An Evaluation of DGA Classifiers

Raaghavi Sivaguru*, Chhaya Choudhary* Bin Yu†, Vadym Tymchenko†,
Anderson Nascimento*, Martine De Cock*,‡
* *School of Engineering and Technology, University of Washington, Tacoma, USA*
{raaghavi, chhayc, andclay, mdecock}@uw.edu
† *Infoblox, Santa Clara/Tacoma, USA*, {biny,vtymchenko}@infoblox.com
‡ *Dept. of Appl. Math., Comp. Sc., and Statistics, Ghent University, Ghent, Belgium*, martine.decock@ugent.be

Abstract—**Domain Generation Algorithms (DGAs) are a popular technique used by contemporary malware for command-and-control (C&C) purposes. Such malware utilizes DGAs to create a set of domain names that, when resolved, provide information necessary to establish a link to a C&C server. Automated discovery of such domain names in real-time DNS traffic is critical for network security as it allows to detect infection, and, in some cases, take countermeasures to disrupt the communication and identify infected machines. Detection of the specific DGA malware family provides the administrator valuable information about the kind of infection and steps that need to be taken. In this paper we compare and evaluate machine learning methods that classify domain names as benign or DGA, and label the latter according to their malware family. Unlike previous work, we select data for test and training sets according to observation time and known seeds. This allows us to assess the robustness of the trained classifiers for detecting domains generated by the same families at a different time or when seeds change. Our study includes tree ensemble models based on human-engineered features and deep neural networks that learn features automatically from domain names. We find that all state-of-the-art classifiers are significantly better at catching domain names from malware families with a time-dependent seed compared to time-invariant DGAs. In addition, when applying the trained classifiers on a day of real traffic, we find that many domain names unjustifiably are flagged as malicious, thereby revealing the shortcomings of relying on a standard whitelist for training a production grade DGA detection system.**

Index Terms—**domain generation algorithms, malware, seed, deep learning, tree ensembles**

I. Introduction

Malware installed on infected computers often seeks to establish a communication channel with a command-and-control server (C&C), for instance to send stolen information to the malware designer (the botmaster) behind the C&C server, or to receive instructions, or a newer version of the malware to update itself with. Domain Generation Algorithms (DGAs) are commonly used to create such a communication channel between infected computers and the botmaster [1]. A DGA dynamically generates a list of domain names, for instance using a publicly available random seed such as the date or the weather forecasts. One of these domain names is registered by the botmaster. Each infected machine queries the domain names from the automatically generated list. Once such a query is successfully resolved, the infected machine has found the domain name registered by the botmaster, and communication can take place. When the registered malicious domain name is discovered by law enforcement and blacklisted, the malware on the infected botnet and the botmaster can simply restart the process by generating a new list of domain names.

There is a growing interest in machine learning (ML) models that can detect DGA domain names in real-time to prevent any C&C communication [2]–[5]. Such systems need to deal with domain names originating from a variety of DGA malware families. Some DGA families are time-dependent, meaning that they incorporate a time source such as the system time of the compromised host or the date field in a HTTP response in their seed [1] while others use a seed that does not depend on time. To be useful in practice, trained DGA classifiers need to be sufficiently robust to detect both *time-dependent* and *time-invariant* DGA families, even when these DGAs start generating domain names based on new seeds that were not seen during training time.

Being able to detect whether a domain name is malicious or not, purely based on the domain name string, can be thought of as a binary text classification problem. Unsurprisingly, existing work on the development of DGA classifiers has drawn inspiration from the field of natural language processing. This includes both methods that leverage human defined lexical features extracted from domain names [5], [6], as well as deep learning methods that learn important features automatically as part of the training process [3], [7], [8].

In addition to distinguishing DGA vs. non-DGA domain names, network administrators are interested in the multi-class classification problem of labeling malicious domains according to their malware family. Detection of the specific DGA malware family provides network administrators with additional information to validate the result and, furthermore, to trust the detection decision with higher confidence. From an ML perspective, DGA malware family classification is a challenging problem because there are many different families, some of which generate many more distinct domain names than others in daily traffic. This corresponds to a multiclass classification task with many different class labels and great class imbalance. The research on DGA malware family classification is still in an initial stage, with varying success, depending on the particular DGA families [3], [7], [9], [10].

The standard approach followed in the literature to train and evaluate ML models for DGA detection is to collect known benign domain names from a whitelist, known DGA domain names from a blacklist, and to randomly assign some of these domain names for training and others for testing, for instance in a 80%-20% stratified split, or using k-fold cross-validation. In practice, DGA algorithms change their seeds in an attempt to evade detection. To allow for a more reliable evaluation of the robustness of trained DGA classifiers, in this paper we therefore create training and testing datasets that are purposely split across seed boundaries, i.e. to avoid seed overlap between the data used for training on one hand, and the data used for validation on the other hand.

Using this data, which originates from a real-traffic stream of passive DNS data, we train and evaluate both kinds of state-of-the-art methods for DGA detection and DGA family classification, namely (1) tree ensemble models based on human engineered features extracted from the domain names, and (2) deep neural networks that learn features automatically. To the best of our knowledge, our work is the first such experimental evaluation of the robustness of DGA classifiers against seed changes observed in real-traffic data. We find that all state-of-the-art classifiers cope well with seed changes of time-dependent malware families, while being significantly less resilient against seed changes of malware families that do not depend on time.

The data used in our study is compromised of domain names from Alexa (whitelist) and Bambenek (blacklist) that were observed in real traffic. When we subsequently apply the trained classifiers to large batches of domain names observed in real traffic, we find that they flag many domain names unjustifiably – yet with great confidence – as malicious. This finding casts doubt on the practical usefulness of the typical whitelist/blacklist trained DGA classifiers that are presented in the literature as solutions for malware detection, in particular whether a whitelist such as Alexa is sufficiently representative of all non-malicious domain names that appear in real traffic. To the best of our knowledge no such analyses were ever presented in the literature.

This paper is structured as follows: after giving an overview of related work on ML methods for DGA classification and DGA malware family detection in Section II, in Section III we provide details about the raw data collected for this study. In Section IV, we describe how we split this data into training and testing sets according to time and seed. In addition, we describe the binary and multiclass classification tasks, and provide a brief justification of the evaluation metrics employed later in the paper. Section V contains a description of the tree-ensemble and deep learning methods used to train the DGA classifiers. Detailed results are presented and analyzed in Section VI, where we highlight the difference in the ability of the classifiers to catch time-dependent and time-invariant DGAs. Finally, in Section VII we present and discuss our findings when applying the trained classifiers to large batches of domain names observed in real traffic.

II. Related Work

ML for DGA detection. A variety of machine learning (ML) approaches for DGA detection have been proposed over the last few years. A useful way to distinguish them is based on the kind of input they require when deployed for DGA detection. There are for instance techniques that retrospectively analyze entire groups of domains extracted from DNS queries that occurred in a certain time window [2], [11] vs. techniques that can classify individual domain names in real-time [3], [5]. There are ML models that only expect the domain name string itself [3], [4], [8] as input vs. ML models that exploit additional context features such as the IP-addresses that the domains are mapped to, or temporal access patterns (e.g. how often the domain was requested, and when) [6], [11]–[13]. Our focus in this paper is on techniques that can detect DGA domains in real-time based purely on the domain name string.

Real-time DGA detection based on domain name string. ML approaches that leverage the domain name string for DGA detection can be categorized into two groups: so-called "featureful" methods that rely on human defined lexical features extracted from the domain names, such as domain name length, vowel-character ratio, bigrams, etc. [2], [5], [6] and "featureless" methods in which the automatic discovery of good features is part of the overall ML model training process, as a form of representation learning [3], [14]. Popular kinds of classifiers used in the featureful approach for DGA detection are logistic regression and tree ensemble methods, while the featureless approach relies on the use of deep neural networks, namely Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNN). Most papers about the featureless approach include a featureful approach as a baseline method [3], [7], [8], [10], [14], and the featureless approach is typically reported to yield better, more accurate results. A note of caution is that in supervised learning in general, the predictive performance of feature based methods heavily relies on the choice of features, and that authors who want to highlight the benefits of featureless, deep learning approaches, might not necessarily go out of their way to carefully select and craft features to strengthen the featureful baseline approach. The fact that featureless approaches do not require such a labor-intensive process of feature engineering, is of course a major advantage of deep learning methods.

Malware family classification. Most of the work on DGA detection is concerned with distinguishing DGA traffic from non-DGA traffic, often treated as a binary classification task. In this paper, we are interested in identifying the malware family that generated the domain name as well, which is a multiclass classification task. In ML, there are two main approaches for multiclass classification. In direct multiclass classification, a classifier is trained to select the proper target label for a new instance. In One-versus-All (OVA) classification, a binary classifier is trained per target label, and new instances are assigned the label of the winning classifier, i.e. the classifier with the highest confidence for the instance at hand. Both techniques have been used already for DGA family classification, often including non-DGA domain names as an additional "benign family".

The earliest attempt, to the best of our knowledge, is the proposal of Antonakakis et al. [2] to train an HMM per target family, i.e. OVA-HMM. This is a featureless approach, since the HMMs consume each domain name as a sequence of characters. This OVA-HMM approach was evaluated on 4 malware families and the benign family in [2] and was later reported to be significantly outperformed by other methods (see below) [3], [10].

Woodbridge et al. [3] were the first to propose a deep neural network approach for DGA detection and family classification. They trained an LSTM network on data from the Alexa[1] whitelist and DGA domain names from the Bambenek Consulting[2] blacklist, including all DGA malware families considered in this paper, except for *locky*. They extended their LSTM network for binary classification (DGA vs. non-DGA) to an LSTM network with a softmax layer for direct multiclass classification (which DGA family), and compared the latter with an OVA-RF (One-versus-All Random Forest) approach, including Alexa as a "benign family". They observed that the LSTM approach outperformed the OVA-RF approach for all families, yet still failed to identify some of the families correctly. Plausible reasons given for this were that the representation of some families in the dataset was too small to be able to do meaningful learning, and that some families (like *Cryptolocker*) were easily mistaken by the classifier for a similar family (like *ramnit*). As a workaround, Woodbridge et al. [3] therefore proposed to train a classifier that assigns domain names to superfamilies, effectively making the multiclass classification problem easier, and resulting in higher predictive accuracy scores. Anderson et al. [15] further extended the approach from Woodbridge et al. [3] by using a character-based generative adversarial network (GAN) to augment training sets in order to harden other ML models (like RF) against yet-to-be-observed DGAs.

Lison and Mavroeidis [7] followed up with a similar LSTM approach, trained and tested on a larger dataset consisting of domain names collected from the whitelists Alexa, Statvoo, and Cisco, and DGA domain names collected from the DGArchive[3], the Bambenek Consulting feeds, and by running reverse engineered domain generators. Their observations are along the same lines as Woodbridge et al. [3], i.e. the trained LSTM network does a good job in identifying some DGA families while performing poorly for others.

Refining the earlier work of Woodbridge et al. [3] further, most recently Tran et al. [10] proposed the use of a cost-sensitive learning algorithm to train an LSTM network for DGA family classification that takes class imbalances into account. In addition, they use a hierarchical classifier architecture: in a first step, domain names are classified as DGA or non-DGA by an LSTM trained for binary classification, and in a second step, domains that were labeled as DGA, are further assigned a family label with an LSTM trained for multiclass classification.[4] Tran et al. [10] trained and evaluated their approach on data from Alexa and Bambenek. Their most important observation is that their approach allows them to achieve a macro-average F1-score that is substantially higher than that of previous approaches, including the earlier work of Woodbridge et al. [3], because their trained model performs better for families with a limited representation in the data, which is something that Woodbridge et al. [3] struggled with. Still, there are families that are not correctly identified by Tran et al.'s approach at all [10], such as the family *locky* which was not included in Woodbridge et al.'s original work [3]. Lison and Mavroeidis [7] reported reasonable results for *locky*, which could be due to the use of a higher number of training examples for this particular family than Tran et al. [10].

Also recently, Choudhary et al. [9] obtained first place for a DGA family classification challenge in the DMD2018 competition[5] with an OVA-RF approach, i.e. a model consisting of binary RF classifiers, namely one per target family. Their success in the competition might have been due to the use of an additional, external dataset for training, instead of a specific choice of features or machine learning algorithm.

Contrary to the work we present in this paper, in all studies on malware family classification mentioned above [3], [7], [9], [10], the data was split in datasets used for training and testing without regard for time or seed.

TABLE I
NUMBER OF UNIQUE DOMAIN NAMES LISTED PER MALWARE FAMILY IN THE BAMBENEK DGA DOMAIN FEED FOR JULY 19, 2018, AS WELL AS HOW MANY, AND HOW OFTEN, WE OBSERVED THEM IN REAL TRAFFIC ON JULY 19, 2018. THE BOTTOM ROW PRESENTS SIMILAR INFORMATION, BASED ON THE ALEXA TOP 1 MILLION DOMAIN NAMES INSTEAD OF ON THE BAMBENEK DGA DOMAIN FEED.

Family	Bambenek (unique)	Real-traffic (unique)	Real-traffic (total)
Cryptolocker-Flashback	6,000	2,570	54,912
dyre	7,998	1,813	36,610
locky	5,352	5,161	132,763
murofet	26,520	22,974	485,730
necurs	28,672	28,532	1,078,489
nymaim	6,000	5,936	168,600
Post Tovar GOZ	66,000	32,571	455,993
pykspa	14,215	14,215	1,815,591
qakbot	40,000	34,624	752,489
ranbyus	13,640	13,323	277,764
banjori	439,223	439,206	7,939,787
tinba	66,688	54,352	445,549
ramnit	56,174	56,138	1,793,601
simda	14,755	14,729	581,788
shiotob/urlzone/bebloh	12,521	12,517	399,143
Total	803,758	738,661	16,418,809
Alexa	1,000,000	793,936	128,292,104

III. Raw Data

The data used in this study originates from a real-time stream of passive DNS data. It consists of roughly 10 billion DNS queries per day collected from multiple ISPs (Internet Service Providers), schools and businesses distributed all over the world. We collected 7 days of traffic, for the following dates: Jul 19, 20, 22, 23 24, 25, and Aug 06, 2018. The time

[1]https://www.alexa.com/topsites
[2]http://osint.bambenekconsulting.com/feeds/
[3]https://dgarchive.caad.fkie.fraunhofer.de/
[4]https://github.com/bkcs-hust/lstm-mi
[5]http://nlp.amrita.edu/DMD2018/

gap between the 6th and the 7th day is intentional, as will become more clear in Section IV.

Out of all the collected traffic, we keep only valid (query and response available) DNS queries of type A and AAAA (IPv4 and IPv6 address records) with response code 0 (SUCCESS) and 3 (NXDOMAIN). Each domain in our dataset consists of a second-level domain (SLD, e.g. *google*) and a top-level domain (TLD, e.g. *com*), separated by a dot. We observed between 50 to 70 million such unique domain names per day in real traffic.

To obtain ground truth labels for the real-traffic data, we matched it with the DGA domain feed from Bambenek Consulting collected for the same days in July and August 2018. This DGA domain feed is generated on a daily basis with reverse engineered DGA malware of 50+ known families. For each family, the feed contains domain names that would have been generated on that day by the DGA algorithm. In Table I we show, as an example, statistics for the 15 DGA families from Bambenek that we observed most frequently in real traffic. The first column shows the DGA malware family name, while the second column indicates the number of distinct domain names that would have been generated by the malware on July 19, 2018 according to the DGA domain feed of Bambenek consulting. Next, we show how many of those we observed in real traffic (unique count) and how often (total count). Note that the same domain name can get queried multiple times, possibly with different subdomains, which explains why the numbers in the last two columns of Table I vary.

The bottom row in Table I is different from the other rows. It is based on the whitelist Alexa. Alexa ranks websites based on their popularity in terms of number of page views and number of unique visitors. For example, according to Alexa, the three highest ranked domain names in terms of popularity in Aug 2018 were *google.com*, *youtube.com*, and *facebook.com*. For the purposes of this study, we assume that the domain names in the Alexa top 1 million list are not malicious. The bottom row in Table I indicates how many of the top 1 million Alexa domain names occurred in the real traffic data on July 19, and how frequently they were requested.

Finally, we mention that the number of Bambenek domain names observed in real traffic fluctuates somewhat from day to day. As Figure 1 shows, 738,661 Bambenek domains for Jul 19 were also seen in the real traffic data on that day. Similarly, we observed 737,927, 737,462, 722,077, 722,345, 736,728 and 720,192 Bambenek domains for the days of Jul 20, 22, 23, 24, 25, Aug 06 in real traffic for the same days.

IV. Problem Description and Evaluation Metrics

We train ML models that can detect DGA domain names in real traffic, based purely on the domain name string, and say which malware family they belong to, without the need to access reverse engineered malware. We evaluate the trained models on a ground truth labeled dataset (described below) as well as on an entire day of real traffic (see Section VII).

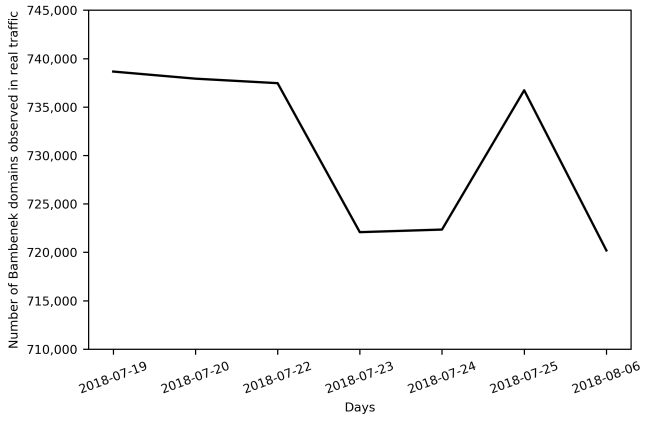

Fig. 1. Number of Bambenek domains observed in real traffic domains on the train and test days

TABLE II
Number of identified domain names observed in real traffic on Day 1 through Day 6 on one hand, and on Day 7 on the other hand. Despite the time gap of 12 days that occurred in practice between Day 6 and Day 7, there is still substantial overlap between Day1-6 on one hand, and Day7 on the other hand, as indicated in the intersection column Day1-6 $\cap$ Day7.

	Family	Day1-6	Day7	Day1-6 $\cap$ Day7
TD	**Cryptolocker-Flashback**	7,469	2,609	1
	dyre	7,766	2,181	3
	locky	9,234	3,883	1
	murofet	49,284	22,383	1
	necurs	58,799	28,684	1,068
	nymaim	12,496	6,048	439
	Post Tovar GOZ	98,873	32,851	4
	pykspa	19,964	14,261	10,188
	qakbot	38,446	18,184	3
	ranbyus	14,367	13,806	10,337
	Total	316,698	144,890	22,045
TI	**banjori**	439,207	439,205	439,205
	tinba	59,564	53,995	53,952
	ramnit	56,138	56,140	56,138
	simda	14,729	14,730	14,729
	shiotob/urlzone/bebloh	12,521	12,519	12,519
	Total	582,159	576,589	576,543
	Alexa	884,752	792,278	787,718

Below, we use Day1-6 to refer to the dataset with all domain names from Alexa and Bambenek that were observed in real traffic on Jul 19, 20, 22, 23 24, 25, while Day7 contains the Alexa and Bambenek domain names that were observed in real traffic on Aug 06. Table II lists the number of domain names per family, including Alexa, in each of these datasets.

We observed a good amount of overlap between both datasets; column "Day1-6 $\cap$ Day7" in Table II contains the number of domain names that occur in both datasets, i.e. domain names that were requested on Day 7 (Aug 06, 2018) as well as on at least one of the six days in July 2018. The existence of such overlap correlates with the nature of the DGA algorithms. Indeed, some DGA algorithms are time-dependent, meaning that they incorporate a time source such as the system time of the compromised host or the date field

TABLE III
NUMBER OF DOMAIN NAMES IN TRAIN AND TEST DATA

	Family	Train	Test	Train ∩ Test
TD	**Cryptolocker-Flashback**	7,469	2,609	1
TD	**dyre**	7,766	2,181	3
TD	**locky**	9,234	3,883	1
TD	**murofet**	49,284	22,383	1
TD	**necurs**	58,799	28,684	1,068
TD	**nymaim**	12,496	6,048	439
TD	**Post Tovar GOZ**	98,873	32,851	4
TD	**pykspa**	19,964	14,261	10,188
TD	**qakbot**	38,446	18,184	3
TD	**ranbyus**	14,367	13,806	10,337
	Total	316,698	144,890	22,045
TI	**banjori**	24,997	7,000	0
TI	**tinba**	42,212	12,882	2,093
TI	**ramnit**	9,907	2,639	0
TI	**simda**	4,006	2,008	22
TI	**shiotob/urlzone/bebloh**	6,002	4,002	1,997
	Total	87,124	28,531	4,112
	Alexa	884,752	792,278	787,718

in a HTTP response in their seed [1] while others use a seed that does not depend on time.

Whether a family is *time-dependent* (TD) or *time-invariant* (TI), is indicated in Table II. For most of the time-dependent DGAs, the overlap between Day1-6 and Day7 is moderate to almost none. These DGAs generate a large number of completely fresh domain names on a daily basis. As expected, the overlap between Day1-6 and Day7 for the time-independent domain names is a lot higher, with barely any new domain names appearing on Day7 that had not already been used by the DGA in Day1-6. Note that even for time-invariant DGA algorithms, blacklisting all domain names is not an adequate defense strategy because the amount of domain names can be very large (too large to check in real time) and once the seed is changed, the blacklist would become useless.

To generate appropriate train and test datasets (cfr. Table III) for our classifiers we adopt the following strategy:

- **TD-DGA.** For the time-dependent DGA families, we take all the unique domains from Day1-6 as training data, and the domain names from Day7 as test data. As Table III indicates, this means that for some of the TD-DGA families, there is a moderate amount of overlap between the train and test data.
- **TI-DGA.** For the time-independent DGA families, we retain all domain names from Day1-6 and Day7 that we were able to trace back to a specific seed using seed information from the DGArchive. The total number of such seeds and corresponding domain names is indicated in Table IV, and varies per family. All these seeds were active both in Day1-6 and in Day7, so instead of following a time-based split in train and test data as for the TD-DGAs, we split the TI-DGA domain names according to seed. To this end, we randomly select 80% of the seeds of each TI-DGA family for training purposes, and keep the remainder for test purposes. The number of seeds and corresponding domain names in the train and test data is shown for each TI-DGA family in Table IV.
- **TI-Alexa.** We use all Alexa domain names that were observed in real traffic on Day1-6 as training data, and all Alexa domain names that were observed in real traffic on Day7 as test data.

For each domain name in the test data, a trained ML model should infer whether the domain name belongs to any of the malware families from Table III, and if so, say which one. Regardless of the ML method used, the first step can be evaluated as a *binary classification problem* while the second step is a *multiclass classification problem*.

Given that blocking legitimate traffic is highly undesirable, a low false positive rate is very important in deployed DGA detection systems. For this reason, for the binary classification task, we evaluate the ML models in terms of the true positive rate (TPR) and false positive rate (FPR). These are defined as usual as TPR = TP/(TP+FN) and FPR = FP/(FP+TN) where TP, FP, TN, and FN are the number of true positives, false positives, true negatives, and false negatives respectively. ML algorithms commonly include hyperparameters that can be tuned to vary the TPR and FPR, resulting in a so-called ROC curve of (FPR,TPR) pairs. In Section VI we also report the AUC-score, which is calculated as the integral of the ROC curve, and independent of any specific choice of threshold for the FPR.

For the multiclass classification task of detecting the correct malware family, we evaluate the ML models in terms of precision, recall, and F1-score per family, and we provide aggregate results in terms of (weighted) macro-average. Precision is defined as TP/(TP+FP), recall is the same as TPR, and the F1-score is the harmonic mean of precision and recall. The macro-average is an unweighted average of the scores per family, while the weighted macro-average takes the class sizes into account, thereby giving more importance to families which have many instances in real traffic.

V. METHODS

A. Feature Based Approach

From each domain name, we extract lexical/linguistic features, many of which are well known in the literature on DGA detection. We extract:

- The following 11 features used by Yu at al. [14]: ent (normalized entropy of characters); nl2 (median of 2-gram); nl3 (median of 3-gram); naz (symbol character ratio); hex (hex character ratio); vwl (vowel character ratio); len (domain label length); gni (gini index of characters); cer (classification error of characters); tld (top level domain hash); dgt (first character digit).
- The following 3 features proposed by Schüppen et al. [5]: ratio of consecutive consonants; ratio of consecutive digits; ratio of repeated characters.
- Length of TLD (tld_len): The number of characters in the TLD.
- Length of SLD (sld_len) [2]: The number of characters in the SLD.

TABLE IV
TRAIN AND TEST SEEDS AND UNIQUE DOMAIN NAMES PER TI-DGA FAMILY

	Family	No. of seeds			No. of domains		
		Train	Test	Total	Train	Test	Total
TI	**banjori**	25	7	32	24,997	7,000	31,997
	tinba	112	28	140	42,212	12,882	55,094
	ramnit	50	13	63	9,907	2,639	12,546
	simda	12	3	15	4,006	2,008	6,014
	shiotob/urlzone/bebloh	4	2	6	6,002	4,002	10,004

- Consonant Ratio (con): The number of consonants in the SLD divided by the length of the SLD.
- Digit Ratio (dig): The number of digits in the SLD divided by the length of SLD.
- 2-gram Circular Median (2gram_cmed): The SLD of the domain is duplicated and concatenated tail to head (e.g. "apple.com" becomes "appleapple") and subsequently the 2-gram median (i.e. the nl2 feature mentioned in [14]) for the resulting string is computed.
- 3-gram Circular Median (3gram_cmed): The SLD of the domain is duplicated and concatenated tail to head and subsequently the 3-gram median (i.e. the nl3 feature mentioned in [14]) for the resulting string is computed.
- Number of Unique Characters in domain (uni_domain): The number of unique characters in domain name string (including SLD and TLD, excluding '.' and '-').
- Number of Unique Characters in SLD (uni_sld): The number of unique characters in the SLD (excluding '.' and '-').
- Number of Tokens in SLD (tokens_sld): The number of tokens in the SLD. A token is a sequence of characters separated by "-". For example, the tokens_sld value of the domain "youtube-mp3.org" is 2.
- Number of digits in SLD (digits_sld): The number of numerical characters in the SLD.
- Longest Consonant Sequence in SLD (lng_con_seq) [16]: The length of the longest consonant sequence in the SLD of the domain. For example, the longest consonant sequence for the domain "google.com" is "gl", hence the value of lng_con_seq is 2.
- Indication Malicious (flag_dga): Boolean flag (0 or 1) that indicates if the domain contains any of the following TLDs that are known to be frequently associated with malicious activity[6]: "study", "party", "click", "top", "gdn", "gq", "asia", "cricket", "biz", "cf". For example, if the domain is "fff.cf", the value of this feature would be 1.

Several of these features are clearly correlated, such as the length of the TLD (tld_len), the length of the SLD (sld_len) and the overall domain name length (len). Since we use the features to train Random Forests (RFs), and the underlying decision tree learning algorithm has a built-in mechanism for good feature selection, we do not perform feature selection a priori.

Our choice for RFs is motivated by the fact that for supervised learning, tree ensemble methods (such as RFs) are among the most common algorithms of choice for data scientists because of their general applicability and their state-of-the-art performance. In addition to resulting in models with good predictive accuracy, the RF training algorithm also scales well to large dataset sizes, which helps to explain its popularity for DGA detection in particular (see e.g. [3], [5], [10], [14]).

We train several kinds of RF models:

- **B-RF**: A binary RF classifier with 100 trees, each tree being trained on a bootstrap sub-sample with a maximum of 20 features and by using entropy as the criterion to select splitting attributes. This classifier is trained on the train data from Table III, with the Alexa domain names labeled non-DGA (represented as label 0) and all other domain names labeled DGA (represented as label 1).
- **M-RF**: A multiclass RF classifier with the same parameter values used in B-RF is adopted for the multiclass classification problem of assigning a domain name to one of 16 classes from Table III, where Alexa is treated as "the benign family".
- **OVA-RF**: A One-versus-All RF classifier consisting of 15 binary RFs, namely one per DGA family from Table III, with the same hyperparameter values as above. Each RF is trained on a dataset that is designed to be balanced with 50% domains belonging to the target family and 50% domains belonging to the other families, including Alexa, in a stratified mix. Once the individual training datasets are prepared, the corresponding 15 binary RF classifiers are trained to identify if the domain belongs to the respective DGA family or not. To deploy the OVA-RF classifier, we directly pass new domains to the B-RF classifier to categorize between DGA and non-DGA domains. Only the domains that are labeled as non-DGA by the B-RF classifier are then passed to each of the 15 binary RF classifiers to perform multiclass classification. Each RF outputs a probability that the new domain belongs to its family. The classifier that outputs the highest probability, indicating that the domain belongs to its family, is taken as the final prediction.

B. *Featureless Approach*

In the feature-based approach described above, expert-defined features are first extracted from the domain names and subsequently used in feature vectors for training and deploying RFs. In contrast, in a featureless approach, the domain name string is passed directly as a sequence of characters as input to the classifiers, which during training automatically learn to extract useful features.

In the featureless approach, each domain name string is first converted to lowercase and then represented as a sequence

[6]https://www.spamhaus.org/statistics/tlds/, Accessed: 2018-07-18

of ASCII values corresponding to its characters. Following Woodbridge et al. [3], we set the maximum length at 75 characters. While domains can technically be longer – the maximum allowed length for SLDs and TLDs is 63 characters each – in practice they are typically shorter. If a domain name has less than 75 characters, we pad with zeroes on the left. If it has more than 75 characters, then we truncate the domain name by removing characters from the right side of the SLD until the desired length is reached. We train a variety of neural networks:

- **B-Endgame**: The B-Endgame classifier is a neural network for binary classification (DGA vs. non-DGA) consisting of an embedding layer, an LSTM layer, and a single node output layer with sigmoid activation, proposed originally by Woodbridge et al. [3].
- **M-Endgame**: An adaptation of the above with an output layer with 16 nodes and "softmax" as the activation function to ensure that the output values are in the range between 0 and 1 and can be used as predicted probabilities. The output value with the largest probability is taken as the final class predicted by the model. This model is used for malware family classification.
- **B-CMU**: The B-CMU classifier is a bidirectional recurrent neural network (RNN) used for binary classification (DGA vs. non-DGA) which consists of an embedding layer, a forward LSTM layer and a backward LSTM layer. In the forward LSTM layer, the input sequence is processed from the left to the right, as in a traditional RNN, while in the backward layer, the processing happens from the right to the left. The output from the forward and the backward layer is then combined and passed on to further layers. We use the same B-CMU classifier architecture as Yu et al. [8], who adapted the bidirectional LSTM that was originally proposed for tweet classification by Dhingra et al. [17] to the problem of DGA detection.
- **B-MIT**: This B-MIT classifier is a hybrid neural network consisting of an embedding layer, a CNN layer with 128 filters and 'ReLU' as the activation function, followed by an LSTM layer with 64 LSTM cells. We use the same B-MIT classifier architecture as Yu et al. [8], who adapted the hybrid neural network architecture that was originally proposed for learning tweet embeddings by Vosoughi et al. [18] to the problem of DGA detection.
- **M-CMU and M-MIT**: We have adapted the B-CMU and B-MIT model for the multiclass classification task of malware family detection by replacing the original output layer by a layer with 16 nodes and "softmax" as the activation function. Similarly as in the M-Endgame model, the output with the largest probability is taken as the final class predicted by the classifier.
- **B-LSTM.MI** and **M-LSTM.MI** are deep learning models with a similar architecture as B-Endgame and M-Endgame. The main distinction is that the LSTM.MI models are trained with a cost-sensitive learning algorithm that takes class imbalances into account, as proposed by Tran et al. [10]. Another distinction is that, while the B-LSTM.MI model is trained for binary classification (DGA vs. non-DGA) just like the B-Endgame model, the M-LSTM.MI model is trained to output one of the 15 malware family labels. This is different from the M-Endgame model which is trained to output one of 16 labels, i.e. with the benign family included. During deployment of the LSTM.MI approach for malware family classification, a domain name is first classified as DGA or non-DGA by the trained B-LSTM.MI model; domain names that received the DGA label are further classified according to their malware family by the M-LSTM.MI model.

All neural networks above were trained on a workstation with an NVIDIA Titan Xp GPU and 12 GB RAM. We used early stopping as the mechanism to select the number of epochs (iterations) used for training: to prevent overfitting, we stopped training when the validation loss (measured over a validation set that was split off for this purpose from the training data) was no longer improving. The LSTM.MI models were the fastest to convergence, after 10 epochs. The B-Endgame, B-CMU, and B-MIT models took an average of 20 epochs, while the M-Endgame and M-MIT models ran for 31 epochs. The M-CMU model took the largest number of epochs to train, with 54 epochs in total.

VI. Results

A. Binary Classification Results

All binary classifiers in this study output a probability that a given instance belongs to the positive class, so we can tune a threshold probability at which to consider a prediction positive. For each model in Table V, we chose the threshold that results in a 0.001 FPR over the test data, and we report the corresponding TPR. A 0.001 FPR means that we allow for no more than 792 of the 792,278 benign domain names in the test data to be misclassified as malicious. The corresponding TPR indicates what percentage of the malicious domain names are caught by the classifier. In addition to the standard TPR, we also report what percentage of the time-dependent (TD-DGA) and what percentage of the time-invariant (TI-DGA) domain names from the test data were caught by the classifier. These numbers are provided in the columns TPR-TD and TPR-TI respectively. Finally, we report the AUC-score, which is independent of any classification threshold choice.

Table V contains the results of the binary classifiers from Section V when trained on the train data and evaluated on the test data from Table III. As can be seen in the Table V, all classifiers are able to flag the time-dependent DGA domains with a high TPR (recall). On the other hand, we also see that the time-invariant DGA domains are difficult to catch by the classifiers, resulting in a lower TPR. Another observation is that the results obtained by Deep Neural Networks (DNNs) are better than the ones obtained with RF. Indeed, all the DNN models in Table V have a comparable performance which is significantly higher than that of the RF model.

TABLE V
BINARY CLASSIFICATION RESULTS IN TERMS OF AUC, FPR (FALSE POSITIVE RATE), TPR (TRUE POSITIVE RATE) = RECALL, TPR-TD (TPR FOR TIME-DEPENDENT DOMAINS FROM TEST DATA), TPR-TI (TPR FOR TIME-INVARIANT DOMAINS FROM TEST DATA)

Model	AUC	FPR	TPR	TPR-TD	TPR-TI
B-RF	0.9366	0.001	0.8843	0.9199	0.6391
B-Endgame	0.9897	0.001	0.9469	0.9753	0.8026
B-CMU	0.9912	0.001	0.9432	0.9744	0.7852
B-MIT	0.9974	0.001	0.9478	0.9741	0.8145
B-LSTM.MI	0.9967	0.001	0.9524	0.9792	0.7613

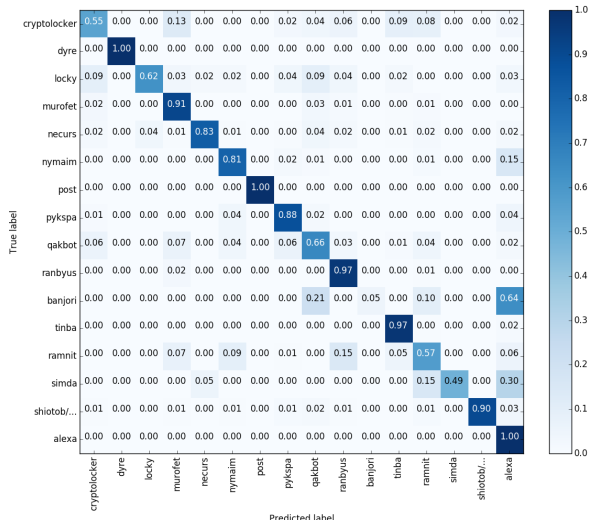

Fig. 2. Normalized confusion matrix for M-LSTM.MI

B. Multiclass Classification Results

The results obtained with the multiclass classifiers from Section V on the data from Table III are presented in Table VI and VII, with the best results in terms of F1-score for each family highlighted. It is immediately obvious from the tables that the RF classifiers are outperformed by the DNN classifiers. Furthermore, the performance of the M-RF and OVA-RF approaches is very similar, offering no strong reason to prefer one over the other. In comparison with the DNN classifiers, the RF classifiers do particularly poorly on the families *nymaim* and *banjori*.

Out of all DNN methods M-LSTM.MI has the highest macro-average F1 (0.7784), with M-CMU coming in as a very close second (0.7731). Unlike the other DNN models, M-LSTM.MI is trained with a cost-sensitive learning algorithm that takes class imbalances into account, resulting in the highest weighted macro-average F1 (0.9701) among all models.

As the confusion matrix in Figure 2 reveals, the M-LSTM.MI model performs very poorly for the malware family *banjori*. Indeed, only 5% of *banjori* domain names in the data are recognized as such by the M-LSTM.MI classifier, while 64% manage to evade the DGA classifier and are labeled as benign. The remaining *banjori* domain names are either mistaken for *qakbot* (21%) or *ramnit* (10%). Interestingly, the reverse does not hold: there is not a single *qakbot* or *ramnit* domain name in the test data that gets mistaken by the classifier for a *banjori* domain name. The poor performance of the M-LSTM.MI model for domain names generated by *banjori*, which is a time-invariant DGA, sheds some insight on the cause of the lower TPR for TI-DGAs reported in Table V. Note that, as becomes clear from Figure 2, the other culprit is the TI-DGA *simda*; 30% of the domain names generated by this family evade the classifier and are labeled as benign. Recall that in our experimental setup, TI-DGA domain names are split between train and test datasets based on their seeds. It is likely that domains generated based on different seeds exhibit different character distributions and hence vary in terms of linguistic features, making them more difficult to detect after a seed change.

VII. REAL TRAFFIC ANALYSIS

We applied the best performing DNN classifier (B-LSTM.MI) and the B-RF classifier from Section VI to a day of resolved domains, collected for 2018-08-27. We restricted our analysis to resolved domains only, since these indicate potential active C&C centers. In the preprocessing step, we removed all `xn--` domains from the rest of the traffic in order to reduce the false positive rate for our classifiers.[7] The resulting data contains 57,667,269 unique resolved domains consisting of a SLD and a TLD without a 3LD (third-level domain). 7,416 of these domains occur in Bambenek and/or DGArchive during the period 2018-08-25 through 2018-08-29, indicating that they were successfully registered domains, used for C&C communication. Note that 7,416 is low compared to the number of overlapping DGAs between real traffic and Bambenek in Fig. 1, because only a small fraction of DGA domains are actually registered. 3,049 of the 7,416 resolved known DGA domain names from Aug 27 belong to the 15 families used to train our classifiers from Section VI, while the majority belongs to other families.

The columns with header "Original" in Table VIII contain results for the models which were exactly as described and trained in Section V and VI, when applied to all resolved domains from Aug 27. The B-LSTM.MI classifier detects an impressively high number of the resolved known DGA domains (6,068 out of 7,416), while the B-RF classifier detects roughly half. It is interesting to notice that the B-LSTM.MI classifier manages to detect many domains from families that it did not see during training.

As can be seen in Table VIII, about 1.45% of the domains observed in real traffic on Aug 27 were flagged by the B-LSTM.MI classifier as malicious (out of which 44% were flagged with probability 1.0). We were able to locate 0.7% of those flagged domains in DGA Archive and/or Bambenek. 99% of the flagged domains were not present in either of these blacklists. As the numbers in Table VIII indicate, while flagging substantially fewer domain names as malicious, the original B-RF classifier likely has a much lower TPR (recall) than the B-LSTM.MI classifier, which is in line with our previous findings from Table V.

[7]https://umbrella.cisco.com/blog/2014/10/16/detecting-pinyin-domains/

TABLE VI
MULTICLASS CLASSIFICATION RESULTS - PART I

	Family	M-LSTM.MI			M-RF			OVA-RF		
		Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
TD	Cryptolocker-Flashback	0.3442	0.5707	0.4294	0.2693	0.2169	0.2403	0.2101	0.4132	0.2786
TD	dyre	0.9991	1.0000	0.9995	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
TD	locky	0.6765	0.5892	0.6299	0.6155	0.4466	0.5176	0.5287	0.5367	0.5327
TD	murofet	0.8885	0.9355	0.9114	0.8345	0.8702	0.8519	0.8647	0.8445	0.8545
TD	necurs	0.9567	0.8412	0.8952	0.9295	0.7999	0.8598	0.9769	0.7761	0.8650
TD	nymaim	0.6702	0.8879	0.7638	0.3294	0.2379	0.2763	0.2814	0.3530	0.3131
TD	Post Tovar GOZ	0.9999	0.9999	0.9999	1.0000	0.9997	0.9998	1.0000	0.9997	0.9998
TD	pykspa	0.8572	0.8715	0.8643	0.6845	0.6657	0.6750	0.6941	0.5969	0.6419
TD	qakbot	0.7781	0.6546	0.7110	0.6098	0.5786	0.5938	0.6847	0.5176	0.5895
TD	ranbyus	0.8759	0.9320	0.9031	0.8870	0.8900	0.8885	0.8626	0.8771	0.8698
TI	banjori	0.9887	0.2507	0.4000	0.1015	0.0384	0.0557	0.1203	0.0434	0.0638
TI	tinba	0.9161	0.9892	0.9513	0.7909	0.8924	0.8386	0.7970	0.8776	0.8353
TI	ramnit	0.2868	0.6582	0.3995	0.2530	0.2755	0.2637	0.2203	0.3441	0.2686
TI	simda	0.9569	0.4980	0.6551	0.7350	0.4930	0.5902	0.6470	0.4920	0.5590
TI	shiotob/urlzone/bebloh	0.9794	0.9163	0.9468	0.9740	0.8698	0.9190	0.9724	0.8621	0.9139
	Alexa	0.9936	0.9972	0.9954	0.9751	0.9888	0.9819	0.9751	0.9888	0.9819
	Macro-average	0.8229	0.7870	0.7784	0.6868	0.6414	0.6595	0.6772	0.6576	0.6604
	Weighted macro-average	0.9743	0.9707	0.9701	0.9402	0.9462	0.9427	0.9427	0.9442	0.9424

TABLE VII
MULTICLASS CLASSIFICATION RESULTS - PART II

	Family	M-Endgame			M-CMU			M-MIT		
		Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
TD	Cryptolocker - Flashback DGA	0.4069	0.3335	0.3665	0.3458	0.4534	0.3923	0.3481	0.4266	0.3834
TD	dyre	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
TD	locky	0.8947	0.4288	0.5797	0.6950	0.5547	0.6170	0.8121	0.4597	0.5871
TD	murofet	0.8824	0.9410	0.9107	0.8716	0.9520	0.9100	0.8947	0.9213	0.9078
TD	necurs	0.9344	0.8689	0.9004	0.9499	0.8507	0.8976	0.9520	0.8587	0.9030
TD	nymaim	0.6876	0.8433	0.7575	0.6604	0.8420	0.7403	0.6979	0.8133	0.7512
TD	Post Tovar GOZ DGA	1.0000	0.9999	1.0000	0.9999	0.9998	0.9999	1.0000	0.9998	0.9999
TD	pykspa	0.8520	0.8789	0.8652	0.8630	0.8729	0.8679	0.8655	0.8500	0.8577
TD	qakbot	0.8015	0.6700	0.7298	0.8224	0.6557	0.7296	0.7270	0.7097	0.7182
TD	ranbyus	0.8783	0.9256	0.9013	0.8647	0.9500	0.9053	0.8668	0.9321	0.8983
TI	banjori	0.9835	0.1786	0.3023	0.9712	0.1782	0.3012	0.5775	0.0059	0.0116
TI	tinba	0.9148	0.9913	0.9515	0.9116	0.9927	0.9504	0.9218	0.9908	0.9550
TI	ramnit	0.3806	0.5828	0.4605	0.4083	0.5528	0.4697	0.3177	0.6332	0.4231
TI	simda	0.9552	0.4885	0.6465	0.9076	0.4995	0.6443	0.9174	0.4975	0.6451
TI	shiotob/urlzone/bebloh	0.9973	0.9100	0.9517	0.9902	0.9130	0.9500	0.9932	0.9180	0.9542
	Alexa	0.9886	0.9986	0.9936	0.9893	0.9971	0.9932	0.9899	0.9979	0.9939
	Macro-average	0.8473	0.7524	0.7698	0.8282	0.7665	0.7731	0.8051	0.7509	0.7493
	Weighted macro-average	0.9712	0.9707	0.9681	0.9711	0.9701	0.9680	0.9682	0.9690	0.9659

TABLE VIII
ANALYSIS ON ONE DAY OF REAL TRAFFIC DATA (27 AUG 2018)

Observation	B-LSTM.MI		B-RF	
	Original	Adapted	Original	Adapted
Total number of resolved domain names	57,667,269	57,667,269	57,667,269	57,667,269
Total number of resolved domain names flagged as malicious by the classifier	839,212	146,067	207,661	108,599
Out of the flagged malicious resolved domains, total number of domain names found in the blacklists Bambenek or DGArchive	6,068	5,623	3,645	2,873
Out of the flagged malicious resolved domains, total number of domain names found in the whitelist Alexa	774	674	14	886
Out of the flagged malicious resolved domains, total number of domain names that are unaccounted	832,370	139,770	204,002	104,840

We manually inspected the domains that were flagged as DGA but were not present in our blacklist and noticed a substantial number (up to a few hundred thousand domains) of legitimate domains, including content distribution networks domains, well-known legitimate hostnames such as *blogspot*, and dynamic DNS associated domains.

A likely cause for the high number of flagged domain names that are unaccounted for, is that the whitelist Alexa might not be sufficiently representative of the set of non-malicious domains that occur in real traffic. To investigate this further, we retrained the original B-RF and B-LSTM.MI classifiers on a dataset which is created by replacing the Alexa domain names from the training data in Table III with a random sample of 1 million resolved domains that occurred in real traffic on Aug 10, 2018. Labeling resolved domains as benign for training purposes in this manner does not result in a cleanly labeled ground truth dataset. Instead, it generates a weakly labeled training dataset that very likely contains some erroneous labels. However, since resolved DGAs are very rare when compared to the total number of resolved domains in a network, the noise in the labels in limited, and the training dataset is still usable in practice. The corresponding results, which are reported in the "Adapted" columns in Table VIII, show that training the models on noisy benign training examples obtained from real traffic data instead of on a clean whitelist, helps to substantially reduce the number of flagged domain names, making the B-LSTM.MI classifier more suitable to deploy in practice. These results suggest that using a heuristically labeled dataset such as in [14] is better than just using Alexa.

VIII. Conclusion

In this paper we compared the performance of state-of-the-art classifiers for DGA domain name detection when trained and evaluated on test data split according to time and seed. In line with previous work, we observed that, on ground truth labeled data, deep learning based models outperformed random forest models both for the task of binary classification (i.e. identifying whether the domain name is malicious or not) and multiclass classification (i.e. assigning a malicious domain name to the generating malware family). We found all classifiers are more robust against changes in the seed of time-dependent DGA families compared to time-invariant DGA families. The latter might be caused by the presence of some TI-DGAs in our data which are particularly hard to detect, such as *banjori*. Finally, when applying the best performing classifier to large batches of real traffic data, we observed that a high number of domain names are unjustifiably classified as malicious with high confidence by the classifier. This is likely due to the fact that Alexa is not sufficiently representative of all non-malicious domain names in real traffic, thereby casting doubt on the practical usefulness of whitelist/blacklist trained DGA classifiers typically proposed in the literature.

Acknowledgement. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *USENIX Security Symposium*, 2016, pp. 263–278.

[2] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou II, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: Detecting the rise of DGA-based malware," in *USENIX Security Symposium*, vol. 12, 2012.

[3] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," *preprint arXiv:1611.00791*, 2016.

[4] M. Pereira, S. Coleman, B. Yu, M. De Cock, and A. Nascimento, "Dictionary extraction and detection of algorithmically generated domain names in passive DNS traffic," in *Proceedings of RAID 2018 (21st International Symposium on Research in Attacks, Intrusions and Defenses)*, ser. LNCS, vol. 11050, 2018, pp. 295–314.

[5] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "FANCI: Feature-based automated NXDomain classification and intelligence," in *USENIX Security Symposium*, 2018, pp. 1165–1181.

[6] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based botnet tracking and intelligence," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2014, pp. 192–211.

[7] P. Lison and V. Mavroeidis, "Automatic detection of malware-generated domains with recurrent neural models," *preprint arXiv:1709.07102*, 2017.

[8] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character level based detection of DGA domain names," in *Proc. of IJCNN at WCCI2018 (2018 IEEE World Congress on Computational Intelligence)*, 2018, pp. 4168–4175.

[9] C. Choudhary, R. Sivaguru, M. Pereira, B. Yu, A. Nascimento, and M. De Cock, "Algorithmically generated domain detection and malware family classification," in *Proceedings of the Sixth International Symposium on Security in Computing and Communications (SSCC'18)*, ser. Communications in Computer and Information Science Series (CCIS). Springer, 2018.

[10] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, "A LSTM based framework for handling multiclass imbalance in DGA botnet detection," *Neurocomputing*, vol. 275, pp. 2401–2413, 2018.

[11] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with DNS traffic analysis," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1663–1677, 2012.

[12] J. Kwon, J. Lee, H. Lee, and A. Perrig, "PsyBoG: a scalable botnet detection method for large-scale DNS traffic," *Computer Networks*, vol. 97, pp. 48–73, 2016.

[13] P. Lison and V. Mavroeidis, "Neural reputation models learned from passive DNS data," in *IEEE International Conference on Big Data*, 2017, pp. 3662–3671.

[14] B. Yu, D. Gray, J. Pan, M. De Cock, and A. Nascimento, "Inline DGA detection with deep networks," in *IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 683–692.

[15] H. S. Anderson, J. Woodbridge, and B. Filar, "Deepdga: Adversarially-tuned domain generation and detection," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, 2016, pp. 13–21.

[16] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive DNS analysis." in *NDSS Symposium*, 2011.

[17] B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. Cohen, "Tweet2vec: Character-based distributed representations for social media," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, vol. 2, 2016, pp. 269–274.

[18] S. Vosoughi, P. Vijayaraghavan, and D. Roy, "Tweet2vec: Learning tweet embeddings using character-level CNN-LSTM encoder-decoder," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 1041–1044.