An Approach for Identifying Malicious Domain Names Generated by Dictionary-Based DGA Bots

doi: https://doi.org/10.1587/transinf.2020NTL0001

LETTER *Special Section on the Architectures, Protocols, and Applications for the Future Internet*

An Approach for Identifying Malicious Domain Names Generated by Dictionary-Based DGA Bots

Akihiro SATOH[†a)], **Yutaka NAKAMURA**[†], **Yutaka FUKUDA**[†], **Daiki NOBAYASHI**[†], ***and*** **Takeshi IKENAGA**[†], ***Members***

SUMMARY Computer networks are facing serious threats from the emergence of sophisticated new DGA bots. These DGA bots have their own dictionary, from which they concatenate words to dynamically generate domain names that are difficult to distinguish from human-generated domain names. In this letter, we propose an approach for identifying the callback communications of DGA bots based on relations among the words that constitute the character string of each domain name. Our evaluation indicates high performance, with a recall of 0.9977 and a precision of 0.9869.

key words: *dga bot, dictionary-based domain generation algorithm, domain name, network security*

1. Introduction

Some of the most serious security threats facing computer networks involve botnets. A botnet is a group of compromised machines, termed bots, that can be remotely controlled through a command-and-control server (C&C). Cybercriminals operate these bots through a C&C to perform malicious activities, such as phishing an organization, spreading bots to other machines, and stealing confidential information.

Although administrators need to swiftly remove bots that reside in their networks, many botnets have domain generation algorithms (DGAs) to avoid detection [1]. A DGA is a mechanism that frequently changes the domain name for the C&C to hide the callback communication from the bot to the C&C. Specifically, the bot uses the DGA to dynamically generate domain names and attempts name resolution for each; the domain name that returns the correct response is taken to be the C&C.

Some previous studies [2]–[4] focused on discernible differences in the character strings of benign and malicious domain names for detecting the callbacks of DGA bots. For example, Truong et al. [3] proposed a method that learns and predicts the character patterns in domain names using bigram models with supervised learning. Anderson et al. [4] extended this method to include character-level modeling with long short-term memory (LSTM) networks. These methods are based on observations: benign domain names tend to reflect the intent of their domain registrants, such as an organization, product, or content; in contrast, malicious domain names are meaningless character strings because conflicts with already registered domain names must be avoided. These efforts have spurred the emergence of more sophisticated DGA bots to defeat these detection methods. These DGA bots have their own dictionary, from which they can concatenate words to dynamically generate domain names that are difficult to distinguish from human-generated domain names [5]. Thus, character-level modeling, which excludes words, is unlikely to ensure sufficient accuracy for detecting these dictionary-based DGA (dict-DGA) bots.

Pereira et al. [6] proposed a method for detecting dict-DGA bots based solely on domain strings. This method has two functions: estimating the dictionary used by dict-DGA bots from their callbacks and identifying malicious domain names using the estimated dictionary. However, this identification is extremely simple: malicious domain names are determined by whether the number of words constituting the character string exceeds a given threshold level in the estimated dictionary. As a result, many false positives are caused by the lack of consideration of benign domain names. Our work focuses on characterizing the differences between benign and malicious domain names by machine learning, and the results improve upon those achieve by the method in [6].

2. Proposal

In this letter, we attempt to detect dict-DGA bots by identifying malicious domain names from massive queries for domain name systems (DNSs). We focus on queried domain names for the DNSs because name resolution is an unencrypted interaction that always occurs prior to the callbacks of DGA bots. Table 1 shows examples of domain names generated by dict-DGA bots. Since there are differences between dynamically generated domain names and human-generated domain names in terms of the frequency of word

Table 1 Examples of domain names generated by dict-DGA bots.

Banjori	`earnestnessbiophysicalohax[.]com`
	`pbmnestnessbiophysicalohax[.]com`
Pizd	`actionwelcome[.]net`
	`brokenforget[.]net`
Rovnix	`toourgovernmentscorrespondence[.]com`
	`ofhistoryandwithoutindependent[.]com`
Suppobox	`windowtherefore[.]net`
	`severadifference[.]net`

Manuscript received August 14, 2020.
Manuscript revised December 13, 2020.
Manuscript publicized February 17, 2021.

[†]The authors are with Kyushu Institute of Technology, Kitakyushu-shi, 804–8550 Japan.
a) E-mail: satoh@isc.kyutech.ac.jp

DOI: 10.1587/transinf.2020NTL0001

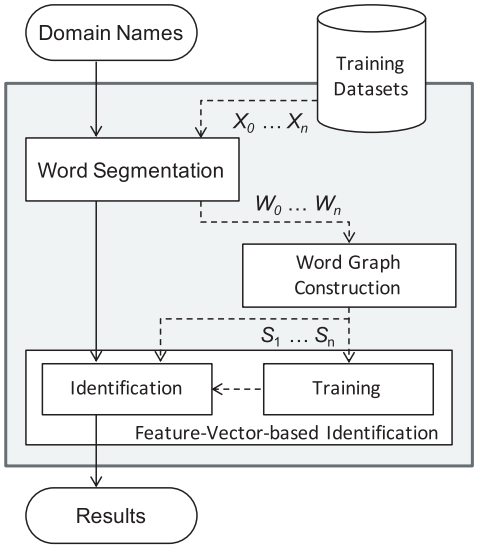

Fig. 1 Overview of the proposed approach for detecting dict-DGA bots based on word relations in domain names.

co-occurrences, we propose an approach for identifying malicious domain names by analyzing relations among the words constituting character strings in each domain name. The novelties of this approach include the following: (1) the use of general graph-theoretical techniques to represent relations among words in domain names; and (2) the use of centrality to quantify the importance of each word in the domain name. These indicators characterize the differences between benign and malicious domain names. Finally, we apply machine learning to the feature vectors derived from these indicators to reduce the number of false positives.

In [7], the domains for classification as benign or malicious were significantly narrowed by focusing on nonexistent domain (NXDOMAIN) responses that occur as the DGA changes the callback destination. In this case, the NXDOMAIN response is an error message that indicates an invalid domain name. Based on this insight, we consider only domains for which the name resolution yielded NXDOMAIN responses.

Figure 1 shows an overview of the proposed approach, which has three steps: (1) word segmentation, (2) word graph construction, and (3) feature-vector-based identification. The following sections describe the training datasets and each of these steps in detail.

2.1 Training Datasets

We denote a dataset comprising the same type of domains by $x \in X_i, i \in 0 \cdots n$, where the dataset X_0 contains benign domains and the remaining n datasets $X_1, \cdots, X_n$ contain malicious domains generated by n different DGA bots. Note that domain names in the datasets are shortened and replaced with primary domain names. A primary domain is the highest-level domain name given to a registrar. For example, the primary domain names for `www.ieice.org` and `ns.kyutech.ac.jp` would be `ieice.org` and `kyutech.ac.jp`, respectively.

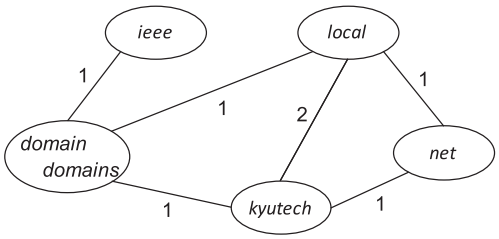

Fig. 2 Example of constructing a word graph using a set of word groups.

2.2 Word Segmentation

This step segments the primary-level character string for domain x into a word group w using a dictionary $\mathbb{D}$. Dictionary $\mathbb{D}$ includes an English dictionary and a corpus collected through web crawling. Word segmentation is based on two conditions: (1) minimize the number of words while maximizing the length of words, and (2) give preference to words that are in the dictionary by increasing their likelihood of being selected. We define this word segmentation approach with the following equation:

$$\mathcal{F}(x) = \arg\max_{w \in \mathbb{W}(x)} \frac{1}{m} \prod_{j=1}^{m} \mathcal{P}(w_j)$$

$$\mathcal{P}(w_j) = \begin{cases} 1 & (w_j \in \mathbb{D}) \\ 1/|\mathbb{D}|^{|w_j|} & (w_j \notin \mathbb{D}) \end{cases}$$

Here, $\mathbb{W}(x)$ is all the candidate segmentations for the primary-level character string of domain x; w is the candidate segmentation comprising words $w_1, \cdots, w_j, \cdots, w_n$; $|w_j|$ is the length of word w_j; and $|\mathbb{D}|$ is the total number of words in dictionary $\mathbb{D}$. Furthermore, $\mathcal{P}(w_j)$ is the selectivity of word w_j, which is based on whether word w_j is in dictionary $\mathbb{D}$. $\mathcal{P}(w_j) = 1$ when $w_j \in \mathbb{D}$, and $\mathcal{P}(w_j) = 1/|\mathbb{D}|^{|w_j|}$ when $w_j \notin \mathbb{D}$. As an example of word segmentation, a primary-level character string `kyutechlocaldomain` is divided into word group {`kyutech`, `local`, `domain`}. When this step of the algorithm is complete, it outputs a set of word groups W_i derived by segmenting the primary-level character strings for all the domains in training dataset X_i. Note that set W_i allows us to contain duplicate elements.

2.3 Word Graph Construction

This step uses a word graph to represent the relations among a set of word groups W_i corresponding to domain strings in training dataset X_i. A word graph G_i is simply a weighted undirected graph with a vertex set and an edge set. Each vertex corresponds to each word in set W_i; the edge weights indicate the frequencies of co-occurrence of words in set W_i, where co-occurrence means that the words belong to the same word group. Note that words with similar character strings share the same vertex. We use the Levenshtein ratio for similarity calculations and unweighted centroid clustering with threshold th_α for word aggregation. Figure 2 shows an example of a constructed word graph using the set of three-word groups [{`kyutech`, `local`, `domain`}, {`kyutech`,

Table 2 Numbers of benign and malicious domains in the datasets.

*D*0	*D*1	*D*2	*D*3	*D*4	*D*5	*D*6	*D*7
Benign	Banjori	Gozi	Matsnu	Pizd	Rovnix	Sisron	Suppobox
3021124	30000	30000	30000	30000	30000	30000	30000

`local`, `net`}, {`ieee`, `domains` }]. Due to the similarity of `domain` and `domains`, the two words share a vertex. Additionally, we define conjunction as a process for two-word graphs. Conjunction is the process of reconstructing word graph $G_{i:j}$ from word group sets W_i and W_j into word graphs G_i and G_j.

Dynamically generated malicious domain names and human-generated benign domain names exhibit clear differences in their frequencies of word co-occurrences. The words that illuminate these distinctions most effectively are those that play central roles in a word graph. To this end, we associate the following importance value with an arbitrary group w comprising words $w_1, \cdots, w_j, \cdots, w_n$:

$$\mathcal{S}_i(w) = \sum_{w_j \in w} |w_j|(C_{0:i}(w_j) - C_0(w_j))$$

Here, $|w_j|$ denotes the length of word w_j. $C_0(w_j)$ and $C_{0:i}(w_j)$ are functions that derive the centrality of word w_j in word graphs G_0 and $G_{0:i}$. Accordingly, this value means that the change in the centrality of word group w resulting from the conjunction of word graph G_0 is constructed from benign dataset X_0 and G_i is constructed from malicious dataset X_i. To compute the values, we use the PageRank centrality, which can be used even if a word graph is undirected and disconnected.

2.4 Feature-Vector-Based Identification

This step first calculates feature vectors from the domains in the training datasets. Based on the importance values in the word graph defined in the previous section, the feature vector for the domain name consisting of word group w is given by the following equation:

$$\vec{w} = (\mathcal{S}_1(w), \cdots \mathcal{S}_i(w), \cdots \mathcal{S}_n(w))$$

Then, we apply machine learning to these feature vectors to construct a training model. We adopt a support vector machine (SVM) as the machine learning algorithm because of its generalization and identification performance. The output of this step is the overall identification results of our algorithm regarding the benign versus malicious nature of the unknown input domain names, which are obtained by segmenting the domain name, calculating the feature vector, and applying the training model.

3. Evaluation

This section presents the experiments conducted to evaluate the effectiveness of the proposed approach. The main focus of the evaluation is the accuracy of identifying benign and malicious domain names. The experimental setup and results are described in Sects. 3.1 and 3.2.

Table 3 Experimental results.

	Recall	Precision
Anderson et al. [4]	0.9977	0.9305
Pereira et al. [6]	0.8873	0.6380
Our work	0.9977	0.9869

3.1 Experimental Setup

Table 2 shows the datasets of the benign and malicious domains used in the experiments. The malicious domains of seven DGAs were published on the websites [8], [9], whereas the benign domains resulting in NXDOMAIN responses were collected via our campus network. We confirmed the absence of bot-related domains in the benign set as follows: suspicious machines were detected by multiple security appliances, and these machines were examined for matches with the characteristic reported in [1], [5], [7] that a DGA bot causes massive NXDOMAIN responses as a result of name resolution for unfamiliar domains during a short period. In the experiments, 5-fold cross-validation of the benign and malicious domains was performed using 20% of the data as the testing datasets and the remainder as the training datasets.

For comparison with our approach, we implemented two different methods for identifying benign and malicious domains based only on their character strings, as described in [4] and [6]. The first implementation uses character-level modeling with LSTM networks applied to the domain strings, whereas the second implementation uses the number of words that constitute the domain string and that are present in the estimated dictionary as the identification threshold.

In the proposed approach, we mainly used words registered in Aspell [10] and the corpus [11] as the dictionary for word segmentation. The total number of words in the dictionary was 500,000. We set the threshold th_α to 0.85 and the SVM kernel to a radial basis function with hyperparameter 1.0 and cost 5.0. These parameters were determined experimentally, and their optimization will be addressed in a future work.

3.2 Experimental Results and Discussion

We employ two common metrics to characterize the identification ability for benign and malicious domains. Recall is the ratio of the number of correctly predicted malicious domains to the total number of actual malicious domains, and precision is the ratio of the number of correctly predicted malicious domains to the total number of predicted malicious domains.

The experimental results are presented in Table 3,

Table 4 Numbers of misidentified domains in the datasets.

	*D*0	*D*1	*D*2	*D*3	*D*4	*D*5	*D*6	*D*7
Anderson et al. [4]	15631	0	154	0	62	27	0	220
Pereira et al. [6]	105706	0	285	26	23286	28	0	36
Our work	2772	0	295	31	29	32	0	82

where each value is the mean of the 5-fold cross-validation results. The proposed approach achieves a recall of 0.9977 and a precision of 0.9869. These results indicate that our approach provides higher accuracy than that of the two previous implementations considering both recall and precision. Possible explanations for the performance deterioration in the two implementations include the following. First, identifying benign and malicious domains based on only character patterns has natural limitations. In the implementation based on [4], the tendency to identify domain names consisting of 8 or more letters alone as malicious was particularly prominent, and the results caused more than 15,000 false positives for benign domains. Next, in the implementation based on [6], uniformly quantifying the importance of words in the domain strings resulted in a large number of misidentified domains. Specifically, the cases of benign and Pizd domains were significant, exceeding 100,000 and 23,000 respectively. Although the misidentification for Pizd may be improved by optimizing the threshold level, false positives for benign domains will also increase. We attribute the success of the proposed approach to navigate these pitfalls to the consideration of differences between words constituting benign and malicious domain names and to the focus on words playing central roles in word graphs.

The numbers of misidentified domains are presented in Table 4, where each value is the sum of the 5-fold cross-validation results. Our approach achieves a extremely strong identification performance for Banjori and Sisron, whereas the other datasets, including the benign domain names, have some misidentifications. Most of these misidentified domain names can be classified into 4 categories: (a) domain names consisting of a single word; (b) domain names consisting of words that appear with extremely low frequency in the training datasets; (c) domain names containing some words common to benign and malicious domains; and (d) domain names comprising words not included in the dictionary. The proposed approach focuses on the relations among words in domain names and leverages the supervised machine learning to identify benign and malicious domains. The properties of this approach naturally make accurate identification difficult in cases (a) and (b). An investigation into the character strings arising in case (c) revealed many occurrences of words such as `domain`, `network`, `host`, and `local`, which are common to both benign and malicious domains. It may be possible to improve the performance in this area by excluding words that commonly appear in domain names, following the technique of stop words in natural language processing. The poor performance in case (d) arose from domain names being segmented into meaningless short character strings. Examples include domain names comprising random character strings, nonalphabetical strings represented alphabetically, or proper nouns. Thus, the use of more comprehensive dictionaries will alleviate the issue observed in case (d).

Despite the above problems, we can confirm that the proposed approach achieves high performance, with a recall of 0.9977 and a precision of 0.9869. The results suggest that dict-DGA bots can be removed from a network by using name resolution for malicious domains as triggers.

4. Conclusions

In this letter, we aimed to detect dict-DGA bots by identifying malicious domain names from massive DNS queries. Our experiments demonstrated that the proposed approach is capable of detecting callbacks from DGA bots with high accuracy. By enabling the ability to swiftly address various bots in networks, the approach contributes to dramatically improving network security. In the future, we plan to evaluate the identification accuracy and computational time of the approach for DNS queries observed over large-scale networks.

References

[1] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," Proc. USENIX Conference on Security Symposium, pp.263–278, Aug. 2016.
[2] A. Satoh, Y. Nakamura, D. Nobayashi, and T. Ikenaga, "Estimating the randomness of domain names for DGA bot callbacks," IEEE Commun. Lett., vol.22, no.7, pp.1378–1381, July 2018.
[3] D. Truong and G. Cheng, "Detecting domain-flux botnet based on DNS traffic features in managed network," Security and Communication Networks, vol.9, no.14, pp.2338–2347, Sept. 2016.
[4] H.S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-tuned domain generation and detection," Proc. ACM Workshop on Artificial Intelligence and Security, pp.13–21, Oct. 2016.
[5] A.K. Sood and S. Zeadally, "A taxonomy of domain-generation algorithms," IEEE Security & Privacy, vol.14, no.4, pp.46–53, July-Aug. 2016.
[6] M. Pereira, S. Coleman, B. Yu, M. DeCock, and A. Nascimento, "Dictionary extraction and detection of algorithmically generated domain names in passive DNS traffic," Proc. International Symposium on Research in Attacks, Intrusions, and Defenses, pp.295–314, 2018.
[7] 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, Jan. 2017.
[8] Fraunhofer FKIE, "DGArchive," https://dgarchive.caad.fkie.fraunhofer.de.
[9] J. Bader, "Some results of my DGA reversing efforts," https://github.com/baderj/domain_generation_algorithms.
[10] K. Atkinson, "GNU Aspell," http://aspell.net.
[11] P. Norvig, "Natural language corpus data: Beautiful data," http://norvig.com/ngrams/.