A dictionary-based method for detecting machine-generated domains

Tianyu Wang , Li-Chiou Chen & Yegin Genc

INFORMATION SECURITY JOURNAL: A GLOBAL PERSPECTIVE
https://doi.org/10.1080/19393555.2020.1834650

A dictionary-based method for detecting machine-generated domains

Tianyu Wang[a], Li-Chiou Chen[b], and Yegin Genc[b]

[a]Department of Computer Science, Pace University, New York, USA; [b]Department of Information Technology, Pace University, New York, USA

ABSTRACT
Internet robots, also known as bots, have transformed the businesses and society with convenience. However, the dynamics of these interactions could be under adversarial circumstances with detrimental effects on network security. Bots that use domain-generation algorithms (DGAs) can generate many random domains dynamically so that static domain blacklists become ineffective in preventing malicious attacks by botnets. Various families of recent botnets have used DGA to establish communication with the bots. Researchers have introduced various detection methods with moderate success. Methods proposed so far either detect only DGAs that use non-variations forms or focus on the classification accuracy instead of time complexity, which would be critical in real-world production. The goal of this article is to explore how machine learning can help in detecting machine-generated domain names. To that end, we propose a dictionary-based n-gram method that can detect 39 DGA variations. We compared our method with existing research and found that our method can improve the performance of the existing classification algorithms. At last, our method can achieve competitive results as the LSTM model while requiring less time and complexity. Our research helps real-time production for DGA detection and provides insight in protecting DNS server and information security.

KEYWORDS
Network security; botnet; domain names; classification algorithms; machine learning

1. Introduction

Internet bots, computer programs that aim to perform repetitive tasks, have significantly transformed the diffusion of information and communications (Dunham & Melnick, 2008). Besides their positive impacts (Tsvetkova et al., 2017), their ill-intended use by attackers is a significant concern among the information security community. They have matured and have become exceedingly sophisticated over the years (Negash & Che, 2015) (Z. Li & Liao, 2014). One common ill-use of such bots is domain name generation, which is also the focus of this work. Attackers use algorithmically generated domain names to maintain communications with the systems they have compromised. Such communication activities, known as command & control (C&C) communications, are integral to the continuity of any attack. C&C communications fail when their hosting domains are detected and blocked. Moreover, usually, once a domain name is associated with C&C communications, it is blacklisted, forcing attackers to use another domain name. To that end, attackers have begun generating domain names dynamically to pass through such blacklists. Specifically, they use domain name generation algorithms (DGAs) to create multiple domain names and register only a subset of these domain names for actual use – a process commonly referred to as "domain fluxing." With domain fluxing, attackers query each of the auto-generated domain names until one of them resolves to an IP address. The resolved IP-address obtained is then used to host the command-and-control (C&C) activities (Yadav et al., 2010) (Yadav et al., 2012).

In short, the use of DGA allows attackers and victims machines to contact the same domain at a given time, so the exchange of information is possible at least temporarily. These connections points are hard to predict for parties who do not have access to the algorithm.

Recent examples of malware attacks with DGA features are variations of the Mirai botnet (Rodriguez, 2016). Mirai botnet has become one

CONTACT Tianyu Wang tianyu.wang@pace.edu Department of Computer Science, Pace University, New York, NY 10570
Supplemental data for this article can be accessed on the publisher's website.

of the most widespread malwares that infected many the Internet of Things (IoT) devices. Mirai botnet can turn networked devices running Linux into remotely controlled "bots" for large-scale network attacks. It primarily targets online consumer devices such as IP cameras and home routers, and it has been used in some of the most significant and most disruptive attacks (Krebs, 2016; Mendez Mena et al., 2018).

The highly disruptive nature of such attacks makes preventative measures imperative. Preventing attacks with DGA features can be done in two ways. One way is to counter these attacks directly. For example, to protect systems from distributed denial of service (DDoS) attacks, technologies that counter these attacks can be implemented. The other method is to automatically predict DGA domain names so that the communication between the attackers and victims can be disrupted on time. Our paper focuses on the latter approach. We investigate DGA classification models using machine-learning algorithms on domain names, especially for those generated from machine and dictionary.

Many studies in the previous literature proposed different methods using varying features to detect DGA domains. A study suggested utilizing the distribution of length of domain names (Mowbray & Hagen, 2014). Other researchers proposed Phoenix – another detecting model that uses a combination of string and IP-based features (Schiavoni et al., 2014). Later, a study suggested that Phoenix can be improved if the Mahalanobis distance is used for classification (Tong & Nguyen, 2016). A different approach focused on excluding human domain names based on the morphemes in the string of domain names (Wei-wei & Qian, 2013). Besides, some researchers proposed a method using masked N-grams, together with other statistics, that provides a superior detection accuracy (Selvi et al., 2019).

While we see increases in accuracy as these detection models are getting more sophisticated, three challenges leave room for improvement for both practitioners and researchers. First, many of the proposed methods have asymmetrically high false-positives rates (Brailsford & Faff, 1996; Sivaguru et al., 2018). A study shows that the annual cost of false-positives investigations can be as high as 1.37 USD million (Crowe, 2017). Second, these models are particularly ineffective if DGA methods are using a dictionary, such as Matsnu bot (Mimoso, 2014). Third, a research study indicates that the accuracy and the speed of attack detection are the most critical factors in statistical attack detection systems (Nooribakhsh & Mollamotalebi, 2020). Thus, research that provides the comparison in both overall accuracy and detection time is missing.

Our study aims to contribute to the DGA detection research by studying methods that address the issues discussed above. To that end, we extended the traditional N-gram method in Nature Language Processing (NLP) and developed two dictionaries to identify DGA domains. These dictionaries help to build features for a similarity-based detection algorithm. Our findings show that these features are more effective in classification tasks than existing features, such as masked n-grams (Selvi et al., 2019). Our features achieved better results than the existing ones. We found that dictionary-based N-gram features can be more useful for detecting 39 DGA variations than entropy-based features and K-L divergence features. Besides, our features can not only help identify dictionary-based DGA but also improve the performance of existing detection models by complementing more established feature extraction methods.

To study the overall detection performance, we implemented these features to four machine learning (ML) classifiers (i.e., Random Forest, Support Vector Machine, Naïve Bayes, and XGBoost). We compared the results among each other and with two deep learning classifiers that do not require feature engineering (i.e., Dense Neural Network (DNN) and Long Short-term Memory (LSTM) models.) Our experimental results showed that ML classifiers using dictionary-based N-gram features can achieve competitive results to the LSTM model but requires less time complexity. These findings, we argue, can help real-time production for DGA detection and provide insight in protecting DNS server and information security.

The rest of the paper is organized as follows: in section 2, we discuss the background of the domain name system, DGA algorithms, and related security issues. Next, we provide the literature review in section 3. Section 4 presents our DGA detection method. In section 5, we detail our experiments and present results. Next, we discuss our findings and future research in section 6. The last section includes our concluding remarks.

2. Background

The Domain Name System (DNS) performs as a phonebook of the Internet allows humans to access information online through domain names. DNS translates domain names to IP addresses is called DNS queries, which are typically neither encrypted, so they are vulnerable to DNS-based misuse and malicious activities. For example, It allows for a malicious payload to beacon out to an adversary; commands could also be received to the requesting application for processing with little difficulty (Das et al., 2017). Botnets often utilize such them to avoid detection.

A botnet, short for "robot network," is a network of devices infected by malware, of which the owner is typically unaware that the devices are compromised. The infected devices are under the control of an attacking party, which is called botmaster, to perform various tasks. Botnets have become the central platform for cybercriminals to send spam or phishing e-mails and attempt to steal information (C. Li et al., 2009). A bot could query a predefined Command and Control (C&C) domain name that resolves the IP address of a server that the malware commands will be received. However, botmasters can lose control of the entire botnet when a C&C server is taken down. Nowadays, to hide the location of C&C servers and to dynamically connect bots to different servers, botmasters have developed various botnets that can locate their servers through random domain names that are generated by computer algorithms, which are called domain-generation algorithms (DGA).

Domain-Generation Algorithms (DGA) is the algorithms that automatically generate domain names for candidate C&C servers based on a given random seed (Yadav et al., 2010), (Krebs, 2014). The bots attempt to resolve these domains by sending DNS queries until one of the domains resolves to the IP address of a C&C server. Because attackers have stopped using hard-coded domain lists, but instead, use this convenient method to keep attacking resilience. If domain names are identified and taken down, the bot will eventually get the valid IP address and using DNS queries to the next DGA domain. The majority of methods that generate DGA domains fall into one of the following three categories. The first category involves the use of time-sensitive "seeds" for random text strings generation. Methods in the second category use a random generator seed to generate random text strings – for example, Conficker, Cryptolocker, and Zeus DGA (Krebs, 2014; Leder & Werner, 2009). The third types are dictionary-based methods, such as Matsnu. These models randomly select several words from a dictionary and combine them to make up DGA domains (Woodbridge et al., 2016).

3. Related work

Many machine-learning-based classification methods have been used to monitor DNS traffic. Regarding the practice of "IP fast fluxing," e.g., where the botnet owner continually keeps changing the IP-addresses mapped to a C&C server implements a detection mechanism based on passive DNS traffic analysis (Perdisci et al., 2009; Pereira et al., 2018). Some researchers presented a technique to detect randomly generated domains that most of the DGA-generated domains would result in Non-Existent Domain responses and that bots from the same bot-net would generate similar NXDomain traffic (Antonakakis et al., 2011, 2012). Besides, earlier papers have analyzed the inner working of IP fast-flux networks for hiding spam and fraud infrastructure (Holz et al., 2008). With regards to botnet detection, many researchers calculate the correlation of network activity in time and space at campus network (Gu, Perdisci et al., 2008; Gu, Zhang et al., 2008).

Furthermore, using DNS traffic analysis to identify flux networks and botnets has been proven efficient even without knowing reverse engineered on DGA before ahead (Yadav et al., 2012), (Barthakur et al., 2013).

Another stream of research in DGA detection largely focuses on feature-based aspects. Characteristics of phishing and non-phishing URLs have been analyzed and suggest that the phishing and non-phishing URLs exhibited different alphabet distributions (McGrath & Gupta, 2008). Besides, some statistical learning techniques based on lexical features to determine if a URL is malicious has been introduced (Ma et al., 2009). An approach focused on detecting spamming botnets by developing regular-expression-based signatures for spam URLs (Xie et al., 2008). An investigated rumor identification scheme has been introduced by applying features based on users' behaviors (Liang et al., 2015).

Another group of researchers looked at the DGA detection using deep neural network. A method has been proposed that combines context-sensitive words with deep learning classifier (Koh & Rhodes, 2018). In contrast, a new approach has been introduced using two DGAs that use a Hidden Markov model and probabilistic context-free grammar methods to generate stealthy domain names that avoid detection (Fu et al., 2017). A set of heuristics method has been proposed for automatically labeling domain names monitored in real traffic (Yu et al., 2019). A novel method that applies long short-term memory (LSTM) in DGA detection has been proposed (Woodbridge et al., 2016). However, some of these methods have failed to detect dictionary-based DGA, such as Matsnu (Sivaguru et al., 2018; Woodbridge et al., 2016). Besides, another research using economic analysis has provided useful and efficient allocation guidance to botnet defenders by attacking more on new entrants to the botnet market relative to the existing botmasters (Z. Li & Liao, 2014).

4. DGA detection

4.1. Existing detection methods

Domain names are a series of text strings, consisting of alphanumeric characters and the dash sign. Thus, DGA domain name detection can be treated as classification short text – a prevalent task for machine-learning methods. We studied a set of state-of-the-art methods with different sets of features and compared them to our own. These are Random Forest, Support Vector Machines (SVM), Naïve Bayes, XGBoost, Dense Neural Network (DNN), and Long Short-Term Memory (LSTM) networks. More detailed about those classification methods can be found in Appendix Classifier section.

Since our research problem is a binary classification problem, each input entry would be classified into either a positive case or a negative case based on the definitions below:

- **Positive**, or "**dga**": a domain name generation algorithm created the domain name; and
- **Negative**, or "**legit**": a human created the domain name; therefore, the domain is legitimate.

4.2. Features

For the traditional machine-learning classifiers – other than the LSTM network, we need to calculate features from each domain name. For LSTM networks, neural networks feed input data using an embedding layer to map input data into a numerical format. Next, to uniquely classify a domain name as either dga (positive) or legit (negative), we defined and calculated five types of features: length, entropy, K-L Divergence, Jaccard Index, and N-gram-based dictionary features.

4.2.1. Length

Previous studies have shown that the number of characters of the domain name, the length, can help distinguish DGA domain names. Gaebler et al. revealed that of 1 million of the most popular websites, the average length of a domain name is six characters (Gaebler, 2009). In general, DGA domain names are much longer (more than 12 characters) than the legitimate ones (Bochkarev, Shevlyakova, & Solovyev, 2015).

4.2.2. Entropy

In information theory, entropy is the expected value of the information contained in each message. We measure the domain name entropy by computing the entropy of character distribution and evaluate the randomness of the domain names (Mitchell, 1997). In our research, we choose to use Shannon entropy as the Appendix Equation 1.

Note that x is the input message, domain names. $I(x)$ is the information content of x. $P(x)$ is the probability of the frequency of the input message. Lower values of P(x) indicate higher levels of uncertainty. We use logarithm to the base 10 (b = 10) for our calculations. Since log10 yields negative results for values less than 1, the log values are multiplied by −1.

4.2.3. K-L divergence features

S. Yadav et al. introduced K-L (Kullback-Leibler) divergence metric and Jaccard Index to measure the difference of two distributions (Yadav et al., 2012). K-L divergence metric is a measure of the distance between two probability distributions (Kullback & Leibler, 1951). Moreover, K-L divergence indicates the expectation of the logarithmic difference between two probability distributions. For the discrete probability distributions of P and Q, K-L divergence can be formulated as Appendix Equation 2. Note that n is the number of possible values for a random variable. Yadav et al. (Yadav et al., 2010, 2012) studied K-L Divergence with both unigrams (sequence of single words) and bigrams (sequence of two adjacent words). Their study suggested that results may vary based on the length of the items when used with a contiguous sequence of items, which is typically the case with textual data.

This method can easily be applied to our context if we ask the following question. Given two sets of domain names: dga names, and legit names, what is the difference between legit and dga probability distributions over their alphanumeric character sequences? More formally, we can characterize the difference as Appendix Equation 3. Note that $D_{KL} \geq 0$ indicates it is a legit domain; and, $D_{KL} \leq 0$, a DGA domain.

4.2.4. Jaccard index features

Another method to measure the similarity between two distributions is the Jaccard Index (JI) or the Jaccard similarity coefficient. JI is a measure of the number of shared and distinct elements between two sets in Appendix Equation 4. Note that A and B are sets of random variables.

In context of domain name classification, domain name characters can be used to measure similarity between them. For example, given two domain names (ex, [google] and [go]), we can split these two domains by each character: set A = [*g, o, o, g, l, e*], B = [*g, o*]. In this example, $A \cap B$ = [*g, o*] and $A \cup B$ = [*g,o,o,g,l,e*]. *JI* = [*g, o*]/[*g, o, o, g, l, e*] = 2/6 = 0.33. JI can help us classify a group of domain names. For example, we can assume that two domain names with high *JI* are more likely to be in the same class than those who have a lower *JI*.

4.3. Proposed method: N-gram-based dictionary features

The methods described above, particularly K-L Divergence and Jaccard Index, defines a similarity score between domain name pairs. We extend this approach by measuring the similarity of domain names to words or phrases (instead of other domain names) that are common in one of the domain classes. For example, if we have a dictionary of common words for legit domains (let's call it legit domains dictionary), the next domain name has a greater chance of being legit if the name is similar to the elements in the legit to domains dictionary.

Our initial exploratory analyses suggest that dga domains are either created by using random characters or variations of semantically meaningful words that lead to incoherent names. These patterns are starkly different from the patterns in legit domain names. Based on these observations, we proceeded with generating dictionaries (key-value pairs for a set of character sequences and how often they appear in the dataset) for each domain name class: dga or legit names. Each of the two dictionaries is essentially a frequency vector for character sequences (of varying sizes) within the domain names of its class. We derived the dga dictionary from previously identified dga domain names and the legit dictionary from previously identified legit domains.

Next, we calculated the similarity of a domain name to each of the two dictionaries by first converting the domain name to a frequency vector of character sequences and then by multiplying its vector with the two of the class-dictionary vectors. This method gives us two measures for any domain name: the similarity to the dga domains and the similarity to legitimate domains. Appendix Algorithm 1 provides the pseudo-code for the process.

Next, with an example, we illustrate the two phases of similarity calculation in Appendix Algorithm 1: 1) constructing an N-gram frequency vector $F_{1,n}$, and 2) computing the similarity score *S*.

For this example, we constructed an N-gram counter vector for two legit domains, "facebook" and "booking." We computed the sequence of characters – N-grams, $L_{1,n}$ = [*face, aceb, cebo, eboo, book, ooki, okin, king*] when *N*, the maximum number of characters in a sequence, is 4. We then mapped a domain name onto this N-gram vector ($L_{1,n}$) by multiplying the elements with 0 if they don't match the domain name, and multiply with 1 if they match. The resulting vector captures whether a domain name is part of existing n-gram features. In the results vector, 1 indicates presence, and 0 indicates absence. Table 1 shows how we would find the domain name "book" within the N-grams derived from "facebook." We repeated this process for all the domains and accumulated the vectors, resulting in an N-gram frequency vector that shows the occurrences of N-gram elements in the evaluated domain names. Table 2 shows how we calculated the frequency vector ($F_{1,n}$) for two domain names: "facebook" and "booking."

The final step, calculating a similarity score, is simply a vector multiplication of the frequency vector $F_{1,n}$, as calculated in the previous phase, and the domain vector in question. In our illustrative example, the similarity of domain name "book" to the legitimate domains in our dataset ("book" and "facebook") can be computed as: $S = \log_{10}([0,0,0,0,1,0,0,0]\times[1,1,1,1,2,1,1,1]^T) = 0.301$. In a similar way, to calculate a similarity score with dictionary, we could replace the legit domain name with dictionary words.

Table 1. Calculate similarity score.

	$L_{1,n}$							
d	face	aceb	cebo	eboo	book	ooki	okin	king
book	0	0	0	0	1	0	0	0

N: the maximum number of characters in a sequence
$L_{1,n}$: n-gram sequences set built from dictionary domains
d: test domain

Table 2. N-gram counter vector for legit domain.

	$L_{1,n}$							
$D_{m,1}$	face	aceb	cebo	eboo	book	ooki	okin	king
facebook	1	1	1	1	1	0	0	0
booking	0	0	0	0	1	1	1	1
$F_{1,n}$	1	1	1	1	2	1	1	1

m: the number of observations
$D_{m,1}$: testing domains
$F_{1,n}$: the n-gram terms of frequency

4.3.1. Words embedding

An approach applied LSTM model to fit domain names into a deep learning network (Woodbridge et al., 2016). Instead of calculating features directly from textual data and feeding them into a neural network hidden layer, they added a word embedding layer mapping each character from domain names to vectors of numerical values.

Since characters in domain names are restricted to 26 lowercase characters, 10 numeric values (0–9), and 1 symbol (dash), we can represent domain name as a matrix of 37 by n where n is the length of the domain name. In our research, we represented domains with vectors of 64 (instead of 37) to cover domain name characters with paddings.

4.4. Data collection

To train our classification algorithms, we need to have labeled training data for both dga (positive) cases and legitimate (negative) cases. We have put together two different data sets for our experiments. For the first data set, we obtained our positive cases from domains generated from various algorithms used in existing botnets and then we obtained the negative cases from the Alexa list of top web sites. This data set is used for both training and testing. For the second data set, we extracted data from an independent source (Bader, 2016; Selvi et al., 2019) for the validation of our model.

4.4.1. Positive cases: DGA domains

In order to include positive cases that are domains generated from a variety of actual malware algorithms, we combined data from two sources of dga domains. The first source is a data repository of dga domain names, and the second source is dga domains that we generated using domain name generation algorithms from actual malware.

From a domain name repository, OSINT Feed, we collected & generated a list of domains on September 20, 2018 (Bambenek, 2019). It contains domain names with their high-level classification in dga. These categories can be further categorized into subclasses of 38 different DGA families from a research (Plohmann et al., 2016). Particularly, in those 38 DGA families, Suppoblox and Nymaim are malware that used dictionary to generate DGA domains. Besides, we collected domains from Matsnu, another malware that used English dictionary words to generate DGA domains. We used a cluster sampling technique on our raw dataset. Specifically, we stratified resampled data from each DGA family and extracted 50,033 entries in total. Details about our DGA sample summary can be found in Table 3.

4.4.2. Negative cases: legit domains

We collected legitimate domains from the Alexa list of top web sites. The Alexa Top Sites web service provides access to a list of web sites ordered by Alexa Traffic Rank (*Alexa Ranking, 2019*). Since legit domains are more likely to attract more traffic than dga domains, our intuition was that highly ranked Alexa domains can represent legit domains. To that end, we collected the top 50,000 unique domains from Alexa's list after a pre-processing step. The pre-processing involved removing the invalid URL and duplicated subdomains or domains. We limited the number of Alexa domains at 50,000 to have a balanced dataset considering we had selected roughly 50 thousand DGA domains via stratified sampling.

In summary, we created a data set of 100 K (100,033) domain names, of which 50 K (50,033) are positive cases (generated domains), and 50 K are negative (legit domains) cases.

Table 3. DGA family sample statistics.

Category	Count	% Category
Banjori	2000	4.0%
Bedep	313	0.6%
Chinad	1306	2.6%
conficker	2000	4.0%
Corebot	238	0.5%
CryptoLocker	2000	4.0%
DirCrypt	612	1.2%
Dyre	2000	4.0%
Fobber	510	1.0%
Geodo	490	1.0%
Hesperbot	214	0.4%
Kraken	2000	4.0%
Locky	2000	4.0%
Matsnu	2000	4.0%
Murofet	2000	4.0%
Mydoom	298	0.6%
Necurs	2000	4.0%
Nymaim	2000	4.0%
p2pgoz	2000	4.0%
padcrypt	490	1.0%
pizd	434	0.9%
proslikefan	663	1.3%
ptgoz	2000	4.0%
pushdo	1428	2.9%
pykspa	2000	4.0%
ramdo	2000	4.0%
ramnit	2000	4.0%
ranbyus	2000	4.0%
shifu	2000	4.0%
simda	2000	4.0%
sphinx	653	1.3%
suppobox	862	1.7%
tempedreve	212	0.4%
tinba	2000	4.0%
unknownjs	153	0.3%
vawtrak	2000	4.0%
vidro	224	0.4%
virut	510	1.0%
Volatile Cedar	423	0.8%
Grand Total	50033	100.0%

4.4.3. Out-of-sample dataset

As a second dataset, we used a previously collected dataset collected from a DGA database (Bader, 2016; Selvi et al., 2019). We refer to this data set as out-of-sample data set. The main goal of this dataset is to test the robustness of our model when come across new data. This data set contains 64 K domain names. We random sampling 60 K domains. 30 K were dga, and remaining 30 K were legitimate domains.

4.4.4. Dictionary

We used a dictionary that contains 479,623 commonly used words (Security, 2013), which are combinations of English vocabulary and commonly used words with a mix of numbers and alphabet. The dictionary is used for calculating the similarity score between domain and dictionary-words benchmark.

5. Experiments and results

N-gram based approaches require that we identify n – the maximum number of characters when extracting character sequences. We started by experimenting n = 3, 4, and 5 because these are numbers close to the average length of the domains. We will discuss how we determine in Discussion session.

We then trained and tested the classification methods using the features calculated from our data set. There are about 52 k legit and 50 k dga domains. We used feature scaling, 10-fold cross-validation, and 80/20 split techniques for our training set and testing set. Table 4 shows the distribution and statistics of features in training and testing set.

Table 4. Distribution of features in training/testing dataset.

	Training		Testing	
	mean	std	mean	std
length	12.54	6.32	12.61	6.36
entropy	3.02	0.48	3.03	0.48
KL_unigram	−2.94	3.99	−2.96	4.01
KL_bigram	−4.22	8.06	−4.25	8.14
alexa_bigram_JI	0.03	0.01	0.03	0.01
alexa_Ngrams	16.97	17.32	17.12	17.43
word_Ngrams	26.56	27.63	26.72	27.58

Table 5. Tuned hyperparameters in classifiers.

Classifier	Parameters
RF	n_estimators = 600
SVM	kernel = 'linear', C = 100
NB	N/A
XGB	learning_rate = 0.1, n_estimators = 500, kernel = tree
DNN	Hidden Layer1: num_of_neuro = 4, activation = relu Hidden Layer2: num_of_neuro = 8, activation = relu, learn_rate = 0, validation_split = 0.1, optimizer = SGD, epochs = 20, batch_size = 10
LSTM	dropout_rate = 0.5, activation = sigmoid embedding_num = 64, epoch = 20, batch_size = 64

Detail tuning results for hyper-parameters used for each classifier are in Table 5. We ran our experiments using Core i7 9700 3.0 GHz, 64GB RAM, GeForce RTX 2070, Ubuntu 18.04, Python 3, Scikit-learn, XGBoost, TensorFlow and Keras modules (Abadi et al., 2016; Chen & Guestrin, 2016; Chollet, 2015; Pedregosa et al., 2011).

5.1. Features selection

In our research, we have seven features calculated from domain names: length, entropy, K-L divergence (unigram and bigram), Jaccard index, n-gram-based dictionary (alexa_gram, word_gram). We proceeded with the feature selection process by comparing the contribution of performance in detection within feature combinations. We chose three different combinations of features. Table 6 shows the comparison of features combination. We selected feature combinations in a forward stepwise fashion by adding new features to the existing ones in each step.

Table 6. Feature selection combination.

Combination	Features
2 F	length + entropy
5 F	length + entropy+ KL_unigram + KL_bigram + JI bigram
7 F	length + entropy+ KL_unigram + KL_bigram + JI bigram + alexa_ngram + words_ngram

5.2. Evaluation metrics

We evaluated the classification accuracies of the models by two commonly used metrics: F1-score and Area Under the Curve (AUC). Our evaluation process involved two steps. First, we explored the optimal number of features and the classification method in a grid search fashion. Second, we compared the optimal result with the results from a deep learning model, LSTM, which commonly yields a higher classification accuracy but takes longer to train.

5.3. Features comparison and classifier comparison

Table 7 shows the F1-scores among all the classifiers with the feature combinations.

The results show that all of the models performed better with 7 features (7 F). Both the RF and XGB achieved the highest F1 score of 0.94. Table 8 shows a more detailed view of the classifier performances with seven features.

Random Forest (RF) and XGBoost (XGB) classifier performed best among all classifiers. SVM and DNN classifiers performed slightly worse but still better than Naïve Bayes. The overall results in error rates were also good for RF and XGB. FPR are 6.63% and 6.50% for RF and XGB, respectively. In cybersecurity, FPR measures the rate of mislabeled security alerts, indicating a threat but actually not. It is important to keep a low FPR in order to reduce the time and energy spent on the false positives. We also found that all of the TPRs were consistently higher than TNRs for every model. Higher TPRs suggest our models are better at identifying DGA

Table 7. F1-score in features comparison.

	2 F	5 F	7 F
RF	0.87	0.93	0.94
SVM	0.86	0.90	0.91
NB	0.84	0.84	0.85
XGB	0.90	0.93	0.94
DNN	0.86	0.91	0.93

Table 8. Classification results on seven features (7 F).

%	RF	SVM	NB	XGB	DNN	Majority Vote
TPR	94.74	94.27	92.69	94.67	93.56	95.03
FNR	5.26	5.73	7.31	5.33	6.44	4.97
FPR	6.63	11.66	21.59	6.50	7.16	6.85
TNR	93.37	88.34	78.41	93.50	92.84	93.15

domain names than identifying legit domain names.

There are many reasons to explain the differences between classifiers. RF uses a bootstrap dataset and takes many votes from a bunch of shallow decision trees to make a forest. XGB is also a gradient boosted algorithm using decision trees. Thus, tree-family classifiers can capture the specific features of textual data very effectively. However, linear SVM works by identifying a hyperplane that separates the classes in the data. Thus, when this classifier processes overlapping data, especially with multi-dimensional features, tree family models can outperform SVM models. The Naïve Bayes (NB) combines conditional probabilities with the underlying assumption that a feature is independent of the occurrence of other features. This assumption may not be a suitable one for our domain name dataset.

One of the advantages of traditional machine-learning algorithms over deep learning algorithms is that we can check the importance of the features based on the weights assigned to the features during classification. We used the permutation method to test the contribution weights within features (Breiman, 2001). In Table 9, it provides us a brief explanation of how this algorithm (classifier) treats the features. Our results suggest that both KL divergence and Alexa_gram play more important weights than the others. That is, the difference between two distribution groups (KL divergence) and the n-grams in similarity score between the test domain and legit domains (Alexa gram) help the machine to distinguish domain names. This is helpful for humans to interpret how machine-learning algorithms make decisions.

5.3.1. Compare with LSTM

Our analysis thus far suggests that our n-gram dictionary-based approach performs best when coupled with either Random Forest (RF) or XGBoost (XGB) classifiers. We compared these two models with a significantly more complex model that does not involve feature engineering but requires training neural networks, namely the LSTM model. LSTM also requires a word-embedding layer that maps the input data into a numerical matrix.

We selected our embedding length as 64. While there are only 36 + 1 = 37 unique characters in domain names, 64 is the smallest embedding size with a base of 2. Consequently, we set the maximum length of embedding as 64, and the maximum number of features is 38. The total number of input parameters, then, is 64 × 38 = 2,432. Table 10 shows the detail model summary. In total, LSTM trained 35,521 parameters.

The comparison of LSTM results to the previous results from Random Forest and XGBoost models is in Table 11. We found that F-1 scores from LSTM are comparable to the results from Random Forest, and XGB. The difference is 0.01. However, this improvement comes at a cost. The significantly higher number of parameters makes LSTM a computationally expensive process. To get a better understanding of the trade-off, we need to compare the computation time among different classifiers. We will test them in detail at the Time Comparison section.

Next, we take another commonly used metric into account to compare binary classifiers, namely the receiver operating characteristic curve (ROC). Figure 1 shows the ROC curve for different classifiers in the 7-features case and LSTM with 2432 features. According to our classification rates and AUC in Figure 1, we found that the LSTM classifier had the highest AUC value (0.99), indicating a better performance. Both RF and XGB achieved slightly lower, but comparative AUC values (two

Table 9. Permutation importance in random forest.

Weight	Feature
0.2627 ± 0.0022	KL_bigram
0.1589 ± 0.0021	alexa_Ngrams
0.1146 ± 0.0055	alexa_bigram_JI
0.0542 ± 0.0014	entropy
0.0344 ± 0.0021	KL_unigram
0.0095 ± 0.0007	word_Ngrams
0.0028 ± 0.0015	length

Table 10. LSTM model summary.

Layer	Output Shape	Param #
Embedding	(None, 53, 64)	2432
LSTM	(None, 64)	33024
Dropout	(None, 64)	0
Dense	(None, 1)	65
Activation	(None, 1)	0

Table 11. Model compare with LSTM.

	RF	XGB	LSTM	Majority Votes
F1-score	0.94	0.94	0.95	0.94

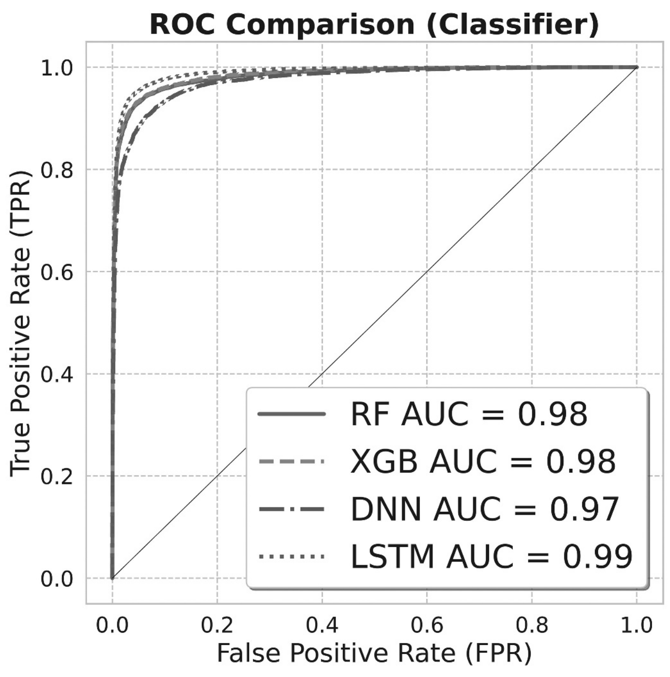

Figure 1. Classifiers ROC curve.

lines overlapped with the same AUC 0.98). All AUC results confirmed our inference in F1-score in Table 11.

Figure 2 shows the ROC comparison for feature selection using random forest classifier and LSTM features. Our result shows that LSTM has the highest AUC value (0.99), which is a support for our analysis above. AUC with 7-features is 0.98, slightly less than LSTM, but they are very competitive.

We also tested whether ensemble methods that combine the predictions of several estimators will

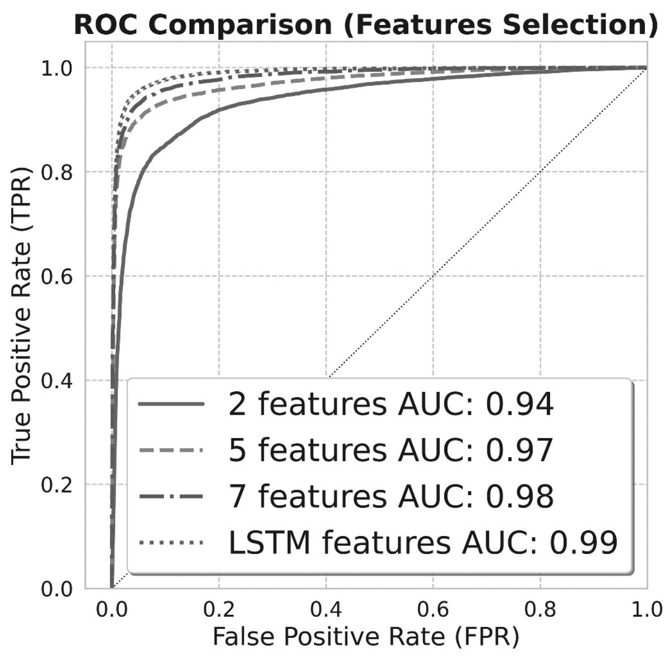

Figure 2. Features ROC curve.

improve generalizability and performance. We used majority votes that combines our best three classifiers using the seven features case. The results are shown in Table 8. We found that combining top-three classifiers will not improve the overall performance. Particularly, TPR is 95.03% and TNR is 93.15%. F-1 score is 0.94 which is the same as RF and XGB classifiers but slightly worse than the LSTM classifier. We also noticed that the ensemble method will decrease FNR (4.97%) but increase FPR (6.85%) compared with any of the single classifiers. Thus, for simplicity purpose, we choose three single classifiers in our research.

5.4. Time comparison

Our analysis shows that the LSTM model performed the best among all. However, since LSTM requires calculating 2432 input parameters from the embedding layer, it takes significantly more time to train the LSTM model than the other classifiers. Computation time is an important aspect of the detection process, especially where real-time detection for DNS servers is necessary. We performed 20 repeated simulations with the same testing environment. The results are in Table 12.

In Table 12, we compare the average time spent in feature preparation, model training, and model prediction. The feature preparing column shows the average time spent on calculating all of the features from 73,279 training data. These features are fed into our machine-learning models except for LSTM. The LSTM model determines its features and weights during the training stage. The testing time is calculated as the average time spent to label each of the 18,320 testing data. We found that NB is the fastest model in training and predicting because of its simple mechanism in probability calculation. In comparison, due to the complexity of backpropagation in deep learning, LSTM was the slowest in both training and predicting. The LSTM model can only handle approximately 18320/1.914 = 10 K DNS queries per second. However, according to a Domain Name System Operations Analysis and Research Center, a DNS flood attack could submit 20 to 50 K queries per second (Operators, 2015). The rapid nature of such attacks suggests that the LSTM model might be too slow to implement in real-world practice. Both RF and XGB have a moderate speed that allows them to fit 70 K data entries within only 30 seconds, making 18 K predictions in less than 0.3 seconds. The equivalent detection rate is 18320/0.304 = 60 K and 18320/0.213 = 86 K. These results suggest that RF and XGB models are strong candidates for DGA detection in real-world production.

Table 12. Average time spent comparison (seconds).

	Feature Preparing	Feature + Training	Predicting	Predict Rate (K/second)
RF		27.989	0.304	60
SVM		111.134	1.974	9
NB	11.361	11.377	0.002	9,160
XGB		26.069	0.213	86
DNN		99.299	0.135	136
LSTM	N/A	405.718	1.914	10

Table 13. Out-of-sample test.

	RF	XGB	LSTM
TPR	97.42	97.44	97.04
FNR	2.58	2.56	2.96
FPR	4.54	4.19	6.22
TNR	95.46	95.81	93.78

Table 14. Out-of-sample test.

	RF	XGB	LSTM
precision	0.97	0.97	0.95
recall	0.96	0.97	0.95
f1-score	0.96	0.97	0.95

5.5. Out-of-sample test

DGA attackers often try to deceive dga detection techniques by changing their dga strategies. Therefore, the dga detection algorithms need to be updated periodically. Also, to evade detection, attackers may try to know the features of the detection system and then change the type of dga generator such that features-based model will not be valid. Thus, in our research, we also test whether our model with a dataset that was collected in a different study (see Bader (2016) for the dataset details.)

Tables 13 and 14 show the results of our model using another dataset source. All classifiers perform well. Especially, the XGB classifier with our proposed method performs the best with 0.97 f1-score. TPR (97.44%) and TNR (95.81%) indicate our model is able to capture both new DGA domains and legit domains very well. FPR (4.19%) and FNR (2.56%) also indicate that our methods are also very robust and effective against new dataset.

6. Discussion

6.1. Features & classifier selection

Selecting useful features and classifiers involves careful consideration and, at times, goes beyond selecting the highest accuracy models in cybersecurity. Models that can handle trade-offs have tremendous practical significance. As our results suggest (Tables 8 and 12), the models that combine tree-based family classifiers, such as Random Forest (RF) and XGBoost (XGB) with our dictionary-based n-gram features, are the most effective among many options. Accordingly, they have high overall performance rates, very time efficient, and much easier in features interpretation.

6.2. Determine "N" in N-gram

N-gram models divide text strings into sub-strings of n-many consecutive characters. Therefore, N is an important parameter that will impact model performance. In this work, we use a grid-search method to find an optimal value of N. Our research indicates that choosing N = [3, 5] is the optimal range in N-gram methods. This finding is in line with a recent study suggesting that one million of the most popular websites averaged around five characters in their domain names (Gaebler, 2009).

6.3. Future work

Detection of dga domains can be used in monitoring real-time DNS traffic. As an anomaly indicator, it can help to protect the DNS servers in organizations. Besides, dga indicators can also contribute to the botnet detection efforts. A potential way to improve the existing work is integrating dictionary-based methods to more complex models such as neural networks. It would be exciting to know whether a specific classifier or feature works more effectively in a specific DGA-family when a reinforcement learning framework is applied. At last, our methods can be applied in a real-time monitoring system framework and some other areas such as the syntactic application in language identification, and news validation in social media (Wang et al., 2018).

7. Conclusion

In this paper, we developed N-gram-based dictionary features from Alexa and English dictionary to identify malicious domain names generated using domain-generated algorithms. We tested 39 DGA-family domain names with six different classifiers, including Random Forest, SVM, Naïve Bayes, XGBoost, DNN, and LSTM. We compared the results with other similar research, and our results indicate that introducing N-grams-based dictionary features in classification models can increase the performance results in DGA detection. Also, when comparing features and models with previous work, we considered the computation time for each model and feature selection. Notably, our experiment showed that using hybrid dictionary-based features with a tree family classifier, such as Random Forest and XGBoost, has the best trade-off in detection accuracy and time complexity. The results, we argue, can be very informative in practical applications of DNS server protection and classifier selection.

Furthermore, we used a grid-search method to find an optimal range of N for N-gram features. We think this work expands our understanding of how botnets can be detected through the domain names they automatically generate, and our findings can be deployed in real-world production to minimize the damage these bots can cause. At last, our work provides insights into the opportunities and the challenges in distinguishing the differences between machine-generated and human-generated information.

Funding

This work was supported by the National Science Foundation [1503760]; National Security Agency [H982307-17-1-0335].

ORCID

Tianyu Wang http://orcid.org/0000-0002-9244-8798

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … Isard, M. (2016). Tensorflow: A system for large-scale machine learning. Paper presented at the 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), Savannah, GA.

Alexa Ranking. (2019). Alexa. https://www.alexa.com/

Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou, N., & Dagon, D. (2011). Detecting malware domains at the upper DNS hierarchy. Paper presented at the USENIX security symposium, Berkeley, CA.

Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., & Dagon, D. (2012). From throw-away traffic to bots: Detecting the rise of DGA-based malware. Paper presented at the USENIX security symposium, Bellevue, WA.

Bader, J. (2016). *Domain generation algorithms (DGAs) of malware reimplement*. GitHub. https://github.com/baderj/domain_generation_algorithms

Bambenek, J. (2019). *OSINT feeds*. Bambenek Consulting. http://osint.bambenekconsulting.com/

Barthakur, P., Dahal, M., & Ghose, M. K. (2013). An efficient machine learning based classification scheme for detecting distributed command & control traffic of P2P botnets. *International Journal of Modern Education and Computer Science*, *5*(10), 9. https://doi.org/10.5815/ijmecs.2013.10.02

Bochkarev, V., Shevlyakova, A., & Solovyev, V. (2015). Average word length dynamics as indicator of cultural changes in society. Social Evolution & History, *14*(2), 153-175. https://www.sociostudies.org/journal/articles/364324/

Brailsford, T. J., & Faff, R. W. (1996). An evaluation of volatility forecasting techniques. *Journal of Banking & Finance*, *20*(3), 419–438. https://doi.org/10.1016/0378-4266(95)00015-1

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Paper presented at the Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, New York, NY.

Chollet, F. (2015). *Keras*, GitHub, https://github.com/fchollet/keras.

Crowe, J. (2017). Companies are wasting massive amounts of money on ineffective security solutions. TechRepublic. https://www.securityweek.com/false-positive-alerts-cost-organizations-13-million-year-report

Das, A., Shen, M.-Y., Shashanka, M., & Wang, J. (2017). Detection of exfiltration and tunneling over DNS. *Paper presented at the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Cancun, Mexico.

Dunham, K., & Melnick, J. (2008). *Malicious bots: An inside look into the cyber-criminal underground of the internet*. Auerbach Publications.

Fu, Y., Yu, L., Hambolu, O., Ozcelik, I., Husain, B., Sun, J., … Brooks, R. R. (2017). Stealthy domain generation algorithms. *IEEE Transactions on Information Forensics and Security*, *12*(6), 1430–1443. https://doi.org/10.1109/TIFS.2017.2668361

Gaebler, K. (2009). *Does Domain Length Matter*. Gaebler. http://www.gaebler.com/Domain-Length-Research.htm.

Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. *Paper presented at the USENIX security symposium*. San Jose, CA.

Gu, G., Zhang, J., & Lee, W. (2008). BotSniffer: Detecting botnet command and control channels in network traffic. *Paper presented at the NDSS*, San Diego, CA.

Holz, T., Steiner, M., Dahl, F., Biersack, E., & Freiling, F. C. (2008). Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. *LEET*, *8*(1), 1–9. https://dl.acm.org/doi/10.5555/1387709.1387718

Koh, J. J., & Rhodes, B. (2018). Inline detection of domain generation algorithms with context-sensitive word embeddings. *Paper presented at the 2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA.

Krebs, B. (2014). Operation Tovar'targets 'Gameover'ZeuS botnet, cryptolocker scourge. Krebs on Security, https://krebsonsecurity.com/2014/06/operation-tovar-targets-gameover-zeus-botnet-cryptolocker-scourge/

Krebs, B. (2016, September 21). KrebsOnSecurity hit with record DDoS. *KrebsOnSecurity*.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86. https://doi.org/10.1214/aoms/1177729694

Leder, F., & Werner, T. J. T. H. P. (2009). *Know your enemy: Containing conficker*. Honeynet, https://www.honeynet.org/papers/kye-kyt/know-your-enemy-containing-conficker/

Li, C., Jiang, W., & Zou, X. (2009). Botnet: Survey and case study. *Paper presented at the 2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, Kaohsiung, Taiwan.

Li, Z., & Liao, Q. (2014). Toward a monopoly botnet market. *Information Security Journal: A Global Perspective*, *23*(4–6), 159–171. https://doi.org/10.1080/19393555.2014.931488

Liang, G., He, W., Xu, C., Chen, L., & Zeng, J. (2015). Rumor identification in microblogging systems based on users' behavior. *IEEE Transactions on Computational Social Systems*, *2*(3), 99–108. https://doi.org/10.1109/TCSS.2016.2517458

Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. *Paper presented at the Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris, France.

McGrath, D. K., & Gupta, M. (2008). Behind phishing: An examination of phisher modi operandi. *Paper presented at the Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, San Francisco, California.

Mendez Mena, D., Papapanagiotou, I., & Yang, B. (2018). Internet of things: Survey on security. *Information*

Security Journal: A Global Perspective, *27*(3), 162–182. https://doi.org/10.1080/19393555.2018.1458258

Mimoso, M. (2014). *Matsnu botnet DGA discovers power of words*. Threatpost.https://threatpost.com/matsnu-botnet-dga-discovers-power-of-words/109426/

Mitchell, T. (1997). *Machine Learning* (Vol. 1). MacGraw-Hill Companies.

Mowbray, M., & Hagen, J. (2014). Finding domain-generation algorithms by looking at length distribution. *Paper presented at the 2014 IEEE international symposium on software reliability engineering workshops*, NW Washington, DC.

Negash, N., & Che, X. (2015). An overview of modern botnets. *Information Security Journal: A Global Perspective*, *24*(4–6), 127–132 https://doi.org/10.1080/19393555.2015.1075629

Nooribakhsh, M., & Mollamotalebi, M. (2020). A review on statistical approaches for anomaly detection in DDoS attacks. *Information Security Journal: A Global Perspective*, *29*(3), 1-16. https://doi.org/10.1080/19393555.2020.1717019

Operators, R. S. (2015). *Events of 2015-11-30*. Operators, Root Server https://root-servers.org/news/events-of-20151130.txt

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Dubourg, V. (2011). Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, *12*(October), 2825–2830. https://dl.acm.org/doi/10.5555/1953048.2078195.

Perdisci, R., Corona, I., Dagon, D., & Lee, W. (2009). Detecting malicious flux service networks through passive analysis of recursive dns traces. *Paper presented at Computer Security Applications Conference, 2009. (ACSAC'09)*, Honolulu, Hawaii.

Pereira, M., Coleman, S., Yu, B., DeCock, M., & Nascimento, A. (2018). Dictionary extraction and detection of algorithmically generated domain names in passive DNS traffic. *Paper presented at the International Symposium on Research in Attacks, Intrusions, and Defenses*, Heraklion, Greece.

Plohmann, D., Yakdan, K., Klatt, M., Bader, J., & Gerhards-Padilla, E. (2016). A comprehensive measurement study of domain generating malware. *Paper presented at the 25th {USENIX} Security Symposium ({USENIX} Security 16)*, Austin, TX.

Rodriguez, D. (2016). *The query volumes of mirai DGAs*, Cisco Umbrella. https://umbrella.cisco.com/blog/blog/2016/12/13/query-volumes-mirai-dgas/

Schiavoni, S., Maggi, F., Cavallaro, L., & Zanero, S. (2014). Phoenix: DGA-based botnet tracking and intelligence. *Paper presented at the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Egham, United Kingdom.

Security, C. (2013). *Data hacking project*. Click Security. http://supercowpowers.github.io/data_hacking/

Selvi, J., Rodríguez, R. J., & Soria-Olivas, E. (2019). Detection of algorithmically generated malicious domain names using masked N-grams. *Expert Systems with Applications*, *124*, 156–163. https://doi.org/10.1016/j.eswa.2019.01.050

Sivaguru, R., Choudhary, C., Yu, B., Tymchenko, V., Nascimento, A., & De Cock, M. (2018). An evaluation of DGA classifiers. *Paper presented at the 2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA.

Tong, V., & Nguyen, G. (2016). A method for detecting DGA botnet based on semantic and cluster analysis. *Paper presented at the Proceedings of the Seventh Symposium on Information and Communication Technology*, Ho Chi Minh City, Vietnam.

Tsvetkova, M., García-Gavilanes, R., Floridi, L., & Yasseri, T. (2017). Even good bots fight: The case of Wikipedia. *PloS One*, *12*(2), e0171774. https://doi.org/10.1371/journal.pone.0171774

Wang, T., Chen, L.-C., & Genc, Y. (2018). An N-gram-based approach for detecting social media spambots. *Paper presented at the 2018 Pre-ICIS SIGDSA Symposium*, San Francisco, CA.

Wei-wei, Z., & Qian, G. (2013). Detecting machine generated domain names based on morpheme features. *Paper presented at the International Workshop on Cloud Computing and Information Security (CCIS 2013)*, Shanghai, China.

Woodbridge, J., Anderson, H. S., Ahuja, A., & Grant, D. (2016). *Predicting domain generation algorithms with long short-term memory networks*. arXiv. CoRR, abs/1611.00791. http://arxiv.org/abs/1611.00791

Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., & Osipkov, I. (2008). Spamming botnets: Signatures and characteristics. *ACM SIGCOMM Computer Communication Review*, *38*(4), 171–182. https://doi.org/10.1145/1402946.1402979

Yadav, S., Reddy, A. K. K., Reddy, A., & Ranjan, S. (2010). Detecting algorithmically generated malicious domain names. *Paper presented at the Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, Melbourne, Australia.

Yadav, S., Reddy, A. K. K., Reddy, A. N., & Ranjan, S. (2012). Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. *IEEE/Acm Transactions on Networking*, *20*(5), 1663–1677. https://doi.org/10.1109/TNET.2012.2184552

Yu, B., Pan, J., Gray, D., Hu, J., Choudhary, C., Nascimento, A. C., & De Cock, M. (2019). Weakly supervised deep learning for the detection of domain generation algorithms. *IEEE Access*, *7*, 51542–51556. https://doi.org/10.1109/ACCESS.2019.2911522