The Design and Implementation of a Covering MDN-Complete-Life-Cycle Malicious Domain Detection Framework

Li-feng REN¹,a, Zhao WANG²,b, Xin LIU³,c, Qing-shan LI⁴,d and Zhong CHEN⁵,e,*

Key Laboratory of High Confidence Software Technologies (MoE), School of EECS, Peking University, 100871, China

arlf@pku.edu.cn, bwangzhao@pku.edu.cn, ccarol.pku@gmail.com, dliqs@pku.edu.cn, ezhongchen@pku.edu.cn

*Corresponding author

Keywords: Malicious domain detection, Life cycle, Framework.

Abstract. Malicious domain name (MDN) detection has seen greatly progress in recent years. In this paper, one covering MDN-Complete-Life-Cycle malicious domain name detection framework is proposed. The framework includes three detection models: DGAD-M (Domain Generation Algorithm Detection Model), DIPD-M (Domain IP Detection Model) and DHTD-M (Domain Host Detection Model), corresponding to the process of the malicious domain generation, malicious domain name resolution and the host requesting a domain. DGAD-M bases on the fact that the domains generated by DGA are always short of natural language features, it adopts Convolutional Neural Network. DIPD-M bases on the fact that the IP addresses of the malicious domains are more disperse and updated frequently. DHTD-M bases on the fact that the domains requested by infected hosts are frequently tend to be malicious. The results of DGAD-M and DIPD-M will be used by DHTD-M. The framework got the accuracy rate of 83.652% with the real network flow and found out 115 suspicious malicious domains.

Introduction

Botnet has been an enormous threat to the Internet security. The latest data from CNERT/CC shows that more than one million hosts have been infected. And about 73% of the hosts are controlled by bots. The attackers can control their malwares to launch Distributed Denial of Service (DDoS), steal information, send spam mails, distribute malicious codes, and conduct other malicious network attacks by command and control (C&C) infrastructure.

C&C infrastructure is the core part of the whole Botnet which is used by the attackers to control the Bots. This infrastructure ensures the flexibility, robustness, availability of the Botnet. Recently, the attackers begin to use dynamic DGA to build up the connection between Bots and C&C server in the Botnet. Dynamic DGA always uses random variables as the seed to generate domain names which are shared by the malwares in the infected hosts and the C&C server. The attacker registers some of the generated domain names and bound them to the IP addresses of C&C server. The malware will generate the same group of domain names periodically and query the domain names one by one through the DNS server. If one or some of the domain names can be resolved to IP addresses, the malware gets the C&C server’s IP addresses by which they can build up a communication channel. With the channel, the attacker can
control the Bot. This DGA has highly improved the viability of the Botnet. With the shared DGA, the attackers can bind a new regenerated domain name to the server and the malwares can finally find the C&C server, no matter a failure or block of the domain name and IP address. Then the attackers regain the full authority. And the validity of domain names generated by DGA makes them difficult to be detected. Furthermore, one change of DGA makes the Reverse Engineering of the sample malware useless. Therefore, how to identify the malicious domains effectively accurately is the key to restrain and destroy the Botnet which uses DGA as the communication mechanism.

To ensure flexibility, robustness and availability, the domain name’s structure, the network distribution and behavior of the Botnet have three main characteristics:

- The malicious domain names are built up by random characters, which are short of natural language features.
- The IP addresses corresponding to the malicious domain names demonstrate some strong discretization features.
- The Bot periodically queries a large number of malicious domain names. The domain names that the Bot accesses have more probability to be malicious.

Those features make the detection of malicious domain names to be possible. In this paper, we put forward a covering MDN-Complete-Life-Cycle malicious domain name detection framework. It is formed by three malicious domain names detection models, DGAD-M, DIPD-M and DHTD-M, corresponding to the process of the malicious domain names generation, malicious domain names resolution and the host requesting a domain. The DGAD-M and DIPD-M detect the domain initially, whose results are used as the initial belief of the DHTD-M. The DHTD-M finishes the final judgment of the domains and IP addresses. The framework can achieve definite detection precision and has the possibility of detecting malicious domains in the real network environment.

Related Work

The recent related researches are falling into three aspects: the natural language features of domain names, the network discretization features, and the inference with the request relationship.

Manos Antonakakis [1] puts forward a way of detecting DGA using NXDomain and related data, combined with classification and clustering. The clustering algorithm clusters domains based on the similarity in the make-ups of domain names as well as the groups of machines which queried these domain names. The classification algorithm is used to assign the generated clusters to the patterns of known DGAs, and if a cluster cannot be assigned to a known pattern, then a new pattern is produced. Yuanchen He [2] finds out that legitimate domain names are always consist of English words or looked like meaningful English while many malicious domain names are randomly generated. They transform this intuitive observation into statistically informative features using second order Markov models. Four transition matrices are built from known legitimate domain names, known malicious domain names, English words in a dictionary, and based on a uniform distribution. In addition to using traditional quantitative features of domain names, Wang [4] uses a word segmentation algorithm to segment the domain names into individual words to expand the size of the feature set greatly. Then they fit a logistic regression model to classify the malicious domain names. Ma [5] designs a statistical method to classify malicious Web site URLs automatically based on their lexical and host-based properties. M Thomas [10] monitors the malicious domain names requested by the infected hosts, they use
clustering algorithm to classify the domain names generated by malicious code variants. They also find out that there is great difference in the amount of domain names requested by the infected ones and the uninfected ones.

Manos Antonakakis [5] finds out that the malicious domain names distribute more discretely. They use the dataset of domain name servers to acquire the network and domain features of the domain names in white and black lists. Combined with the characteristic data of malicious codes obtained from the honeypot and the sandbox, using the method of clustering, the domain names are gathered into different categories, then they use the Decision Tree to determine the possibility of a domain name to be a malicious one, and put forward the Notos model, which is the first prediction classifier using statistical features and machine learning algorithm to construct the confidence of the domain names. Chen [15] builds a malicious domain classifier from five aspects: diversity, time, growth, relatedness and IP information, consisting of 18 features, based on the Passive DNS dataset. Chen also brings up an idea called domain pool which describes the relationship among domains.

Manadhata P K [7] promotes a malicious domain name detection method based on the graph inference, they construct a host-domain graph from HTTP proxy logs, and set the graph with minimal ground truth information, and then use Belief Propagation to estimate if the marginal probability of a domain is malicious. Igor Mishsky [14] projects the famous expression “Tell me who your friends are and I will tell you who you are”, which motivates many social trust models, on the internet domains world. A domain which is related to malicious domains is more likely to be malicious as well. Zhang [8] promotes malicious domain name detection method based on DNS graph mining. He builds up the DNS request-response graph through the relationship between the IP address and domain name in the domain name server. He uses Belief Propagation algorithm to infer the reputation of every node in the graph, then recognizes the malicious ones. P Cameo [9] designs the CONDENSER, storing and retrieving the data through graph structure. CONDENSER can recognize the activities of Botnet and even detect the new Botnet. It can also cluster a child graph with the same communication pattern, and judge whether the child graph was a part of Botnet.

Leyla Bilge [12] designed the EXPOSURE, which extracts the time, DNS response, TTL and the domain name itself, 15 features in total, based on DNS flow dataset. EXPOSURE trains a classifier using those 4 aspects of features to recognize the malicious domain names. Manos Antonakakis [13] designed the Kpois, which detects malicious domains at the upper DNS hierarchy, using the network spatial distribution of the hosts which requests the domain names, the features of the hosts, and the reputation of the IP address corresponding to the domain name. Xu W [11] promotes a malicious domain names prediction system, based on the life cycle and reutilization of malware domain name, as well as DGA.

Researchers always focus on one aspect of the three parts, ignoring other features. Those methods based on DGA are trying to approximate the Domain Generation Algorithm, which rely on a large number of DGA malware samples. Those methods based on the corresponding relation between the domain name and IP address; highly rely on the DNS historical data, which can’t detect a domain name if there is short of historical data. Those methods based on access relationship between the host and the domains it requests highly depend on the white and black lists, which ignore malicious domain name itself and the network distribution. Based on those researches, we put forward a new malicious domain detection framework.
Covering MDN-Complete-Life-Cycle Malicious Domain Detection Framework Architecture

The framework covers the complete life cycle of the malicious domain name, consist of three detection models: DGAD-M, based on the malicious domain names generated by DGA, DIPD-M, based on the corresponding relationship between the domain name and IP address, DHTD-M, based on access relationship between the host and the domains it requests. These three detection models are corresponding to three main parts through the malicious domain name life cycle: the malicious domain generation, malicious domain name resolution and the process that host requests a domain name. The DGAD-M and DIPD-M respectively detect the domain names, and the results are used as initial belief, which are provided for the DHTD-M. Given a set of network flow, we can use the black and white lists, DIPD-M and DGAD-M to initially judge the malevolence of domain names. Then we put the initial belief and network flow into DHTD-M to predict the probability of being malicious domain name. Even the potential malicious domain names and the infected hosts can be inferred with this framework. The models of the framework will be introduced in the following paragraphs.

Figure1. Malicious Domain Detection Framework.

DGAD-M extracts the natural language information from the domain names, including the single domain name and the group domain names. The characteristic of the single domain is that the benign one is different from the malicious one in natural language. More specific, malicious domain names generated by DGA are always lack of natural language characteristics. While the benign domain names are approximating
the natural language and meaningful. The characteristics of the group domains are combined with the time series and combination features of malicious domain names.

DIPD-M abstracts the network discretization from the domain name and the information of its corresponding IP addresses. The former researches show that the life cycle of a malicious domain names is very short, frequently changed and the distribution of the IP addresses are more discrete. In DIPD-M, we define the dispersion, dispersion ratio and update rate of the corresponding IP addresses’ BGP prefix, ASN, country, register date, owner and local Internet registry, 18 features in total.

DHTD-M abstracts the access relationship between the host and the domain it requests. A host infected by malware always has a series of malicious behaviors, and the malicious domain names always get involved in the communication between the malware and attacker. The infected host more likely requests malicious domain names. Similarly, if a host requests a malicious domain name, it is more likely infected. DHTD-M regards the host IP addresses and the requested domain names as the nodes in the graph, and the access relationship is treated as the edge of the graph, which constitutes the bipartite graph. With the white and black lists, DGAD-M and DIPD-M to initialize belief, DHTD-M uses Belief Propagation to approximately compute the marginal probability of the nodes in the graph; the results are used as the final judgement of each IP address and domain name.

With those three models, we build up the covering MDN-Complete-Life-Cycle malicious domain detection framework. We use the corresponding dataset to train the models, and then use the real network flow to verify our framework and detect the unknown malicious domain names.

**DGAD-M**

DGAD-M uses Convolutional Neural Network, which has been widely used in the natural language processing [19] [20]. In DGAD-M, we build up two CNN models, one for the single domain name that focuses on the character structure features of the domain name and the other for the group domain names that focuses on the composition of a group of domain names in addition.
We use one-hot coding to encode every character of the domain names, with the scope of 37 characters including ‘a-z’, ‘0-9’ and ‘-’. After the one-hot coding, every character will be transformed into one-dimensional vector, with the length of 37. In this paper, we suppose that the second-level domain and the third-level domain aren’t longer than 16 characters.

**Single-Domain-Name CNN (S-CNN).** The S-CNN has 5 layers, with 2 convolutional layers and 3 fully connected layers. The input of S-CNN is a series of single domain names which have been encoded.

<table>
<thead>
<tr>
<th>Table 1. S-CNN convolutional layers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

Table 2. S-CNN fully connected layers.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Perceptron numbers</th>
<th>Dropout</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>256</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>256</td>
<td>0.15</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The length of convolutional kernel in the first layer is 2, which means convolutional operation on 2 characters. The length of convolutional kernel in the second layer is 4, corresponding to convolutional operation on 8 characters. The two convolutional layers aim to extract character structure features of single domain name, which will be learned by the fully connected layers.

**Group-Domain-Name CNN (G-CNN)**

Based on the former researches and our own experiments, we find out that the characters’ frequency distribution of malicious domain names is different from benign domain names. In addition to the character structure features of single domain name, we consider that learning on the groups of domain names may abstract the DGA better. We design the G-CNN.

The G-CNN has 6 layers, with 3 convolutional layers and 3 fully connected layers. The input of G-CNN is the groups of domain names which have been encoded.

<table>
<thead>
<tr>
<th>Table 3. G-CNN convolutional layers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Table 4. G-CNN fully connected layers.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Perceptron numbers</th>
<th>Dropout</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The first layer is same as S-CNN’s 1st layer. The kernel’s length of second layer is 16, corresponding to convolutional operation on 2 domain names and the third layer abstracts the composition of 6 domain names. The 2nd and 3rd layer aim to extract the features of the time sequence and compositional pattern. The 4-6 layers learn those features.

**DIPD-M**

In DIPD-M, we define the dispersion ($dd$), dispersion ratio ($ddr$) and update rate ($ur$) of a domain name and its corresponding IP addresses’ BGP prefix, ASN, country, register date, owner and local Internet registry, 18 features in total. The dispersion ($dd$) features include the amount of the BGP prefix, ASN, country; register date, owner and local Internet registry, which have been proved by the former researches. But we find out that there is a certain degree of coupling between malicious domain names and the benign ones. By analysis of the experiment data, we promote the dispersion ratio and the update rate to expand the former proved features. We define $F$, the set of BGP prefix, ASN, country, register date, owner and local Internet registry.

We define the dispersion ratio ($ddr$):

$$ddr = \frac{dd}{Ips}, \ i \in F$$

The $Ips$ is the amount of IP addresses corresponding to the domain name. The results show that the malicious domain names have larger $ddr$.

We define the update rate of one period, $ur_i$: $\phi$ is a non-negative function. $R_{t-1}$ represents feature $i$’s record set in the period $t$. The operator “$−$” extracts the different elements of the two sets.

$$ur_i = \phi\left(\left( R_{t-1} \cup R_i \right) - \left( R_{t-1} \cap R_i \right)\right), \ i \in F, \ t \in T$$

The $T$ is the period and one period means changing from $t-1$ to $t$. $R_i$ represents feature $i$’s record set in the period $t$. The function $\phi(x)$ represents the number of elements in the set. We define the update rate:

$$ur_t = \sum_{i=1}^n \frac{ur_i}{n}, \ t \in T$$

The results show that the malicious ones have larger update rate.

**DHTD-M**

DHTD-M abstracts the access relationship between the host and the domains it requests. With the white and black lists, DGAD-M and DIPD-M to initialize the belief, DHTD-M uses Belief Propagation to approximately compute the marginal probability of the nodes in the graph and the results are used as the final judgement of each IP address and domain name.

**The Ground Truth of DHTD-M.** We first use the black and white lists, then the DIPD-M and DGAD-M. The priority is based on that the black and white lists can be
more precise to classify the domain names, while the DIPD-M is based on the statistical characteristics of the network discretization of the domain name and the its corresponding IP addresses’ information, whose precision depends on the historical records. And we train the DGAD-M with the malicious domain names samples from Conficker.B, which can classify the specified domain names. The Table.5 shows the ground belief of DHTD-M.

Table 5. The Ground Belief of DHTD-M.

<table>
<thead>
<tr>
<th>Judgement</th>
<th>Malicious Belief</th>
<th>Benign Belief</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black List</td>
<td>0.99</td>
<td>0.01</td>
</tr>
<tr>
<td>White List</td>
<td>0.01</td>
<td>0.99</td>
</tr>
<tr>
<td>Malicious by DIPD-M</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>Benign by DIPD-M</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>Malicious by DGAD-M</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>Benign by DGAD-M</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>Unknown</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>IP</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

The Potential Matrix of DHTD-M. In the DHTD-M, we define the potential matrixes to reflect the dependency of two adjacent nodes, supposing the probability that the adjacent nodes have the similar attributes is larger, which means that the adjacent nodes are more likely both malicious and benign compared with one malicious and the other benign. In the Table.6, we define the potential matrix. With the potential matrix, we define four potential energy of propagation, as shown in Table.7.

Table 6. The Potential Matrix of DHTD-M.

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$x_j$</th>
<th>Malicious($M$)</th>
<th>Benign($B$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malicious($M$)</td>
<td>0.51</td>
<td>0.49</td>
<td></td>
</tr>
<tr>
<td>Benign($B$)</td>
<td>0.49</td>
<td>0.51</td>
<td></td>
</tr>
</tbody>
</table>

Table 7. The Definition of Four Potential Energy.

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MM$</td>
<td>As node $i$ is $M$, the probability of adjacent node to be $M$ is 0.51</td>
</tr>
<tr>
<td>$BM$</td>
<td>As node $i$ is $B$, the probability of adjacent node to be $M$ is 0.49</td>
</tr>
<tr>
<td>$MB$</td>
<td>As node $i$ is $M$, the probability of adjacent node to be $B$ is 0.49</td>
</tr>
<tr>
<td>$BB$</td>
<td>As node $i$ is $B$, the probability of adjacent node to be $B$ is 0.51</td>
</tr>
</tbody>
</table>

Analysis of the Non-Record Domain Names. The malware using domain name as the mechanism to find the C&C server may request a lot of domain names that never exist. In the DHTD-M, we design a way to handle those nonexistent domain names, which is based on the request results of several third-part domain names resolution services. The less recorded, the more malicious the domain name is. Supposing the amount of services is $n$ and $n > 1$, if a domain name has no record among all the services, we define its possibility to be malicious is:
If a domain name is recorded in \( m \) services, we define the possibility to be malicious is:

\[
\frac{1}{n + a\sqrt{n}}
\]

Define the malicious threshold as \( T \), \( T > 0 \). When the constant \( a \) is permanent, it needs \( m \) of \( n \) to reach the \( T \), and the rate of no record is \( r \). For the larger service numbers \( n_2, n_2 > n_1 \), to reach the same \( T \), it needs the larger no record rate \( r_2, r_2 > r_1 \).

And if the constant \( a \) is larger, it need a much larger no record rate \( r_3, r_3 > r_2 \), to reach the same malicious threshold \( T \).

**Implementation and Experiments Analysis**

**DGAD-M**

**Implementation.** We collect the malicious domain names generated by the Conficker.B as samples for the DGAD-M. In the experiment, we find out that the Conficker.B will request 250 malicious domain names after running and it will request another 250 every 3 hours after the first request behavior. The periodical behavior is monitored by the IOGraph of Wireshark, as shown in the Figure.

![Figure 3. The Periodically Malicious Domain Names Request Behavior of Conficker.B.](image)

We find out that all the malicious domain names generated by Conficker.B are constructed by TLD (Top-level domain) and SLD (Second-level domain), which are formed with 5-11 characters. The frequency of all 26 characters is similar.

We use the Keras to build up the DGAD-M, which is a high-level neural networks API, written in Python and capable of running on top of either TensorFlow or Theano. We use the generated domain names as malicious ones and the Alexa’s top ranking domain names as benign ones. For the S-CNN, each single domain name will be encoded into a 37*16 vector. For the G-CNN, every 10 domain names will be encoded into a 37*16*10 vector.
**Experiment Analysis.** We use 6000 domain names to train the S-CNN and 1500 for testing. The result of train set stabilises at around the accuracy rate of 99.8%. Then we use the trained S-CNN to detect the test dataset and get the accuracy rate of 93.2%.

Different from the S-CNN, we group the train and test dataset with every 10 domain names. The result of train set stabilises at around the accuracy rate 99.17%. Then we use the trained G-CNN to detect the test dataset and get the accuracy rate of 98.67%.

Our experiments show that the DGAD-M can effectively classify the malicious domain names generated by Conficker.B, which is inspired by the natural language processing.

![Table 8. The Results of DGAD-M.](image)

<table>
<thead>
<tr>
<th></th>
<th>S-CNN</th>
<th>G-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train Set</td>
<td>Test Set</td>
</tr>
<tr>
<td>Convergence Period</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Accuracy Rate</td>
<td>99.82%</td>
<td>93.2%</td>
</tr>
</tbody>
</table>

**DIPD-M**

**Implementation.** To train the DIPD-M, we collect 20690 malicious domain names from DNS-BH. We get corresponding 20690 benign domain names from the Alexa top ranking. And the historical DNS datasets are from NDSCensus2013 [26] and Rapid7 [27], from which we resolve the IP addresses corresponding to the domain names and compute the dispersion, dispersion ratio and update rate of the BGP prefix, ASN, country, register date, owner and local Internet registry. We use SVM to build up the DIPD-M with the scikit-learn, which is a free software machine learning library designed by the Python programming language.

**Experiment Analysis**

The results show that the DIPD-M get the accuracy rate of 93.58%, with the malicious domain names’ false negative rate of 15%. Among the three types of features, the update rate is the most precise, reaching 91.81% and the lowest benign domain names false negative rate is 5.3%. The dispersion feature has the lowest malicious domain names false negative rate. But the dispersion ratio does not reach our expectation.

![Table 9. The Results of DIPD-M.](image)

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Accuracy Rate</th>
<th>False Negative Rate (Malicious Domains)</th>
<th>False Negative Rate (Benign Domains)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispersion</td>
<td>0.87719</td>
<td>0.05</td>
<td>0.13245</td>
</tr>
<tr>
<td>Dispersion Ratio</td>
<td>0.85965</td>
<td>0.75</td>
<td>0.0596</td>
</tr>
<tr>
<td>Update Rate</td>
<td><strong>0.91813</strong></td>
<td>0.3</td>
<td><strong>0.0530</strong></td>
</tr>
<tr>
<td>All 3 features</td>
<td>0.93567</td>
<td>0.15</td>
<td>0.0530</td>
</tr>
</tbody>
</table>
DHTD-M

**Implementation.** We collect a set of real network flow, including 14780619 domain name access records. After the processing of the dataset, we get 74408 nodes which are the host IP addresses and domain names, 337677 edges which are the access relationship between the hosts and domain names. The white list is also from Alexa top ranking domain names and the black list is form DNS-BH and other third part organizations.

From the white and black lists, 2295 of 36829 domain names are matched. The initial belief will be assigned to (0.99, 0.01) if the domain name is in the black list. The initial belief will be assigned to (0.01, 0.99) if the domain name is in the white list. Then we randomly choose 1/4 of black-list matched domain names and 1/4 of white-list matched domain names and set them to be unknown. The domain names that are not matched and the unknown domain names will be input into DIPD-M. The DIPD-M finds out 249 domain names. The initial belief will be assigned to (0.9, 0.1) if the domain name is considered to be malicious by the DIPD-M. The initial belief will be assigned to (0.1, 0.9) if the domain name is considered to be benign by the DIPD-M. The domain names that are still not matched will be input to DGAD-M. In the experiment, we choose the S-CNN for that we have to detect the domain name one by one. The DGAD-M finds out 5789 domain names. The initial belief will be assigned to (0.8, 0.2) if the domain name is considered to be malicious by the DGAD-M. The initial belief will be assigned to (0.2, 0.8) if the domain name is considered to be benign by the DGAD-M.

We use LibDAI to build up the DHTD-M, which is an open source C++ library that provides implementations of various (approximate) inference methods for discrete graphical models.

**Experiment Analysis.** We define the malicious belief threshold to be 0.8, which means that a domain name is malicious if its malicious belief is equal or greater than 0.8. We define the benign belief threshold value to be 0.8, which means that a domain name is benign if its benign belief is equal or greater than 0.8. The DHTD-M gets the detection accuracy rate of 83.65%.

Compared to the DGAD-M and DIPD-M, DHTD-M doesn’t get the similar accuracy rate. Because the DGAD-M and DIPD-M mostly focus on the domain names, their internal structure and corresponding information. But DHTD-M maps the relationship that the host requests the domain names to a bipartite graph, making the domain names indirectly interact, so as the IP addresses. Every node is based on all the other nodes in the subgraph. The detection precision depends on the whole structure of the graph. But based on this interaction, DHTD-M has the ability to infer the unknown malicious domain names, infected host, even the infected subgraphs.

The DHTD-M finds out 122 malicious nodes, including 5 IP addresses and 117 domain names. For the 117 domain names, we choose 4 third part services to lookup them. Only get 2 records with virustolal.com and the other 115 domain names have no records.
Table 10. The Results of Non-Record Domain Names.

<table>
<thead>
<tr>
<th>Lookup Services</th>
<th>Total</th>
<th>No record</th>
<th>Recorded</th>
</tr>
</thead>
<tbody>
<tr>
<td>virustolal.com</td>
<td>117</td>
<td>115</td>
<td>2</td>
</tr>
<tr>
<td>Threadcrowd.org</td>
<td>117</td>
<td>117</td>
<td>0</td>
</tr>
<tr>
<td>socket.gethostbyname</td>
<td>117</td>
<td>117</td>
<td>0</td>
</tr>
<tr>
<td>python-whois</td>
<td>117</td>
<td>117</td>
<td>0</td>
</tr>
</tbody>
</table>

We find out that those suspicious ones are formed with 5-11 random characters. According to the way we promote to analyze the non-record domain names, $a = 0.5$, 2 of 117 suspicious domain names get 0.6 risk factor and 115 of 117 suspicious domain names get 0.8 risk factor.

Summary

Malicious domains have been widely used by malware to build up Command and Control, which improve the flexibility, robustness and availability of the C&C network. This increases the recognition, detection and clearance difficulty of malwares. It also fortifies the security threat of the network. Recently, researchers of malicious domains detection mainly focus on three parts of the life cycle of malicious domain.

- Compositional pattern. Malicious domains that generate by DGA, are formed by random characters and short of natural language features.
- The corresponding relation between domain name and IP address. The IP addresses mapping to the malicious domains are more disperse and frequently updated.
- The access relationship between the host and the domains it requests. A host which is infected by malware is more likely to request malicious domains. And if a host which has accessed a malicious domain, is more likely to be infected.

Researchers always focus on one point of the three parts, and ignore other features. According to the life cycle of a malicious domain, we put forward a covering MDN-Complete-Life-Cycle malicious domain detection framework. It is formed by DGAD-M, DIPD-M and DHTD-M, corresponding to the three parts of the life cycle. The DGAD-M and DIPD-M detect the domain initially, whose results are used as the initial belief of the DHTD-M. The DHTD-M finishes the final judgement of the domains and IP address.

DGAD-M focuses on the compositional pattern of domains generated by DGA, which is inspired by natural language processing. DGAD-M uses the single domain and group domains to train the Convolutional Neural Network. The single-domain CNN gets a precision of 93.2% and the group-domains CNN gets a precision of 98.67%. DIPD-M focuses on the corresponding relationship between domain name and IP address. Totally 18 features such as the corresponding information’s dispersion, dispersion ratio and update rate are used to train Support Vector Machine. DIPD-M gets a precision of 93.567%. DHTD-M focuses on the access relationship between the host and the domains it requests. The domain names and IP addresses are the nodes in the bipartite graph, with the access relationship as the edges. DHTD-M uses Belief Propagation to compute the belief of the domain names and IP addresses. It gets the detection accuracy rate of 83.652% with the real network flow and find out 115 suspicious malicious domains.
In this paper, we put forward a malicious domain detection framework, which can achieve definite detection precision during the process of the malicious domain generation, malicious domain resolution and host requesting a domain. The framework has the possibility of detection malicious domains in the real network environment.

Acknowledgement
This work was sponsored by the National Science Foundation of China under grant No. 61371131.

References


[27] Rapid7 Reverse DNS, https://scans.io/study/sonar.rdns


[34] https://en.wikipedia.org/wiki/Convolutional_neural_network


