A Machine Learning Framework for Studying Domain Generation Algorithm (DGA)-Based Malware: 14th International Conference, SecureComm 2018, Singapore, Singapore, August 8-10, 2018,...

Chapter · August 2018

DOI: 10.1007/978-3-030-01701-9_24

CITATIONS
12

READS
2,328

4 authors, including:

Kaiqi Xiong
University of South Florida
148 PUBLICATIONS **3,892** CITATIONS
SEE PROFILE

Chengbin hu
University of South Florida
7 PUBLICATIONS **290** CITATIONS
SEE PROFILE

A Machine Learning Framework for Studying Domain Generation Algorithm (DGA)-Based Malware

Tommy Chin[1(✉)], Kaiqi Xiong[2], Chengbin Hu[2], and Yi Li[2]

[1] Department of Computing Security, Rochester Institute of Technology, Rochester, USA
tommy.chin@ieee.org

[2] Florida Center for Cybersecurity, University of South Florida, Tampa, USA
xiongk@usf.edu,{chengbin,yli13}@mail.usf.edu

Abstract. Malware or threat actors use a Command and Control (C2) environment to proliferate and manage an attack. In a sophisticated attack, a threat actor often employs a Domain Generation Algorithm (DGA) to cycle the network location in which malware communicates with C2. Network security controls such as blacklisting, implementing a DNS sinkhole, or inserting a firewall rule is a vital asset to an organization's security posture. However, all of them are typically ineffective against a DGA. In this paper, we propose a machine learning framework for identifying and clustering domain names to circumvent threats from a DGA. We collect a real-time threat intelligent feed over a six month period where all domains have threats on the public Internet at the time of collection. We then apply the proposed machine learning framework to study DGA-based malware. The proposed framework contains a two-level model, which consists of classification and clustering is used to first detect DGA domains and then identify the DGA of those domains. Our extensive experimental results demonstrate the accuracy of the proposed framework. To be precise, we achieve accuracies of 95.14% for the first-level classification and 92.45% for the second-level clustering, respectively.

Keywords: Malware · Domain Generation Algorithm
Machine learning · Security · Networking

1 Introduction

A computer network and its assets are frequently under a variety of attacks including malware attacks, where threat actors attempt to infiltrate layers of protection and defensive solutions [1–3]. Anti-malware software, for the longest time, is a critical asset for an organization as it provides a level of security on computer systems to deter and remove malicious threats. However, many anti-malware solutions typically utilize static string matching approaches, hashing

R. Beyah et al. (Eds.): SecureComm 2018, LNICST 254, pp. 433–448, 2018.
https://doi.org/10.1007/978-3-030-01701-9_24

schemes, or network communication whitelisting [4]. Sophisticate threat actors and authors/developers to new malware strands purposely integrate evasive techniques and covert communication channels to bypass most detection techniques, which presents a grand challenge in securing an enterprise.

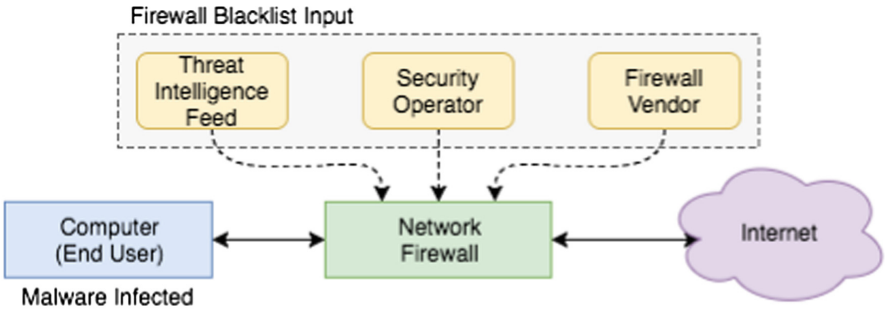
One component to some variations of malware strands is a method to communicate with a centralized server to service a Command and Control (C2) using either a static or dynamic method [5]. In the static method, the malware has been pre-written with a value such as an IP address or a domain name that becomes permanently fixed throughout the lifespan of the malware and that once a security operator identifies such an illicit network, a simple firewall rule will relieve the threat. In the dynamic method, the creator of the malware implements a technique to communicate back to a variety of servers, based on a sequencing approach known as Domain Generation Algorithm (DGA) [6]. The dynamics of a DGA commonly utilizes a seeded function. That is, given an input such as a timestamp, a deterministic output would follow as pre-defined by the DGA. The challenge behind deterring a DGA approach is that an administrator would have to identify the malware, the DGA, and the seed value to filter out past malicious networks and future servers in the sequence.

Network security measures such as Access Control List (ACL), firewalls, and Domain Name System (DNS) sinkholes have been the prominent best practice to reduce the proliferation of unauthorized access and the spread of malware strands. A DGA, however, increases the difficulty to control malicious communication as a sophisticated threat actor implements the ability to change the server or location periodically the malware communicates back (callback) to the C2 in an automated fashion. Overall, utilizing a DGA would primarily establish a game of cat and mouse for both security operators mitigating the threat while the centralized server for the C2 would frequently change location.

This study evaluates known malicious domains that exclusively belong to DGAs and we attempt to apply machine learning approaches including multiple feature extractions, classification, and clustering techniques. Computer systems frequently query domain names using DNS due to vastly broad running applications and services [7]. Security appliances that monitor and evaluate each DNS query needs to determine whether a particular domain has some level of maliciousness and specifically, whether or not a specific query originates from a DGA. If so, which one. Moreover, this study utilizes a real-time threat intelligence feed that has been collected over a six-month period on a daily basis while leveraging high-performance nodes [8,9] from the Global Environment for Network Innovation (GENI) [10] to conduct extensive data processing. We further propose a machine learning framework to classify and detect DGA malware and experimentally evaluate the proposed framework through a comparison of various machine learning approaches. Specifically, our machine learning framework consists of the following three main components: (1) Blacklist with a pattern filter that first filters the incoming DNS queries and stores them in the blacklist. (2) Feature extractor that extracts features from those domains that are not in the blacklist. The domains will be sent to the next component. (3) Two-level

classification and clustering. To identify DGA domains, we start with the first-level classification to classify DGA domains and normal domains. We then apply the second-level clustering to group domains sequenced by the DGA. The overall goal is to determine the technique the DGA employs so that our proposed framework can prevent future communication to the C2. Our evaluation results show that we can achieve the accuracy of 95.14% for the first-level classification and the accuracy 92.45% for the second-level clustering, respectively.

The rest of the paper is organized as follows. Section 2 defines the research problem while Sect. 3 demonstrates existing related work. Section 4 presents data collection and the proposed machine learning framework and Sect. 5 dicusses the experimental and evaluation of the framework. Lastly, Sect. 6 concludes our studies and presents future work.



DGA Sequence	Firewall Blacklist	Internet Domain	Filtered
swgqwmnpxuorh.com	swgqwmnpxuorh.com	swgqwmnpxuorh.com	Yes
ghluburrkboax.net	ghluburrkboax.net	ghluburrkboax.net	Yes
veivkqvvamlgx.biz	veivkqvvamlgx.biz	veivkqvvamlgx.biz	Yes
wykwohntbxrlh.org		wykwohntbxrlh.org	No
kjpbsprvnertx.co.uk			
ahmsbkdcffkst.info		ahmsbkdcffkst.info	No

Fig. 1. Threat models: multiple conditions for a DGA to function in a network environment where filtering results in a firewall that prevents the communication and an empty cell in Internet domain that results in an NX domain error. Note, the domains listed in the figure belong to existing live threats.

2 Problem Statement

Firewall blacklisting constantly expands as the multiple sources of inputs expand filtering rules. However, sequences in a DGA may not be known to these inputs

promptly. Moreover, for the malware to communicate correctly to an appropriate domain, a threat actor must register each respective domain name in the sequence to maintain the C2 or risk the loss of a node in the distribution. Figure 1 demonstrates an example scenario for such a case.

Our research problem is to accurately identify and cluster domains that originate from known DGA-based techniques where we target to obtain a security apparatus that autonomously mitigates network communication to unknown threats in a sequence.

2.1 Assumptions and Threat Models

Threat actors need a method to control and maintain the malware in a C2 environment while operating in an unnoticeable manner from network security systems hence, a DGA. The successfulness of the malware does not require a domain to be registered or valid and that a DGA may iterate a sequence that results in an NX DOMAIN situation (unregistered). Blacklisting, establishing a DNS sinkhole, or implementing a firewall rule are standard techniques to prevent malicious network activity from malware and the signatures to implement these mitigation techniques are often provided by threat intelligence feeds. However, this research does not utilize any blacklisting or pre-known malicious domains to block traffic derived from a DGA in the initial stages of our analysis and that such features are built over our observations. The main reason behind our implementation is that many threat intelligence feeds and heuristic data often provide signatures to malware that has plagued a network or public Internet. A sophisticated threat actor would implement or utilize a 0-day style malware (malicious code that has never been seen or known to the public) and therefore, blacklisting would be inappropriate for our analysis. Our proposed machine learning framework aims to solve the problem of detecting DGA sequences using machine learning techniques derived from observations in a network.

3 Related Work

Current Internet and end-user systems are frequently plagued with the hazards of malware. The landscape of the modern Internet grows as mobile devices, Internet of Things (IoT), and network connected vehicles expand where a threat actor may attack the increasing number of potential targets a daily basis [11,12]. Many of such systems are susceptible to malware attacks due to mismanagement issues, poor patching behaviors, and dangerous 0-day attacks [13].

Because DGA-generated domain names contain significant features that can be used to differentiate DGA domain names from normal ones [14]. Therefore, many studies aim to target blocking those DGA domain names as an defense approach [15,16]. The DGA that generates the domain fluxing botnet needs to be known so that we can take countermeasures. Several studies have looked at understanding and reverse engineering the inner workings of botnets [17,18]. Thomas et al. [19] proposed an automatic method to extract DGA from current

malware. Their study focused on domain fluxing malware and relied on the binary extraction for DGA. Their approach is only effective for certain types of malware [20]. Besides blocking and extracting DGAs from normal domains, deeper study has been explored based on the features of DGA domain names.

Since the DGA domains names are usually randomly generated, the lengths of DGA domains are very long, which is a good feature that can be used for detecting DGA domains. However, shorter DGA domain names are more difficult to detect. This is because most normal domains are tend to be short. Ahluwalia et al. [21] proposed a detection model that can dynamically detect DGA domains. They apply information theoretic features based on the notion of domain length threshold. Their approach can dynamically detect the DGA domains with any length. Many other works have been done on DGA detection based on the DGA domain features.

Ma et al. [22] proposed a lightweight approach to detect DGA domains based on URLs using both lexical and host-based features. They consider lexical features of the URL such as length, number of dots, and special characters in the URL path. Antonakakis et al. [23] propose a novel detection system, called Pleiades. They extract a number of statistical features related to the NXDomain strings, including distribution of n-grams. Wang et al. [24] proposed using word segmentation to derive tokens from domain names to detect malicious domains. The proposed feature space includes the number of characters, digits, and hyphens. Similar to Ma et al. [22], McGrath et al. [25] also take a close look at phishing URLs and found that the phishing URLs and DGA domains have different characteristic when compared with normal domains and URLs. Therefore, they proposed a model for detecting DGA domains based on domain length comparison and character frequencies of English language alphabets. The similar approach based on DGA features can be find in [15,26].

In order to classify DGA domain names, Schiavoni et al. [5] proposed a feasible approach for characterizing and clustering DGA-generated domains according to both linguistic and DNS features. In the study, they proposed that DGA domains have groups of very significant characters from normal domains. By grouping the domains according to their features, the authors applied a machine learning classifier could distinguish them from all the domains easily. Several machine-learning techniques have been studied to classify malicious codes. They include neural networks, support vector machines (SVM) and boosted classifiers [27]. There are also several studies aiming to predict DGA domain names from historical DGA domains [28]. Woodbridge et al. [29] used DNS queries to find the pattern of different families of DGAs. Their approach does not need a feature extraction step but requires a long short-term memory (LSTM) network, which needs time to accumulate data. Similar to Woodbridge et al. [29], Xu et al. [30] checked DNS similarity and pattern to predict future DGA domains. Their approach is effective for some DGAs. Recently, researches have proposed deep learning techniques for detecting DGAs learn features automatically, which require no effort from human for feature analysis [31,32].

4 Design

Establishing a viable source for this research requires two components: (1) domains derived from a DGA; (2) machine learning that (2a) would encompass multiple feature extraction techniques and (2b) would entail clustering the domains. Multiple online sources from simple Google searching provide example codes for a DGA construction. However, a majority of these techniques are trivial and fundamental at best. Online threat intelligence feeds give a more realistic approach to examine current and live threats that roam the public Internet. This section describes the approach for data collection and proposed a machine learning framework for DGA malware analysis.

4.1 Threat Intelligence Feed and Ongoing Threat Data

DGAs are plentiful through multiple online examples that are found from Google searching and Github repositories. However, sophisticated threat actors purposely create tailored DGA to evaluate current detection systems. Using real-time active malicious domains derived from DGAs on the public Internet measures the accuracy of the proposed approach. Specifically, threat intelligence feeds collected from Bambenek Consulting [33] over a period of six months were obtained through daily manual querying demonstrated trends of ongoing threats. The structure of the data is presented in a CSV format of domain names, originating malware, and DGA membership with the daily file size of approximate 110 MB. Figure 2 demonstrates an example feed from the collected data.

```
sjthgkvsnpkfq.net,Domain used by Cryptolocker - Flashback DGA for 16 Jan 2018,2018-01-16
tfysaplgjhsnp.biz,Domain used by Cryptolocker - Flashback DGA for 16 Jan 2018,2018-01-16
ubvjhbndmsjoj.ru,Domain used by Cryptolocker - Flashback DGA for 16 Jan 2018,2018-01-16
vwbubgdqikrwr.org,Domain used by Cryptolocker - Flashback DGA for 16 Jan 2018,2018-01-16
wlxnxfvwqsnyu.co.uk,Domain used by Cryptolocker - Flashback DGA for 16 Jan 2018,2018-01-16
```

Fig. 2. Example sample dataset from Bambenek consulting gives domain names, malware origins, DGA schema, and date collected.

4.2 The Machine Learning Framework

We propose a machine learning framework that consists of three important steps, as shown in Fig. 3. We first have the input DNS queries with the payload, then it will be passing into our process step, which consists of 4 important components: (1) We first use a domain-request packet filter to get domain names and store them in the blacklist. If the input is a known domain, we will skip (2)–(4), and directly go to the output, otherwise, we will proceed to next component. (2) Then, a feature extractor is used to extract domain features. (3) Next, we apply the first-level classification to distinguish DGA domains and non-DGA domains

and second-level clustering to group similar DGA domains. (4) Finally, we detect the DGA domains. After the domain name goes through the process step, we will append this domain to the blacklist.

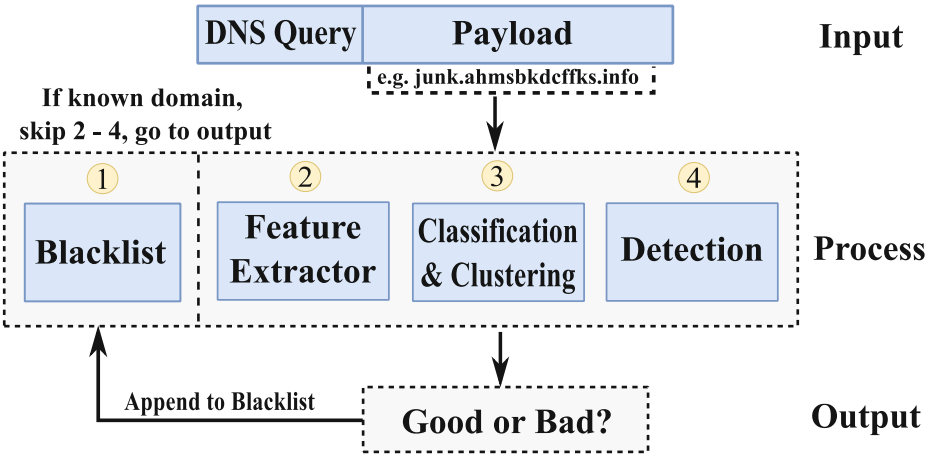


Fig. 3. Machine learning framework.

The rest of this section discusses these important steps in details.

Blacklist. To perform the classification and clustering in our following steps, we only need the information of domain names. Since the collected raw data contains some trivial information that is useless in our experiment, we apply a domain-request packet filter to remove that trivial information to obtain only domain names. This is done by using the Gruber Regex Pattern Filter [34]. All the network traffics undergo this filtering process. The filtered domain names are stored in the blacklist [35] and then sent to the feature extractor in next step.

Feature Extractor. The second step is to extract features from the domain names obtained from the first step. We consider each domain name as a string. To efficiently classify domains, we use two types of features: linguistic features and DNS features. We start with the discussions of linguistic features and then the DNS features.

There are six linguistic features: Length, Meaningful Word Ratio, Pronounceability Score, Percentage of Numerical Characters, Percentage of the Length of the Longest Meaningful String (LMS), and Levenshtein Edit Distance. The detailed description and calculation of each linguistic feature are given as follow:

Length: This feature is simply the length of a domain name, denoted by $|d|$.

Meaningful Word Ratio: This feature measures the ratio of meaningful words in a string (domain name). It can be calculated as follow:

$$f_1 = \sum_{i=1}^{n} \frac{|w_i|}{|d|} \tag{1}$$

where w_i is the i-th meaningful substring of this string, $|w_i|$ is the length of i-th meaningful substring. A high ratio stands for a safer domain, whereas a lower ratio indicates that it could be a DGA domain. We strict the length of each meaningful substring $|w_i|$ in the string to be at least 4 letters because most legitimate domain names have meaningful substrings with more than 3 letters. For example, for a domain name of *yivdbook*, we have $f_1 = (|book|)/8 = 4/8 = 0.5$. If a domain name is *homedepot*, we have $f_1 = (|home| + |depot|)/9 = (4 + 5)/9 = 1$, the domain is fully composed of meaningful words.

Pronounceability Score: In the linguistic sense, the more permissible the combinations of phonemes are, the more pronounceable the word is, and thus the higher pronounceability score. Since DGA generated domains have a very low number of such combinations, a pronounceability score becomes a useful feature in the first-level classification. This feature uses an n-gram lookup table to evaluate the pronounceability of a string. We calculated the feature by extracting the n-grams score of a domain d. We choose the substring length $l \in 2, 3$ in our computation and count their occurrences in the English n-gram frequency text. For a domain d, the n-grams score is calculated as follow:

$$f_2 = \frac{\sum n - gram(d)}{|d| - n + 1} \tag{2}$$

where n is the length of the matching word in the n-gram list.

Percentage of Numerical Characters: This feature measures the percentage of numerical characters in a string. It can be simply calculated by $f_3 = |n|/|d|$, where $|n|$ is the number of numerical characters.

Percentage of the Length of LMS: This feature is to measure the length of the longest meaningful string in a domain name. The calculation can be written as $f_4 = |l|/|d|$, where $|l|$ is the length of longest meaningful string.

Levenshtein Edit Distance: This feature measures the minimum number of single-character edits between a current domain and its previous domain. For example, given two strings "kitten" and "sitting", the Levenshtein Edit Distance between them is 3, because the characters that need to be edited are k to s, e to i and adding a g at the end.

Aside from linguistic features, we also look into DNS features where 27 DNS features in Table 1 are used in this research. We utilize DNS features because a DGA domain usually contains less information, whereas a legitimate domain does. For example, DGA domains tend to have short time and their creation dates are typically within one year.

Table 1. DGA classification features

Features	Description	(+/−)
Expiration date	If longer than 1 year	+
Creation date	If longer than 1 year	+
DNS record	If DNS record is documented	+
Distinct IP addresses	#. IP addresses related to this domain	+
Number of distinct countries	#. countries related this domain	+
IP shared by domains	#. domains are shared by the IP	−
Reverse DNS query results	If DN in top 3 reverse query results	+
Sub-domain	If domain is related to other sub-ones	+
Average TTL	DNS data time cached by DNS servers	+
SD of TTL	Distribution SD of TTL	−
% usage of the TTL ranges	Distribution range of TTL	+
# of distinct TTL values	Different value of TTL on server	−
# of TTL change	How frequently TTL changes	+
Client delete permission	If client has delete permission	−
Client update permission	If client has update permission	−
Client transfer permission	If client has transfer permission	−
Server delete permission	If server has delete permission	−
Server update permission	If client has update permission	−
Server transfer permission	If client has transfer permission	−
Registrar	The domain name registrar	+
Whois Guard	If use Whois guard to protect privacy	−
IP address same subnet	If IP address is in the same subnet	−
Business name	If domain has a corporation name	+
Geography location	If domain provides address	+
Phone number	If domain provides a phone number	+
Local hosting	If use local host machine	+
Popularity	If on the top 10000 domain list	+

Note: DN - Domain name. TTL - Time-To-Live. SD - Standard deviation. All the features used in our model. (+/−): "+" stands for positively related to normal domain, whereas "−" stands for negatively.

Two-Level Model: Classification and Clustering. To understand DGA domains, we propose a two-level machine learning model consisting of the first-level classification and the second-level clustering. In the former, we use a classification model called J48 classifier to classify input domains into DGA domain (bad domain) and non-DGA domain (good domain). Then, the classified DGA domains will be sent to the second-level clustering, where we use the DBSCAN-based clustering [5] to divide the DGA domains into several groups, as shown in Fig. 4.

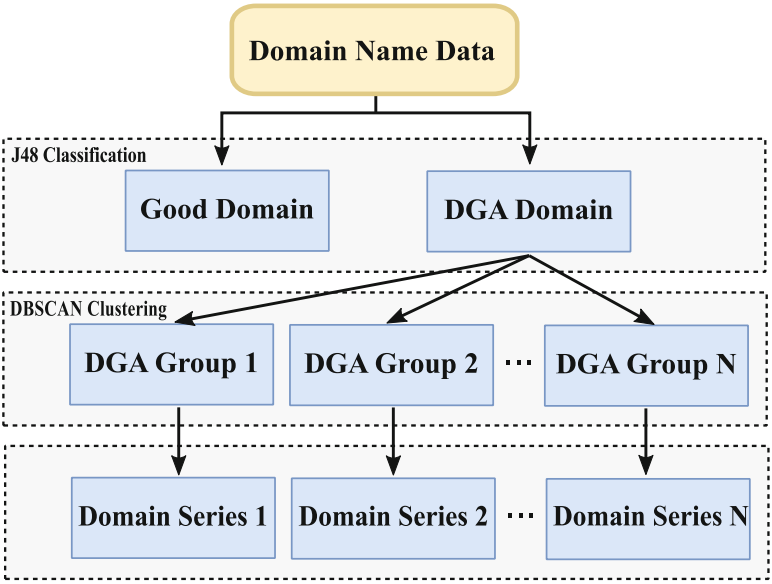


Fig. 4. Two-level model of classification and clustering

First-Level Classification: A perfect classification algorithm for classifying DGA domains and non-DGA domains requires the maximal difference between them. By using the features obtained above, we test different machine learning classifiers including Decision Tree-J48, Artificial Neural Network (ANN), Support Vector Machine (SVM), Logistic Regression and Naive Bayes to find the best classifier. Among those classifiers, we notice that J48 is the best to classify DGA domains (its detailed discussion is given in Sect. 5), so J48 is chosen as a classifier in our first-level classification.

Second-Level Clustering: Only the classified DGA domains are used for the second-level clustering. Our clustering model is based on the DBSCAN algorithm, where we use domain features to compute the distance of their domain names and to group these domains according to their domain feature difference. Let d_i and d_j be domain names, where $i \neq j$. We first set $i = 0$ representing the first domain and then calculate the overall distance between d_i and all other domains. Since we have two types of features: linguistic feature and DNS feature, the overall distance is a combination of linguistic distance and DNS similarity. The linguistic distance is computed based on the six linguistic features followed by the following equation:

$$D_l(d_i, d_j) = \sqrt{\sum_{k=1}^{6} dis_k(d_i, d_j)}, \tag{3}$$

where $dis_k(d_i, d_j)$ is the distance of each linguistic features between two domains d_i and d_j. To get the DNS similarity, we first construct a weight matrix $\mathbf{M} \in \mathbb{R}^{K \times L}$, where K and L are the number of DNS features and linguistic features of all the DGA domains $\mathbb{D}$ classified from the first-level classification, respectively. The relationship between K and L is represented by a bipartite graph that is represented in $\mathbf{M}$, where each component $\mathbf{M}_{k,l}$ holds the weight of an edge (l, k). For each DNS record, weight $\mathbf{M}_{k,l}$ is computed by:

$$\mathbf{M}_{k,l} = \frac{1}{|\mathbb{D}(k)|}, \text{ for any } l = 1, ..., L, \tag{4}$$

where $|\mathbb{D}(k)|$ is the cardinality of the subset of domains that are pointed to the DNS record. We then use a matrix $\mathbf{S} \in \mathbb{R}^{L \times L}$ to store DNS similarity information, where for each component, $\mathbf{S}_{d_i,d_j}$ is the similarity value of domains d_i and d_j. Our intuition is that when two domains point to the same DNS record k, they should have high similarity. Therefore, we could calculate the similarity matrix based on the weight matrix $\mathbf{M}$. Let $\mathbf{N}$ be $\mathbf{M}$ normalized by columns. We have:

$$\mathbf{N} \equiv \mathbf{M} \text{ when } \left(\sum_{k=1}^{K} M_{k,l} = 1, \forall l = 1, \cdots, L \right). \tag{5}$$

Final similarity matrix is calculated by:

$$\mathbf{S} = \mathbf{N}^T \odot \mathbf{N} \in \mathbb{R}^{L \times L}. \tag{6}$$

The overall distance is a combination of linguistic distance and DNS similarity, which is calculated by:

$$D(d_i, d_j) = S_{d_i,d_j} + log(\frac{1}{D_l(d_i, d_j)}) \tag{7}$$

After we have the overall distance, we can get all points density-reachable from d_i based on the threshold distance, ϵ. If $D(d_i, d_j) > \epsilon$, we add those points d_j to a cluster C. The minimal cluster points, $MinPts$, is used to determine a core point. Let d_i be a core point. If the number of point in $C > MinPts$, then a cluster is formed. If d_i is a border point, implying that no points are density-reachable from d_i, then our DBSCAN model visits the next domain. The above steps will be repeated until all of the domains have been processed.

5 Evaluation

5.1 Global Environment for Network Innovation

GENI is an NSF funded heterogenous testbed solution. Leveraging high-performance nodes aided in the ability to process large volumes of real-time data feeds in a timely manner. The nodes selected for the evaluation consisted of systems running: Intel(R) Xeon(R) CPU E5-2450 @ 2.10 GHz, 16 GB of hard drive space, and 1GB of memory where the size could be manipulated, based on reservation.

Experimental Setup. To evaluate our model thoroughly, we use five datasets of DGA domain data: CryptoLocker, Tovar, Dyre, Nymaim, and Locky from the latest DGA-feed [36–38]. We collected the DGA domain names over a period of six months in 2017. 160,000 domain names were tested in our model. To provide a list of normal control group domain names, we choose the top 1 million most popular Internet domains listed in domain punch [39]. We mix the control domain names and DGA domains names with a 1:1 ratio for the first-level classification. In the second-level clustering, we use classified DGA domain names from the first-level classification to cluster different groups of DGA domains.

5.2 Our Execution and Results

Experimental Results. To find the best model for the first-level classification, we test five different machine learning models, J48, ANN, SVM, Logistic Regression, and Naive Bayes. Figures 5(A) and (B) show the performance of different algorithms on the classification of the DGA domains. We find that J48 has the highest average accuracy, 95.14%, compared to other machine learning algorithms. Figure 5(B) also shows that J48 is the fastest one with an average of 0.0144 ms to classify the domain names. To see the accuracy of J48 associated with scalability, we test five groups of samples for each DGA generated domain with a total number of 1000, 5000, 15000, 20000, 50000 domain names. We find that J48 performs the best for CryptoLocker domain names.

Figure 6(A) shows how the second-level clustering algorithm performs on different DGAs. When we use both linguistic distance and DNS similarity as the overall distance, its average accuracy is 87.64%, whereas if we only use DNS similarity as the overall distance, the average accuracy is 89.02%. This is because most of DGAs have very similar string composition and length. These features can not help the clustering algorithm to identify similar DGA domains from each other. Furthermore, we test the accuracy of clustering when more groups are mixed together.

As Fig. 6(B) shows, we test all the two group combinations for all the five DGAs. When we mix Cryptolocker with other DGAs, the average accuracy for clustering is 81.43% for all the features. However, when we use only DNS features as the DBSCAN distance, its accuracy increases to 92.45%, which means that most Cryptolocker domains are clustered into one group. Similarly, when testing other groups, we find that the accuracies of clustering are 91.05%, 92.22%, 92.89%, and 92.57% for ovar, Dyre, Nymaim, and Locky, respectively. The result demonstrates that the clustering model is efficient to group the same DGA domains into one group.

5.3 Discussions

As seen in our experimental evaluation, the proposed machine learning framework has demonstrated the efficient way to predict a future DGA domain name. We have evaluated the proposed machine learning framework with most latest DGA domain names from DGA-feed to cluster and predict DGA domains

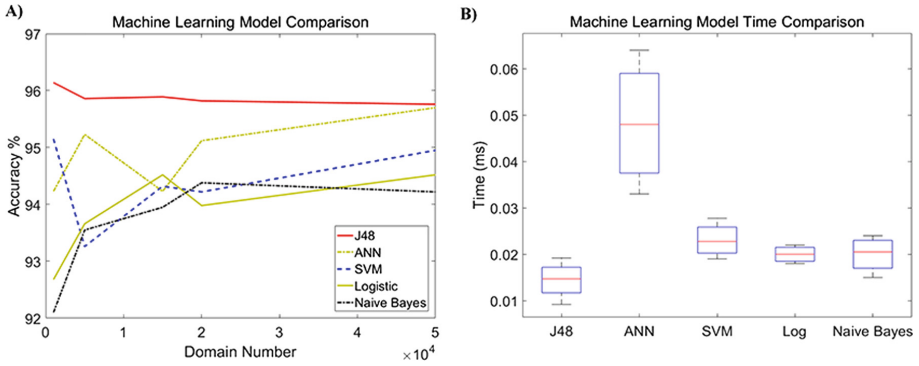


Fig. 5. (A) Accuracy of different machine learning algorithms (B) Classification time of different machine learning algorithms.

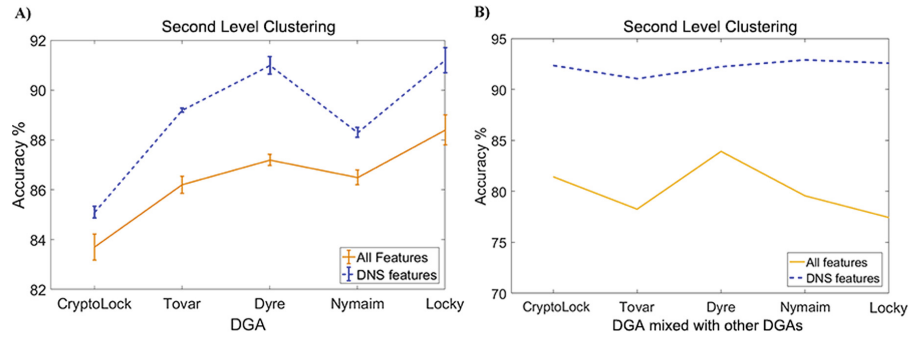


Fig. 6. (A) Clustering accuracy for each DGA. (B) Clustering accuracy for each two DGAs group.

from these real-world data. Our evaluation has shown that with 33 features we proposed in our model, the J48 classification algorithm performed the most effective and efficient in comparison to ANN, SVM, Logistic, and Naive Byes due to the minimal classification time, 0.0144 ms, and the highest accuracy, 95.14%. We have also tested the clustering accuracy. Our result has shown that the DBSCAN clustering model has the accuracy of 92.45%. We have noticed that the best accuracy we get from clustering is the one where only DNS query features are used. The experimental results have proved that a cluster of DGA domains usually points to several specific server IPs. DNS information of these domains are very similar and therefore clustering them with only DNS features is very accurate.

6 Conclusions and Future Work

The dichotomy of DGA in malware presents a grant challenge in securing an organization. Firewall blacklisting is constantly expanded since filtering rules

are constantly added through the multiple sources of inputs. However, DGA sequences may not be known by the multiple sources promptly as sophisticated malware developers can integrate DGA to bypass a majority of network controls. In this research, we have proposed the machine learning framework to circumvent threats from a DGA. The proposed framework consists of a blacklist, feature extractor, classification and clustering, and detection. Furthermore, we have collected a real-time threat intelligence feed over a six month period where all domains live threats on the Internet. Based on our extensive experiments on the real-world feed, we have shown that the proposed framework can effectively extract domain name features as well as classify, cluster and detect domain names. In the future, we will explore deep learning algorithms such as Convolution Neural Network (CNN) via tensor-flow for this research and evaluate them on a real-world testbed such as GENI [40–42].

Acknowledgments. We acknowledge National Science Foundation (NSF) to partially sponsor the research work under grants #1633978, #1620871, #1636622, #1651280, and #1620862, and BBN/GPO project #1936 through an NSF/CNS grant. We also thank the Florida Center for Cybersecurity (FC2) located at the University of South Florida (USF) to support the research through its funding that is open to all institutions in the State University System of Florida.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of NSF, FC2, and USF.

References

1. Rieck, K., Holz, T., Willems, C., Düssel, P., Laskov, P.: Learning and classification of malware behavior. In: Zamboni, D. (ed.) DIMVA 2008. LNCS, vol. 5137, pp. 108–125. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70542-0_6
2. Chin, T., Xiong, K., Rahouti, M.: SDN-based kernel modular countermeasure for intrusion detection. In: Lin, X., Ghorbani, A., Ren, K., Zhu, S., Zhang, A. (eds.) SecureComm 2017. LNICST, vol. 238, pp. 270–290. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78813-5_14
3. Ghosh, U., et al.: An SDN based framework for guaranteeing security and performance in information-centric cloud networks. In: Proceedings of the 11th IEEE International Conference on Cloud Computing (IEEE Cloud) (2017)
4. Khancome, C., Boonjing, V., Chanvarasuth, P.: A two-hashing table multiple string pattern matching algorithm. In: Tenth International Conference on Information Technology: New Generations (ITNG), pp. 696–701. IEEE (2013)
5. Schiavoni, S., Maggi, F., Cavallaro, L., Zanero, S.: Phoenix: DGA-based botnet tracking and intelligence. In: Dietrich, S. (ed.) DIMVA 2014. LNCS, vol. 8550, pp. 192–211. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08509-8_11
6. Sood, A.K., Zeadally, S.: A taxonomy of domain-generation algorithms. IEEE Secur. Priv. **14**(4), 46–53 (2016)
7. Xiong, K.: Multiple priority customer service guarantees in cluster computing. In: Proceedings of the IEEE International Symposium on Parallel & Distributed Processing (IPDPS), pp. 1–12. IEEE (2009)

8. Xiong, K.: Resource optimization and security for cloud services. Wiley, Hoboken (2014)
9. Xiong, K.: Resource optimization and security for distributed computing (2008). https://repository.lib.ncsu.edu/handle/1840.16/3581
10. Mark, B., et al.: GENI: a federated testbed for innovative network experiments. Comput. Netw. **61**, 5–23 (2014)
11. Xiong, K., Chen, X.: Ensuring cloud service guarantees via service level agreement (SLA)-based resource allocation. In: Proceedings of the IEEE 35th International Conference on Distributed Computing Systems Workshops, ICDCS Workshops, pp. 35–41. IEEE (2015)
12. Chin, T., Xiong, K.: Dynamic generation containment systems (DGCS): A moving target defense approach. In: Proceedings of the 3rd International Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC), vol. 00, pp. 11–16, April 2016
13. Sornalakshmi, K.: Detection of DoS attack and zero day threat with SIEM. In: International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 1–7. IEEE (2017)
14. Yadav, S., Reddy, A.L.N.: Winning with DNS failures: strategies for faster botnet detection. In: Rajarajan, M., Piper, F., Wang, H., Kesidis, G. (eds.) SecureComm 2011. LNICST, vol. 96, pp. 446–459. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31909-9_26
15. Yadav, S., Reddy, A.K.K., Reddy, A.N., Ranjan, S.: Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. IEEE/ACM Trans. Netw. **20**(5), 1663–1677 (2012)
16. Guo, F., Ferrie, P., Chiueh, T.: A study of the packer problem and its solutions. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) RAID 2008. LNCS, vol. 5230, pp. 98–115. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87403-4_6
17. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F.C., et al.: Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. LEET **8**(1), 1–9 (2008)
18. Zhang, L., Yu, S., Wu, D., Watters, P.: A survey on latest botnet attack and defense. In: IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 53–60. IEEE (2011)
19. Barabosch, T., Wichmann, A., Leder, F., Gerhards-Padilla, E.: Automatic extraction of domain name generation algorithms from current malware. In: Proceedings of NATO Symposium IST-111 on Information Assurance and Cyber Defense, Koblenz, Germany (2012)
20. Gardiner, J., Nagaraja, S.: On the security of machine learning in malware c&c detection: a survey. ACM Comput. Surv. (CSUR) **49**(3), 59 (2016)
21. Ahluwalia, A., Traore, I., Ganame, K., Agarwal, N.: Detecting broad length algorithmically generated domains. In: Traore, I., Woungang, I., Awad, A. (eds.) ISDDC 2017. LNCS, vol. 10618, pp. 19–34. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69155-8_2
22. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1245–1254. ACM (2009)
23. Antonakakis, M., et al.: From throw-away traffic to bots: detecting the rise of DGA-based malware. In: USENIX security symposium, vol. 12 (2012)
24. Wang, W., Shirley, K.: Breaking bad: detecting malicious domains using word segmentation. arXiv preprint arXiv:1506.04111 (2015)

25. McGrath, D.K., Gupta, M.: Behind phishing: an examination of phisher modi operandi. LEET **8**, 4 (2008)
26. Mowbray, M., Hagen, J.: Finding domain-generation algorithms by looking at length distribution. In: IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 395–400. IEEE (2014)
27. Shabtai, A., et al.: Detection of malicious code by applying machine learning classifiers on static features: a state-of-the-art survey. Information Security Technical Report (2009)
28. Sharifnya, R., Abadi, M.: A novel reputation system to detect DGA-based botnets. In: 3th International eConference on Computer and Knowledge Engineering (ICCKE), pp. 417–423. IEEE (2013)
29. Woodbridge, J., Anderson, H.S., Ahuja, A., Grant, D.: Predicting domain generation algorithms with long short-term memory networks. arXiv preprint arXiv:1611.00791 (2016)
30. Xu, W., Sanders, K., Zhang, Y.: We know it before you do: predicting malicious domains. In: Virus Bulletin Conference (2014)
31. Yu, B., Gray, D.L., Pan, J., De Cock, M., Nascimento, A.C.: Inline DGA detection with deep networks. In: IEEE International Conference on Data Mining Workshops (ICDMW), pp. 683–692. IEEE (2017)
32. Saxe, J., Berlin, K.: eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. arXiv preprint arXiv:1702.08568 (2017)
33. Bambenek: OSINT feeds from bambenek consulting. Bambenek Consulting
34. Yang, L., Karim, R., Ganapathy, V., Smith, R.: Fast, memory-efficient regular expression matching with NFA-OBDDs. Comput. Netw. **55**(15), 3376–3393 (2011)
35. Kührer, M., Rossow, C., Holz, T.: Paint it black: evaluating the effectiveness of malware blacklists. In: Stavrou, A., Bos, H., Portokalidis, G. (eds.) RAID 2014. LNCS, vol. 8688, pp. 1–21. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11379-1_1
36. JBT Organization: Domain feed of known DGA domains (2017)
37. Jarvis, K.: Cryptolocker ransomware. Viitattu **20**, 2014 (2013)
38. Chaignon, P.: A collection of known domain generation algorithms (2014)
39. Technologies: Top million websites & TLDs (2016)
40. Chin, T., Mountrouidou, X., Li, X., Xiong, K.: An SDN-supported collaborative approach for DDoS flooding detection and containment. In: 2015 IEEE Military Communications Conference, MILCOM 2015, pp. 659–664. IEEE (2015)
41. Lenkala, S.R., Shetty, S., Xiong, K.: Security risk assessment of cloud carrier. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 442–449. IEEE (2013)
42. Xiong, K., Perros, H.: SLA-based service composition in enterprise computing. In: 16th International Workshop on Quality of Service, IWQoS 2008, pp. 30–39. IEEE (2008)