious domain is often associated with nonmalicious server or CDN (Content Delivery Network) service, and the CDN usually only serves legitimate websites. It is highly impossible that IP addresses of C&C server are in the range of IPs owned by a large website. Therefore, we call it relevance between nonmalicious domain and nonmalicious server.

The above laws mean that the message delivery mechanism of BP algorithm ideally suits for malware mining based on DNS graph. The purpose of mining malware is to let the infected hosts, the malicious domains, and C&C server's IP addresses have a lower reputation, in other words, have a high probability $\Pr(x_i = l_{\text{mal}})$. Meanwhile, let the legitimate domain names, servers, and normal hosts have a high reputation.

In the BP algorithm, in order to express this homogenization relationship, we define the inference function $\varphi_{ij}(x_i, x_j)$ to be the conditional probability $\Pr(x_i \mid x_j)$. We expressed it in inference probability matrix form shown in (5), where $0 < \varepsilon < 0.5$.

Obviously, it is consistent with the meaning of above three laws that the definition of inference probability matrix denotes that the adjacent nodes of the legal nodes (domain or host) are more likely to be legitimate. And the adjacent nodes of malware nodes are also more likely to be malicious. In the implementation, $\varepsilon$ takes the smallest value $\varepsilon = 0.01$ to distinguish nuance of probability. Consider

$$\begin{array}{c|cc} \varphi_{ij}\left(x_i, x_j\right) & x_i = l_{\text{good}} & x_i = l_{\text{mal}} \\ \hline x_j = l_{\text{good}} & 0.5 + \varepsilon & 0.5 - \varepsilon \\ x_j = l_{\text{mal}} & 0.5 - \varepsilon & 0.5 + \varepsilon \end{array} \tag{5}$$

*3.4. Prior Knowledge.* Prior knowledge is a preset reputation $\text{rep}_i$ for some part of nodes of DNS graph, that is, the probability of labels $x_i = l_{\text{good}}$ and $x_i = l_{\text{mal}}$. Prior knowledge as the initializing data of BP algorithm helps to infer the reputation of other "unknown" nodes. For the vertices without prior knowledge, its probability of being legal and malicious is equal at the beginning; that is, $\text{rep}_i = \Pr(x_i = l_{\text{good}}) = \Pr(x_i = l_{\text{mal}}) = 0.5$. Other nodes tend to be legitimate if initial reputation is higher than 0.5 or tend to malicious if initial reputation is lower than 0.5.

Considering the requirement of detecting malware based on DNS graph, we use the following domain and IP prior knowledge of BP algorithm.

*3.4.1. Domain Prior Knowledge.* Domain prior knowledge that we use contains a collection of well-known legitimate domains, a collection of known malicious domains, and domain reputation based on Alexa ranking and freely registerable subdomains.

Sources of known malicious domains include DNS-BH and MDL (Malware Domain List) and other domain lists for malicious software publicly available on the Internet, as well as those C&C domains for Zeus, Palevo, and SpyEye tracked by abuse.ch. For malicious software in China, we also use domain source like CNCERT/CC and ANVA. Malicious domain in blacklists receives an initial reputation of 0.01.

At the same time, in the experiment, we collect domain suffixes used by DDNS provider such as No-IP, DynDNS, and 3322.org, as well as ccTLDs often exploited by malware authors, as the clue of suspicious domains.

In the legitimate part, we use the Alexa ranking data as reference. Top 100 Alexa ranking domains are considered as legitimate, with initial reputation set to 0.99. For domains ranked 101 to 10,000, we set the reputation value based on their ranking using a nonlinear mapping function shown in Figure 3. The lower the ranking is, the closer the reputation is to 0.5 (the unknown state).

*3.4.2. Host Prior Knowledge.* The principle of host prior knowledge is similar to that of domains. We collect a list of known legal or malicious IP addresses for initial host reputation.

Tracking systems run by abuse.ch provide C&C IP addresses of Zeus, Palevo, and SpyEye botnets. Also, MDL provides blacklist for active malware IP addresses. DROP (Do not Route Or Peer) list by Spamhaus includes network segments hijacked or controlled by hacker groups. IP addresses

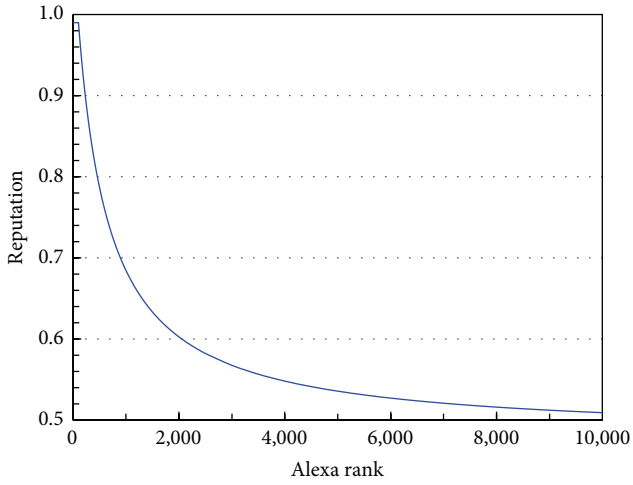



Figure 3: Alexa reputation function.

on the blacklists are considered known malicious; thus, the initial reputation is set to 0.01.

OpenBL.org and Brute Force Blocker report IP addresses conducting various kinds of Internet attack. In this paper, these attackers' IP addresses are used as clues for low reputation ($0.01 < \text{rep} < 0.5$).

In the terms of legal IP addresses, all IP segments assigned to Google and Microsoft are considered to be legal, which can be obtained from DNS SPF records.

## 4. Experimental Evaluation

*4.1. Data Collection and Algorithm Implementation.* DNS data used in this paper are extracted from mirrored traffic of 5 DNS servers in Shanghai Jiao Tong University campus network. More than 250 million DNS queries are processed by these servers daily (3000 QPS on average).

In order to build DNS graph from real-time traffic, we built a program to analyze the DNS query and response packets and store them in the form of vertices and edges into a database directly. Vertices are stored as $\langle vid, vtype, vname\rangle$ while edges are stored as $\langle vid_1, vid_1 2\rangle$, where *vtype* indicates the type of the vertex (domain or host) and *vname* is the domain name or the IP address. Since DNS graphs are undirected, we eliminate the possibility of multiple edges during database insert.

We monitor the traffic for 7 consecutive days to produce DQRG. By recording all the DNS queries and responses, the final DQRG has 9,335,270 vertices and 102,004,729 edges. In order to get more complete and comprehensive observation results of DNS data, we take 3 months to collect data for PDG. The resulting PDG consists of 19,340,820 vertices and 24,277,564 edges. Simply comparing it with DQRG, we can find that PDG was relatively sparse, which means that the average number of queried domains by a client machine is much higher than the number of domains mapped to a single server.

We show the statistics of the two DNS graphs in Table 1. The database size is the data size after storing in a MySQL database (does not include the index), while the graph file comprises an edge list and a vertex weight file (does not include domain name or IP address strings).

Table 1: Statistics of DNS graphs.

| | DQRG | PDG |
|---|---|---|
| Data collection period | 7 days | 3 months |
| Vertex number | 9,335,270 | 19,340,820 |
| Edge number | 102,004,729 | 24,277,564 |
| Database size | 1,004 MB | 1,235 MB |
| Graph file size | 1,373 MB | 672 MB |

Since the number of vertices and edges in the graph is huge, the efficiency of DNS graph mining algorithm is critical. According to the analysis in Section 3.2, the message delivery process for BP algorithm can be parallelized; we use a parallel computing framework, GraphLab [13], to implement the algorithm. The structured prediction programs contained in the GraphLab toolkit include a reference implementation of Loopy BP. The BP implementation in GraphLab divides the algorithm into three operations, Gather, Apply, and Scatter, which maps to message receiving, belief calculating, and message passing, respectively. Since GraphLab provides two types of operation for the message receiving (Sum and Max), operations of BP algorithm is done in Log-space. In Section 3.3, we use $\varepsilon = 0.01$, which also corresponds to *smoothing* $\approx 4.6$ in Log-space. The convergence threshold is 0.005 in our experiment; thus, the BP iteration continues until the convergence:

$$\sum_{x_j \in L} \left|\log\left(m_{ij}^{t}\left(x_j\right)\right) - \log\left(m_{ij}^{t-1}\left(x_j\right)\right)\right| \leq 0.005. \quad (6)$$

The machine in the experiment is equipped with two Intel Xeon E5620 processors and a total of 24 GB physical memory. Each processor has 4 cores with hyperthreading enabled, running the 2.4 GHz. The server runs 64-bit Ubuntu Linux Server, and the version of GraphLab is 2.1.4434. We run the algorithm on a single machine, using 16 threads in parallel.

*4.2. Evaluation Methods.* On PDG and DQRG collected in Section 4.1, first we use prior knowledge in Section 3.4 to label the vertices' initial reputation.

The statistics of the distribution of vertex reputation is shown in Table 2. 22.8% vertices in DQRG and 31.9% vertices in the PDG are covered by blacklists, whitelists, and other reputation data from Section 3.4. But only 1/4 of these vertices are in the known blacklist or whitelist (with initial reputation 0.01 or 0.99). The results show that the prior knowledge data in Section 3.4 have a desirable effect in the initialization for DNS graph. Meanwhile, more than 90% of all vertices are not in a blacklist or whitelist, on which our algorithm has positive significance in reality to detect and judge.

In the evaluation, we adopt true positives (TP) and false positives (FP) as the main evaluation metrics. To accurately evaluate the effect of the algorithm, we use 10-fold cross validation for the test. First, we randomly divide the vertices in blacklist and whitelist into ten equal pieces. Then we

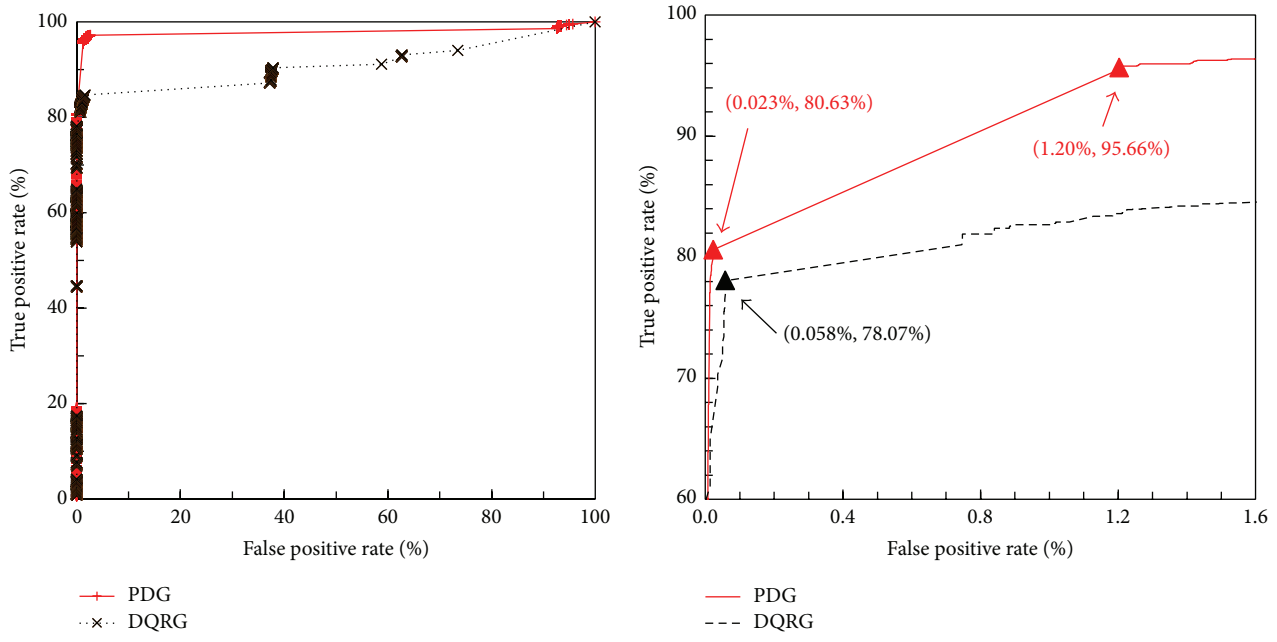
Table 2: Initial reputation of vertices.

(a) DQRG

| Category | Initial reputation | Host | Domain | Total |
|---|---|---|---|---|
| White listed | 0.99 | 6,047 | 486,026 | **492,073** |
| Good reputation | 0.5 < rep < 0.99 | 0 | 1,264,835 | **1,264,835** |
| Unknown | 0.5 | 1,599,724 | 5,604,221 | **7,203,945** |
| Bad reputation | 0.01 < rep < 0.5 | 11,612 | 343,270 | **354,882** |
| Black listed | 0.01 | 1,412 | 18,123 | **19,535** |
| Total | | **1,618,795** | **7,716,475** | **9,335,270** |

(b) PDG

| Category | Initial reputation | Host | Domain | Total |
|---|---|---|---|---|
| White listed | 0.99 | 10,236 | 1,561,571 | **1,571,807** |
| Good reputation | 0.5 < rep < 0.99 | 0 | 3,054,734 | **3,054,734** |
| Unknown | 0.5 | 3,456,157 | 9,712,161 | **13,168,318** |
| Bad reputation | 0.01 < rep < 0.5 | 24,965 | 1,497,372 | **1,522,337** |
| Black listed | 0.01 | 2,142 | 21,482 | **23,624** |
| Total | | **3,493,500** | **15,847,320** | **19,340,820** |



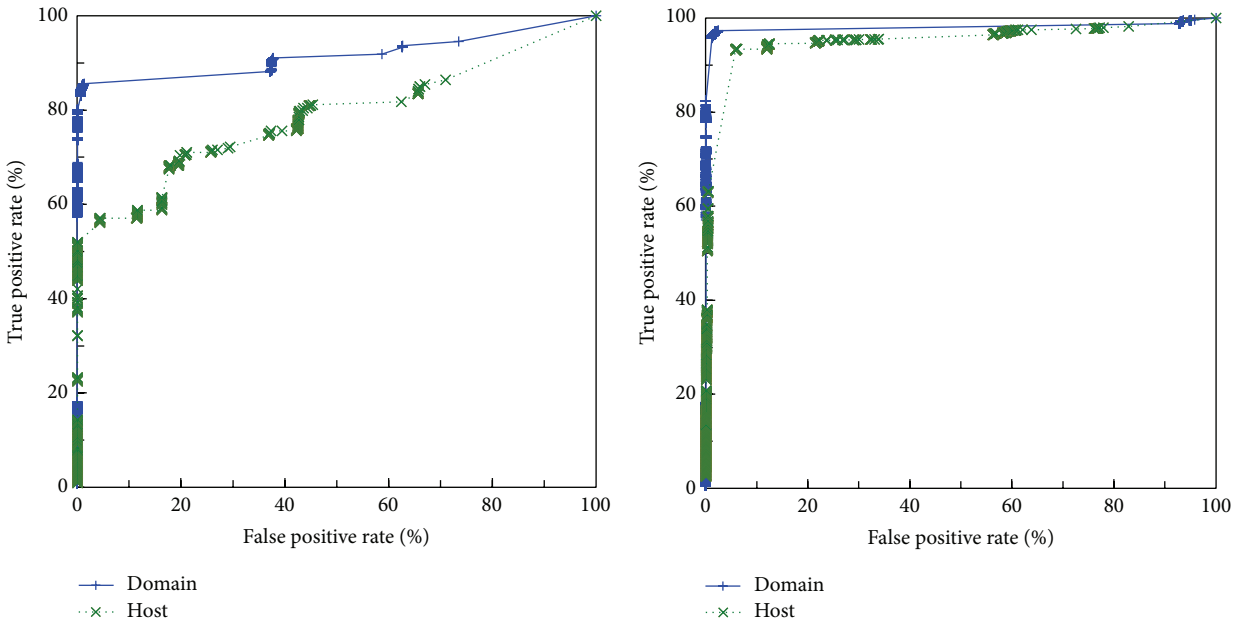

(a) ROC curve (b) Threshold decision

Figure 4: ROC curve of DNS graph mining.

operate belief propagation for ten rounds. In each round, we choose one out of ten pieces as test set. To test set, we do not simply remove these vertices from DNS graph but update its initial state as 0.5 (unknown state), and then we set the threshold to distinguish between legitimate and malicious vertex and evaluate TP and FP of malware detection.

*4.3. Evaluation and Analysis.* The legitimate and malicious node is classified according to the calculating result of node's reputation with the selected threshold. The threshold is denoted as *Thre*. If node $i$ with the reputation is inferred by BP algorithm $\text{rep}_i < \textit{Thre}$, the node is determined as malicious, which means the node is either malware-related or malware-infected.

The threshold is variable; therefore, detection algorithm can trade off between true positive rate and false positive rate. ROC (Receiver Operating Characteristic) curve reflects the detection effectiveness with different threshold. We draw ROC curve in Figure 4(a) for DQRG and PDG. The larger the area under ROC curve (AUC (Area Under the Curve)),




(a) DQRG (b) PDG

Figure 5: ROC curve comparison for host and domain.

the better the result. ROC curve shows that the overall effectiveness of detection on passive DNS is better than that on DQRG.

With the ROC curve, under different requirements, we can balance the true positive rate and false positive rate to select the most appropriate threshold. We enlarge the top left corner of the ROC curve and show it in Figure 4(b) and mark several most ideal combinations of true positive rate and false positive rate.

For DQRG, our algorithm can achieve 78.07% true positive rate with 0.058% false positive rate (threshold $Thre = 0.5$). To continue increasing tolerance of the false positive rate does not significantly improve the true positive rate (82.9% true positive rate with 1% false positive rate); therefore, we decide that $Thre = 0.5$ is the best choice for DQRG. False positive rate as low as 0.058% definitely meets the requirements of practice detection system, and the true positive rate close to 80% can also effectively capture most malware threats.

For PDG, we discuss two types of application scenarios. When our mining algorithm is used for standalone malware detection, to reduce the disturbance of false alarms, it is usually required to have a very low false positive rate. With $Thre = 0.5$, PDG achieved a true positive rate of 80.63%, while the false positive rate is only 0.023%, which is far better than the result of DQRG. In another scenario, our algorithm can be used in a reputation system. In this scenario, false positive rate around 1% is fine [14, 15] because the system usually combines other mechanisms to reduce the false positive rate of reputation evaluation. Thus, we choose $Thre = 0.5176$, which improves the true positive rate to 95.66% with 1.20% false positive rate.

Vertices in DNS graph are of two types: hosts and domains. In previous experiments, we suppose they are using the same threshold. In order to investigate the difference between two types of nodes, we draw ROC curves for the domain and host nodes in DQRG and PDG separately. The result is shown in Figure 5. According to the ROC curve, the reputation evaluation of domain nodes is more accurate than that of host nodes.

We interpret these results as follows:

(1) By analyzing false positives and false negatives, we summarize the main reason for these errors. The inferred reputation of most false alarms or misses is still 0.5 after belief propagation. It is because these nodes are not connected to any nodes that have prior knowledge. Therefore, the prior knowledge cannot affect the reputation of these nodes during message passing. As the result, reputation of these nodes remains 0.5 after the computation. This also explains, to achieve low false positive, why the threshold we selected is $Thre = 0.5$. When the threshold is above 0.5, many legitimate nodes which are not effectively evaluated will become false positive. We observe that this phenomenon is particularly evident in the PDG. The main reason is mentioned in Section 4.1, where PDG is sparser compared to the DQRG.

(2) Comparing the difference between DQRG and PDG result, we find the cause of weaker performance on DQRG. There are a large quantities of client hosts in DQRG. Although we suppose that client host nodes in DQRG is credible, some client host nodes are faked due to IP spoofing, an attacking technic sending DNS

Table 3: Performance test result.

| | DQRG | PDG |
|---|---|---|
| Vertices | 9,335,270 | 19,340,820 |
| Edges | 102,004,729 | 24,277,564 |
| Number of update operations | 35,994,145 | 61,199,201 |
| Memory usage | 21,944 MB | 10,310 MB |
| Running time | **496.5 s** | **111.9 s** |

packets with forged source IP address. As a result, it degrades the performance of DQRG. However, it would not affect the performance of PDG because there have not client host node in PDG. To overcome this shortcoming, the filtering method [16] of IP spoofing can be applied.

(3) Comparing the detection result difference between host and domain, we find out why detection of host nodes on two DNS graphs is weaker than that of domain nodes. We give the conclusion that the acquisition of prior acknowledge is an important reason. Observing the statistics of prior knowledge shown in Table 2, there is not any "good reputation" for hosts and the whitelist because IP address only includes IP segments of Microsoft and Google, an obvious small coverage, which gives rise to a great limitation on host detection. Therefore, it would effectively improve the detection performance if we expand the data source of host prior knowledge, for example, adding more IP addresses to whitelist.

*4.4. Algorithm Efficiency.* Since the algorithm efficiency is also an important metric, we evaluate it based on hardware environment mentioned in Section 4.1. The server used in the experiment has 16 logical processors and we maximize the parallelization to 16 threads to accelerate the calculation. The result was summarized in Table 3.

The belief propagation calculation on DQRG took about 8 minutes, and the memory consumption is 21.4 GB. The calculation on PDG took less than 2 minutes, and the memory required is 10.0 GB. The result shows that our implementation of the algorithm has excellent efficiency and can be feasibly applied to much larger dataset.

*4.5. Real-World Application.* The evaluation in Section 4.3 is based on the test dataset with known black and whitelist. In this section, we will use the real-world dataset described in Table 1 with all initial prior knowledge to conduct reputation inference.

First, we test actual detection effectiveness on the DQRG, which is used to evaluate the performance of reputation algorithm in real-world. We choose the DQRG because it can not only detect the malicious domains and C&C servers but also identify the client machines infected with those malware in local network, which has more significance for network administration.

According to the results in Section 4.3, we use the threshold $Thre = 0.5$. The detection result is shown in Table 4. We determined that 88,592 hosts are infected with malware

Table 4: Real-world detection result.

| Classification | Host | Domain | Total |
|---|---|---|---|
| Malicious | 88,592 | 117,971 | **206,563** |
| Legitimate | 1,530,203 | 7,598,504 | **9,128,707** |

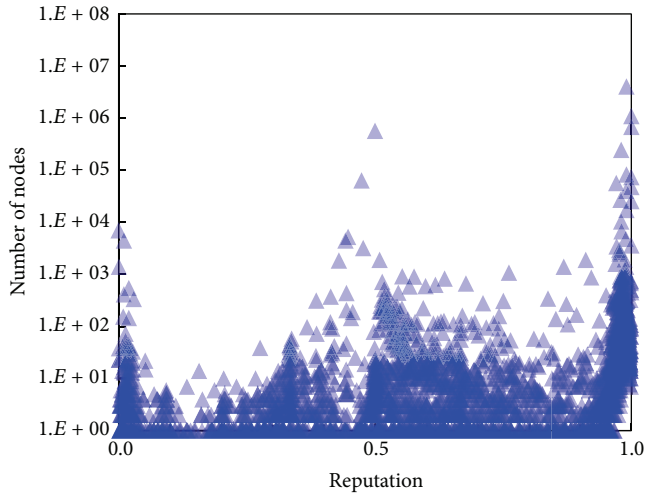



Figure 6: Reputation distribution of previously unknown nodes.

or are C&C servers, accounting for 5.47% of all IP nodes. Meanwhile, 117,971 domains are considered to be related to malicious activities, accounting for 1.5% of all domains.

We call the node that is not covered by prior knowledge "unknown node" and their initial reputation is 0.5. Figure 6 shows the reputation distribution of these unknown nodes after inference.

In the DQRG, the total number of "unknown nodes" is 7,203,945. After the reputation inference, 6,651,349 vertices (92.3%) get the reputation evaluated; namely, the reputation is no longer the initial value 0.5. The result shows that the belief propagation algorithm on DNS graph can effectively infer the reputation for most of unknown nodes.

Among these previously unknown nodes, 6,552,754 are determined as legitimate, while 98,595 are determined as malicious, accounting for 1.37% of all unknown nodes.

In real-world application, PDG is more suitable to explore the internal data structure and association of DNS system, because the PDG does not contain any client information and is constructed only with DNS resource record.

Because all host nodes in PDG are IP address from domain resolution, we can get the distribution of C&C server as shown in Figure 7. Meanwhile, domain nodes in PDG are all from valid resource records, unlike in DQRG where invalid and nonexistent domain node may exist; we can more accurately get the TLD distribution of malware domains based on the detection result on PDG, as shown in Figure 8.

Also, we use GeoLite City [17] database to look up the country, city, and coordinate of IP addresses and find that the USA, China, and Russia are top 3 countries where malware servers are located. Meanwhile, some ccTLDs are often used by malware, which is consistent with our assumption in

Figure 7: Geographical distribution of malware servers.

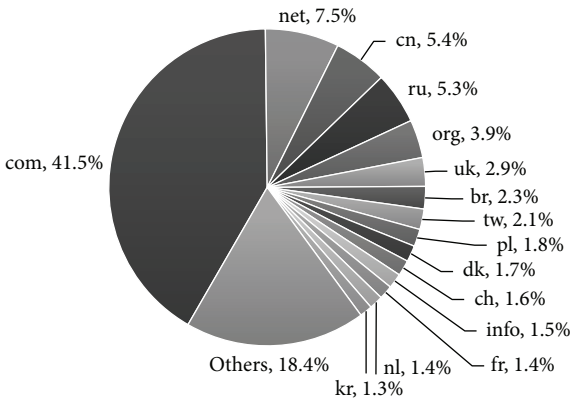



Figure 8: TLD distribution of malware domains.

Section 3.4.1. Moreover, we found that the number of malicious domain was evenly distributed to some ccTLDs. We believe this phenomenon is due to Domain-Flux of malware, where malware authors do not register all DGA domains but just expect that at least one of them is effective so that malicious software can connect to the C&C server. Sinkholes deployed by the Internet security agencies hijacked some DGA domains such as Conficker to a specific IP address, so they can collect statistics on malware infections. For example, the Conficker sinkhole 221.8.69.25 deployed by CNCERT/CC has been associated to 3,887 random domains under.cn, and the sinkhole addresses 149.20.56.34, 149.20.56.32, and 149.20.56.33 deployed by another organization has been linked to 58,216 domains in our DNS graph.

Fast-Flux botnets make special groups in PDG. The most distinguishing feature of them is that one or more domains connecting to a large number of hosts, and hosts nodes are largely overlapped for domain names controlled by the same FFSN (Fast-Flux Service Network). As an example, one connection component in PDG has 161 domains and 938 IP addresses; however, the number of the edges between these domains and IP addresses is as much as 46,039, which means that the average number of IP address associated with each domain is 286.

We choose another FFSN with fewer nodes for visualization [18]. The result is shown in Figure 9. Apparently the objective of the malware is to host phishing or advertising website for diet pills. The reputation of those nodes is all lower than $10^{-9}$. The upper part of Figure 9 shows the relationship between malicious domains and FFSN hosts, while the lower part shows NS (Name Servers) of these domains, which clearly shows that the FFSN is a Double-Flux botnet and has IP address overlapped among NS and A records.

## 5. Related Works

Several papers used graph mining techniques in network security researches. RiskRank [19] constructs graph model based on network communication among hosts, then assigns edge weight according to suspicious TCP/UDP port, and finally evaluates host risk with PageRank and HITS [20] algorithm. BotCloud [21] also constructs graph model based on IP Flow, mining P2P-botnet nodes with PageRank algorithm with prior knowledge of malicious hosts acquired with honeypots. Polonium [15] of Symantec Corp. firstly builds bipartite graphs of machine-files collected from Norton security product users and then detects malicious files with belief propagation algorithm. NetProbe [22] uses belief propagation algorithm to detect auction fraud on eBay. Our method in this paper is another exploration of graph mining for network security. We use belief propagation in a way similar to Polonium and NetProbe. However, our method is different in terms of graph construction and the application scenario. We focus on the protocol properties of DNS to dig out the potential exploit by malwares, which provides a new route to detect various types of malwares with protocol-independency.

Gao et al. [23] proposed an algorithm to detect malware domains with time correlation clustering. Its detection is based on initially known malicious domains, which is similar to our method in some ways. But our method is obviously superior where the malware domains detected by our approach do not need to have strong correlation with initially blacklisted domains. Both Notos [6] and EXPOSURE [7] focus on DNS-based malware domains detection with machine learning. They also need the initially known legitimate or malicious domains and hosts to calculate the reputation for unknown nodes. However, our method has two main differences. (1) Our method does not need to extract any domain properties but just constructs DNS graph with the relationship between host and domain. (2) Notos is based on passive DNS and EXPOSURE needs monitoring clients' queries; our method is more adaptable so it can be applied to both environments or data sources by means of PDG or DQRG. On the other hand, operation of Kopis [8] needs traffic data from large DNS service providers or TLD servers, which is a huge limitation for the common application.

## 6. Conclusions

In this paper, we presented the approach of mining malware domains, servers, and hosts based on DNS graph, which is scalable and effective for malware detection. We made the following contributions:

(1) We propose the detection algorithm-based DNS graph. With two type of DNS graph representing the relationship between domains and its resolving IP addresses, we infer the reputation of graph nodes

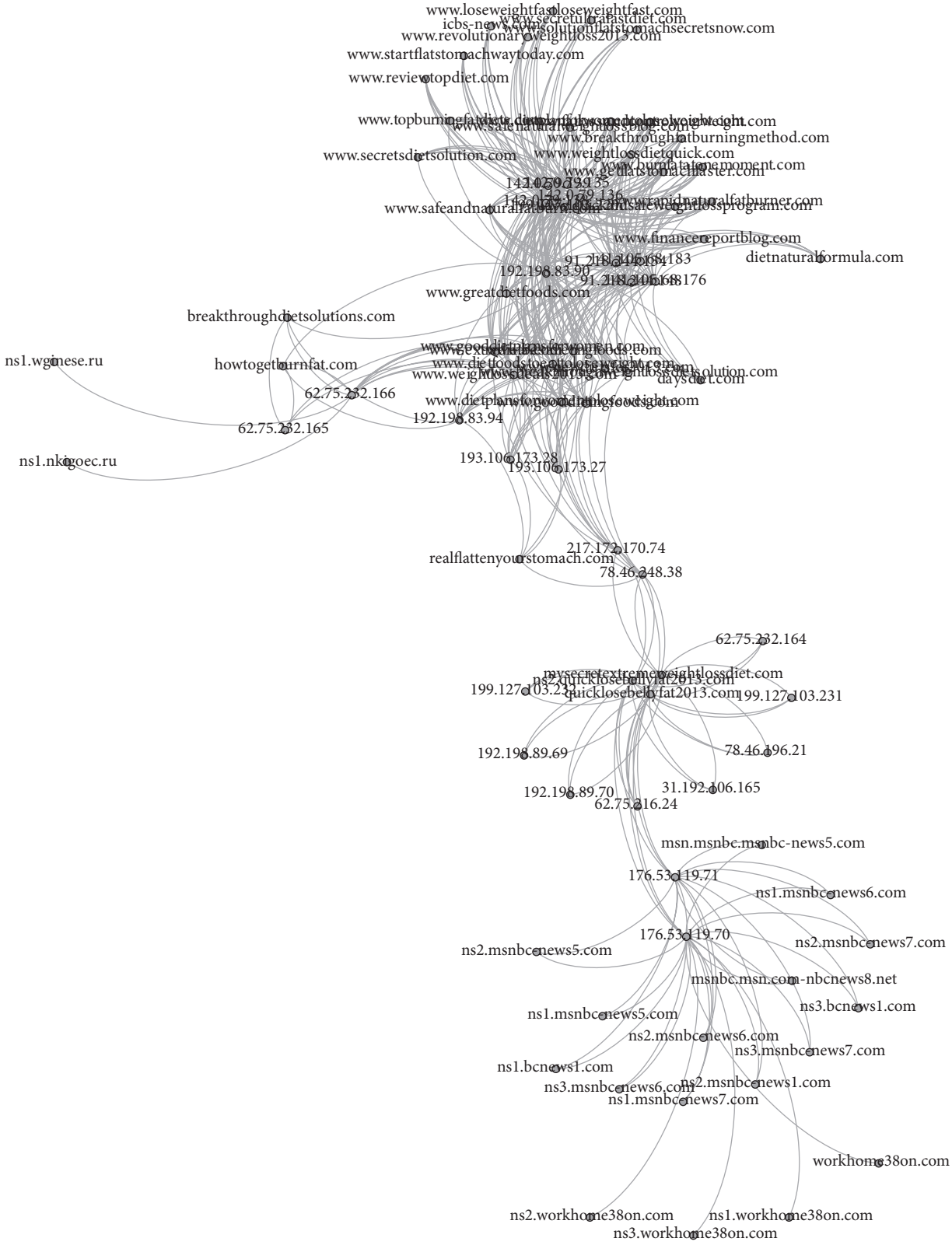



Figure 9: Visualization of a Fast-Flux service network.

using belief propagation. Our test on real-world traffic achieves a true positive rate of 80.63% with a false positive rate as low as 0.023%. With a false positive of 1.20%, the true positive rate improves to 95.66%. The result shows that our method is an efficient and effective way for malware detection in realistic application.

(2) Our method detects not only malware domains and C&C server but also the victim clients as well. It is also easy to do statistical analysis of malware infection in different aspects, which is more insightful. With the DQRG, victim client nodes can be clearly identified, while, with the PDG, the distribution of C&C servers and TLDs of malicious domains can be figured out.

(3) Our method does not tie to a specific C&C protocol and is independent of specific DNS technique. Thus, it has a feature of good generality for detecting various types of malware. Moreover, our method can effectively find the structural information of the malwares such as DGA-malware and FFSN-malware, all in a visualized way.

(4) We evaluated our algorithm performance on real-world data collected from campus DNS servers lasting three months. We constructed a large DNS graph consisting of 19,340,820 vertices and 24,277,564 edges and the calculation on the graph only takes a few minutes.

(5) We detailed the construction of DNS graph and its' features. Furthermore, we analyzed the reason of false positive and false negative and proposed the responding countermeasures.

In summary, our experiment and positive results with this work have demonstrated one promising solution to the problem of detecting malware. Currently, we just construct DNS graph with A and CNAME resource records. In the future, we will extend the record types to include NS, MX, and PTR and so forth, so as to enhance the relationship among nodes and assess the detection effect after these improvements.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work was supported by The National Key Basic Research Program (973 Program) of China (2013CB329603), National Natural Science Foundation of China (no. 61271220), and the Open Project of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security. Weixiong Rao's work is partially supported by Shanghai Science and Technology Commission (15ZR1443000), IBM Faculty Award 2014, and Open Lab of 360 Inc., China.

## References

[1] J. Nazario and T. Holz, "As the net churns: fast-flux botnet observations," in *Proceedings of the 3rd International Conference on Malicious and Unwanted Software (MALWARE '08)*, pp. 24–31, Alexandria, VA, USA, October 2008.

[2] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium*, San Diego, Calif, USA, February 2008.

[3] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*, pp. 48–61, ACM, Melbourne, Australia, November 2010.

[4] M. Antonakakis, R. Perdisci, Y. Nadji et al., "From throw-away traffic to bots detecting the rise of DGA-based Malware," in *Proceedings of the 21st USENIX Security Symposium*, p. 24, Bellevue, Wash, USA, August 2012.

[5] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. V. Steen, and N. Pohlmann, "On botnets that use DNS for command and control," in *Proceedings of the 7th European Conference on Computer Network Defense (EC2ND '11)*, pp. 9–16, Gothenburg, Sweden, September 2011.

[6] M. Antonakakis, R. Perdisci, D. Dagon et al., "Building a dynamic reputation system for DNS," in *Proceedings of the 19th USENIX Security Symposium*, pp. 273–290, Washington, DC, USA, August 2010.

[7] L. Bilge, E. Kirda, C. Kruegel et al., "Exposure: finding malicious domains using passive DNS analysis," in *Proceedings of the 18th Annual Network and Distributed System Security Symposium*, San Diego, Calif, USA, February 2011.

[8] M. Antonakakis, R. Perdisci, W. Lee et al., "Detecting malware domains at the upper DNS hierarchy," in *Proceedings of the 20th USENIX Security Symposium*, p. 27, San Francisco, Calif, USA, August 2011.

[9] F. Weimer, "Passive DNS replication," in *Proceedings of the 17th Annual FIRST Conference on Computer Security Incident Handling*, Singapore, June-July 2005.

[10] R. Kindermann and J. L. Snell, *Markov Random Fields and Their Applications*, American Mathematical Society, Providence, RI, USA, 1980.

[11] J. Pearl, "Reverend bayes on inference engines: a distributed hierarchical approach," in *Proceedings of the 2nd National Conference on Artificial Intelligence*, pp. 133–136, Pittsburgh, Pa, USA, August 1982.

[12] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," in *Exploring Artificial Intelligence in the New Millennium*, G. Lakemeyer and B. Nebel, Eds., pp. 239–269, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 2003.

[13] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. Hellerstein, "GraphLab: a new framework for parallel machine learning," in *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI '10)*, pp. 340–349, Catalina Island, Calif, USA, July 2010.

[14] Z. Ramzan, V. Seshadri, and C. Nachenberg, "Reputation-based security: an analysis of real world effectiveness [R/OL]," https://scm.symantec.com/resources/reputation_based_security.pdf.

[15] D. H. P. Chau, C. Nachenberg, J. Wilhelm et al., "Polonium: tera-scale graph mining and inference for malware detection,"

in *Proceedings of the SIAM International Conference on Data Mining*, pp. 131–142, Mesa, Ariz, USA, April 2011.

[16] T. Ehrenkranz and J. Li, "On the state of IP spoofing defense," *ACM Transactions on Internet Technology*, vol. 9, no. 2, article 6, 2009.

[17] MaxMind, GeoLite Free Downloadable Databases[CP/OL], http://dev.maxmind.com/geoip/legacy/geolite.

[18] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: an open source software for exploring and manipulating networks," in *Proceedings of the 3rd International AAAI Conference on Weblogs and Social Media*, San Jose, Calif, USA, May 2009.

[19] S. Wang, R. State, M. Ourdane, and T. Engel, "RiskRank: security risk ranking for IP flow records," in *Proceedings of the 6th International Conference on Network and Service Management (CNSM '10)*, pp. 56–63, IEEE, Niagara Falls, Canada, October 2010.

[20] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[21] J. François, S. Wang, W. Bronzi, R. State, and T. Engel, "BotCloud: detecting botnets using MapReduce," in *Proceedings of the IEEE International Workshop on Information Forensics and Security (WIFS '11)*, pp. 1–6, Iguacu Falls, Brazil, December 2011.

[22] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, "NetProbe: a fast and scalable system for fraud detection in online auction networks," in *Proceedings of the 16th International World Wide Web Conference (WWW '07)*, pp. 201–210, Banff, Canada, May 2007.

[23] H. Gao, V. Yegneswaran, Y. Chen et al., "An empirical reexamination of global DNS behavior," in *Proceedings of the ACM SIGCOMM conference on SIGCOMM*, pp. 267–278, Hong Kong, China, August 2013.