Hardening Inline DGA Classifiers Against Adversarial Attacks

Raaghavi Sivaguru

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Computer Science and Systems

University of Washington

2019

Reading Committee:

Dr. Martine De Cock, Chair

Dr. Anderson Nascimento

Dr. Femi Olumofin

Jonathan Peck

Program Authorized to Offer Degree:
Computer Science and Systems

University of Washington

Abstract

Hardening Inline DGA Classifiers Against Adversarial Attacks

Raaghavi Sivaguru

Chair of the Supervisory Committee:
Professor Dr. Martine De Cock
School of Engineering and Technology

Domain Generation Algorithms (DGAs) are widely used by cybercriminals to generate domain names on-the-go for C&C (command-and-control) purposes of establishing communication with the bots and instructing them to perform malicious activities. It is therefore important to detect domains generated by DGAs to block the communication between the bot and C&C. In recent years, Machine Learning based DGA detection systems are widely used to address this problem. However, it is found that classifiers that rely only on the domain name to detect DGAs are highly vulnerable to adversarial attacks. Adversarial attacks are intentionally devised by an attacker to fool a classifier and cause it to produce erroneous results. This is a serious concern as it degrades the performance of DGA detection classifiers.

In this thesis, we aim to defend DGA detection classifiers against adversarial attacks, without compromising the performance of existing state-of-the-art classifiers in the literature. One such technique is to use side information features obtained from the DNS query/response that cannot be easily manipulated by the adversary. Although there are past research works that use DNS features for a retrospective analysis of DNS traffic, to the best of our knowledge, there are no studies that leverage such data for *inline* detection of DGA domains. In our work, we train machine learning models based on tree ensembles and deep learning for DGA detection using side information (in addition to the domain name), which can be easily obtained in practice without relying on external data sources such as WHOIS. Besides, we

also disregard methods that analyze past DNS data to extract side information features, thereby resulting in a relatively lightweight computation for detecting DGA domains in real-time DNS applications. In the end, we also perform an empirical evaluation by applying the best performing classifiers trained using side information on one day of passive DNS traffic to compare its performance against well known state-of-the-art classifier that relies only on a domain name for DGA detection. Results show that classifiers trained using a combination of lexical and side information features, not only provide high performance but are also more robust to adversarial attacks than the classifiers that rely only on the domain name for *inline* DGA detection.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I express sincere gratitude to Dr. Martine, Dr. Anderson, Dr. Femi and Jonathan for their incredible insights and guidance. Special thanks to Infoblox Inc. for providing great research opportunities to explore real-time applications of Machine Learning techniques in the context of DGA detection. It has been my honor to work with inspiring people at the University of Washington Tacoma and Infoblox during the course of my thesis. I would also like to thank all my professors, friends and family who have helped me succeed through this amazing journey.

DEDICATION

Dedicated to my dear husband, Pratap Karthick Udayakumar

Chapter 1

INTRODUCTION

1.1 Background

Domain Generation Algorithms (DGAs) refer to malware code that takes a seed input to generate a pseudo-random sequence of characters, and concatenates it with a public suffix to form a domain name string. In recent years cybercriminals widely use DGAs, registering automatically generated domain names on the go to establish communication between the command-and-control (C&C) server and infected machines. Once the communication channel is established, the C&C center instructs the bots to perform malicious activities.

It is therefore important to identify domain names generated by DGAs in order to block the communication with the C&C center and blacklist them. Machine learning (ML) based DGA detection methods have become popular in recent years, and there are well known state-of-the-art DGA classifiers in the literature to address this issue. Some of the techniques include a "featureful" approach where the ML model is trained using human-engineered features extracted from the domain name string [26, 25, 29, 13, 34] or side information [16] or combination of both[1, 6, 5, 10, 7, 17]; whereas others follow a "featureless" approach where the ML algorithm automatically learns the features from the domain name [36, 18, 41].

Although there are high performing ML-based systems for DGA detection, they are highly vulnerable to new DGAs and adversarial attacks [21, 30, 2, 12, 27, 32]. In this thesis, we focus on a specific kind of adversarial attacks in which an adversary attempts to fool a classifier through artificially generated inputs, also known as "adversarial instances", that are intentionally crafted to cause the ML model to misclassify it. An attacker devises such adversarial attacks against ML classifiers for various purposes, including performance degradation, evasion from being detected etc. Hence it is necessary to build robust models

that can defend against adversaries.

1.2 Contributions

In Chapter 2, we trained classifiers for two tasks, namely DGA detection (binary classification) and DGA malware family classification (multiclass classification). We used both featureful and featureless approaches for training the classifiers using Random Forest and deep learning architectures respectively. This work was primarily done as part of the DMD2018 [1] competition. Our models ranked *first* in two out of the four challenges posted by the organizers. The best performing model that won the DGA detection task is an ensemble that consisted of 3 deep learning classifiers namely Invincea[24], Endgame[36] & NYU[42]. The model that outperformed other deep learning architectures in DGA malware classification task is the one-vs-rest Random Forest classifier proposed by us.

In Chapter 3, we train and evaluate DGA classifiers using state-of-the-art neural network architectures [31, 36, 11, 33] and random forest algorithms [8] for DGA detection and malware family classification. The main contribution & novelty in this work are as listed below:

- We assessed the vulnerability of the DGA classifiers to changes in time and seed of the DGA generator. To do this, we created two sets of training & testing data, which were split based on the seed of the DGA generator and based on the time when the DGAs were queried in real passive DNS traffic. We found that the classifiers were robust to change in seed of the time-dependent malware families (TD-DGAs) as compared to the time-invariant DGA families (TI-DGAs).

- Leveraging on publicly available data sources, such as Alexa[2] top 1 million domains, for obtaining the ground truth for the benign domains may not result in a good performance when deployed in real-time DNS applications. This is mainly due to the

[1] https://nlp.amrita.edu/DMD2018/

[2] https://aws.amazon.com/alexa-top-sites/

fact that the domains ranked by Alexa are very different from the actual benign domains that occur in real-time DNS systems. Hence DGA classifiers that were trained by obtaining ground truth labels from Alexa resulted in higher false positives, which is strictly undesirable in cybersecurity applications as we do not want to block the legitimate traffic.

In Chapter 4, we propose to devise a simple adversarial attack algorithm called "CharBot" [21] that can evade state-of-the-art DGA classifiers trained in Chapter 3 and measure the vulnerability of classifiers that rely only on the domain name to detect DGAs. To this end, we trained and measured the robustness of the classifiers against 3 different adversarial attacks in literature such as DeepDGA[2] & DeceptionDGA[30] and our own attack algorithm "CharBot". The main contribution and findings in this work are as follows:

- A simple, yet powerful, *"black-box targeted evasion attack"* algorithm that generates samples to evade from being detected as malicious by any DGA classifiers with high probability.

- Simple adversarial training by appending CharBot domains to the training data set may not be a strong defense technique. State-of-the-art DGA classifiers that use just the domain name are highly vulnerable to adversarial attacks like CharBot. Hence, there is a need for more robust DGA classifiers that uses additional information apart from the domain name to detect DGAs.

In Chapter 5, we propose to build inline DGA classifiers that use side information in addition to the domain name, as a defense strategy against adversarial attacks. The side information features are carefully chosen to perform *inline* detection of DGA domains, where the prediction is done on a per-domain basis and does not require analysis of past DNS traffic or any additional resources, such as WHOIS, to extract values for the feature. To the best of our knowledge, our work is the first to use side information features, in addition to the

features extracted from the domain name, to perform *inline* DGA detection. This technique not only provides higher predictive performance but also makes the classifiers robust against adversarial attacks when compared to classifiers that rely only on the domain name.

1.3 Publications

The work in this thesis has led to several publications corresponding to chapters, namely:

- **Chapter 2:** Choudhary, C., **Sivaguru, R.**, Pereira, M., Yu, B., Nascimento, A.C. and De Cock, M., 2018, September. "Algorithmically generated domain detection and malware family classification". In International Symposium on Security in Computing and Communication (pp. 640-655). Springer, Singapore.

- **Chapter 3: Sivaguru, R.**, Choudhary, C., Yu, B., Tymchenko, V., Nascimento, A. and De Cock, M., "An evaluation of DGA classifiers", In 2018 IEEE International Conference on Big Data (Big Data) (pp. 5058-5067). IEEE.

- **Chapter 4:** Peck, J., Nie, C., **Sivaguru, R.**, Grumer, C., Olumofin, F., Yu, B., Nascimento, A. and De Cock, M. "CharBot: A Simple and Effective Method for Evading DGA Classifiers", IEEE Access, vol. 7, pp 91759 - 91771, 2019.

Chapter 2

ALGORITHMICALLY GENERATED DOMAIN DETECTION AND MALWARE FAMILY CLASSIFICATION

This chapter is focused on training ML classifiers for DGA detection (binary classification) and identification of examined both featureless and featureful methods for the above tasks and investigated on choosing the right model parameters for building the classifier. The main contributions in this chapter include:

- Developing code for feature extraction from the domain names, which are used to train the Random Forest (RF) classifier.
- Trained RF classifier for binary and multiclass classification tasks
- Submitted the above models to DMD2018[1] competition and ranked first on one test set 1 and third on a test set 2 provided by the DMD organizers.

Bibliographical details:

*Choudhary, C., **Sivaguru, R.**, Pereira, M., Yu, B., Nascimento, A.C. and De Cock, M., 2018, September. "Algorithmically generated domain detection and malware family classification". In International Symposium on Security in Computing and Communication (pp. 640-655). Springer, Singapore.*

[1] https://nlp.amrita.edu/DMD2018/

Algorithmically Generated Domain Detection and Malware Family Classification

Chhaya Choudhary[1], Raaghavi Sivaguru[1], Mayana Pereira[2], Bin Yu[2], Anderson C. Nascimento[1], and Martine De Cock[1,3]

[1] University of Washington, Tacoma
{chhayc,raaghavi,andclay,mdecock}@uw.edu
[2] Infoblox Inc.
{mpereira,biny}@infoblox.com
[3] Ghent University
martine.decock@ugent.be

Abstract. In this paper, we compare the performance of several machine learning based approaches for the tasks of detecting algorithmically generated malicious domains and the categorization of domains according to their malware family. The datasets used for model comparison were provided by the shared task on Detecting Malicious Domain names (DMD 2018). Our models ranked first for two out of the four test datasets provided in the competition.

Keywords: domain generation algorithms, malware, supervised learning, deep learning, random forest

1 Introduction

Domain Generation Algorithms (DGAs) are widely used by malware as a way to create a communication channel between infected machines and *command-and-control* servers. The development of techniques for automatic detection of DGA domains has been extensively studied in the past few years, leading, among other things, to machine learning models that are effective at detecting such domains in traffic.

There are two main machine learning approaches for automatic detection of malicious domains: 1) Combining feature engineering of network and lexical/linguistic characteristics of known DGA domains and benign domains with supervised machine learning techniques [3, 9, 16]; 2) Leveraging modern feature-less deep learning techniques for text classification [6, 15, 17].

In this paper, we apply and compare both approaches to solve two distinct tasks. The first task is regarding binary domain classification, i.e. classify domains as either DGA generated or legitimate domains. The second task is a multiclass classification problem of detecting and categorizing the DGA generated domains according to their malware family.

Our trained classifiers outperformed the ones proposed by other teams in the DMD2018 challenge for two of the four competition scenarios, based on several metrics, including accuracy, F1-score, recall and precision. Rankings of the models are presented in Section 5. In particular, we obtained first place (1) for one of the binary classification tasks with a deep neural network that was trained to discover important features automatically, and classify domain names as benign or malicious accordingly, and (2) for one of the multiclass classification tasks with a Random Forest that was trained on human defined features extracted from the domain name strings. Specific details about the shared tasks can be found on the DMD2018 website[4].

Table 1: Overview of recent deep learning model architectures for character based text classification [18]

Model Name	**Architecture**	**Reference**
Endgame	single LSTM layer	[15]
Invincea	parallel CNN layers	[8]
CMU	forward LSTM layer + backward LSTM layer	[5]
MIT	stacked CNN layers + single LSTM layer	[14]
NYU	stacked CNN layers	[19]

Background and related work Many DGA algorithms start from random seeds, producing domains that are distinctly different from usual benign domains [7]. They appear more "random looking", such as, for example, the domain sgxyfixkhuark.co.uk generated by the malware *Cryptolocker*. DGA domains are typically detected by techniques that leverage the distribution of characters in the domain, either through human engineered lexical features [3, 9] or through training deep neural networks [6, 8, 10, 15, 17, 18]. In deep learning, useful features are discovered automatically, thereby offering the potential to bypass the human effort of feature engineering and allowing easier adaptation of the models to new and emerging malware families.

A variety of deep neural network architectures were proposed recently for tasks related to text classification. They are relevant for DGA domain name detection, which can be thought of as a short text classification task. In [18], five state-of-the-art architectures as presented in Table 1 are applied to the binary task of detecting whether a domain name is benign or malicious. Out of these five deep neural network architectures for character based text classification, the first two were originally proposed for the detection of malicious domain names [15] and URLs [8], while the remaining ones [5, 14, 19] were proposed for text classification in general, and adapted in [18] for the specific task of DGA detection. The studied neural networks contain Long Short Term Memory (LSTM) layers [15], bidirectional LSTM layers which process the input string in a forward and a backward layer and then combine the output from these layers to pass on to further layers [5], Convolutional Neural Network (CNN) layers, either stacked

[4] `http://nlp.amrita.edu/DMD2018/`, Accessed: 2018-07-18

[19] or in parallel [8], or a combination of both LSTM and CNN layers [14]. For a comprehensive overview of all architecture details, we refer to Yu et al. [18].

In our experiments, we reproduce previously proposed methodologies for DGA domain detection and compare all methodologies by testing them on the same benchmark datasets, for DGA detection (binary classification) as well as for malware family detection (multiclass classification).

2 Datasets

We received two training datasets and four testing datasets from DMD2018, namely one training dataset and two testing datasets for each of the binary and multiclass classification tasks. All the datasets are highly unbalanced, meaning the number of samples in our class of interest (malicious class) is much smaller or rarer than the other (benign class), or vice versa. All the datasets contain domain name strings that consist of at least a second-level-domain (SLD) followed by a top-level-domain (TLD), separated by a dot, as in e.g. google.com. Many domains have a third-level-domain (3LD) as well, as in e.g. ns-738.awsdns-28.net where ns-738 is the 3LD.

To compose the datasets, malicious domain names were collected by the DMD2018 organizers using publicly available DGA algorithms[5], the OSINT feeds from Bambenek Consulting [2], and netlab-360[6], while benign domain names were collected from Alexa [1] and openDNS[7]. Additional data was collected privately by the DMD organizers within a lab using a port mirroring approach. Passive sensors were deployed in an internal network to collect the Domain Name System (DNS) traffic from different DNS servers. The experimental set-up and data collection process is reported in detail in [11, 12, 13].

2.1 DMD Shared Task Datasets

The description of the shared task datasets received from DMD is as follows:

Subtask 1 - Binary Classification. Subtask 1 has two classes namely benign and DGA (malicious). The original subtask 1 training dataset contains 790,739 domains out of which 655,683 domains are benign and 135,056 domains are DGA. All the benign domains are labeled as 0 and all the DGAs are labeled as 1. There are two testing datasets Test 1 and Test 2, the distributions of which are shown together with that of the training data in Table 2. The correct labels of the domain names in the test sets are not given, i.e. it is not known in advance to DMD competition participants which of the domains in Test 1 and Test 2 are benign and which ones are malicious.

[5] `https://github.com/baderj/domain_generation_algorithms`, Accessed: 2018-07-24

[6] `https://data.netlab.360.com/dga/`, Accessed: 2018-07-24

[7] `https://umbrella.cisco.com/blog/2016/12/14/cisco-umbrella-1-million/`, Accessed: 2018-07-24

Subtask 2 - Multiclass Classification. The dataset used for multiclass classification has a collection of domains belonging to the "benign family" and 20 distinct DGA families, thereby summing to a total of 21 families. The original subtask 2 training dataset contains 397,777 domains out of which 100,000 domains are benign and 297,777 domains are DGAs. In this task too, there are two testing datasets, namely Test 1 and Test 2, each having a varied proportion of samples belonging to the 21 classes (see Table 3).

Table 2: Data Statistics, Subtask 1 - Binary Classification

Type	Benign	DGA	Total
Training	655,693	135,056	790,739
Test 1	2,349,331	108,076	2,457,407
Test 2	182	2,740	2,922

Table 3: Dataset Description, Subtask 2 - Multiclass Classification

Family	Label	Train	Test 1	Test 2	Family	Label	Train	Test 1	Test 2
benign	0	100k	120k	40k	pyskpa	11	15k	25k	2k
banjori	1	15k	25k	10k	qadars	12	15k	25k	2,300
corebot	2	15k	25k	10k	qakbot	13	15k	25k	1k
dircrypt	3	15k	25k	300	ramdo	14	15k	25k	800
dnschanger	4	15k	25k	10k	ranbyus	15	15k	25k	500
fobber	5	15k	25k	800	simda	16	15k	25k	3k
murofet	6	15k	16,667	5k	suppobox	17	15k	20k	1k
necurs	7	12,777	20,445	6.2k	symmi	18	15k	25k	500
newgoz	8	15k	20k	3k	tempedreve	19	15k	25k	100
padcrypt	9	15k	20k	3k	tinba	20	15k	25k	700
proslikefan	10	15k	20k	3k	**Total**	21	397,777	587,112	103,200

2.2 Data Cleaning of DMD Training Datasets

We performed exploratory data analysis for both the DMD training datasets, and removed duplicates as well as domains not having a valid SLD or TLD. For example, we found 20,434 domains which occurred more than once in the subtask 1 training dataset and 28,740 domains in which either the SLD or TLD was missing.

2.3 Additional Datasets

In addition to the datasets provided by DMD, we used the following datasets to train our classifiers:

– **Alexa-Bambenek.** The AlexaBambenek dataset consists of the top 1M domains from Alexa [1] (considered benign) and 1M DGA domains from the OSINT feeds [2]. For more details about this dataset, we refer to [18].
– **DGArchive.** The DGArchive dataset[8] is a repository of known DGA domains.
– **Real-Traffic.** The real-traffic dataset originates from a real-time stream of passive DNS data obtained from Farsight Security[9], weakly labeled using heuristic rules as described in [17].

Table 4 contains statistics for the additional datasets used to train various models as explained in Section 4. All the domains in the datasets listed above have at least an SLD and a TLD, and some of the domains have a 3LD too.

Table 4: Data Statistics, Additional Datasets

Dataset	Benign	DGA	Total
Alexa-Bambenek	1M	1M	2M
DGArchive	NA	15,772,535	15,772,535
Real-Traffic	15,534,803	18,247,899	33,782,702

3 Method

In our experiments, we evaluated two main approaches of machine learning based automatic detection of malicious domains: the first approach is based on human-engineered features combined with supervised machine learning algorithms, such as Random Forests; and an alternative approach based on featureless Deep Neural Network (DNN) architectures for text classification and categorization. We give a detailed description of the features extracted from domain names, as well as all the machine learning techniques used in our experiments.

3.1 Featureful Approach

In the featureful approach we convert the benign and malicious domains into feature vectors, using 28 lexical/linguistic features, many of which are well known in the literature on DGA detection. These 28-dimensional feature vectors consist of the features mentioned in [17], as well as the following features:

– **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[10]: “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.

[8] `https://dgarchive.caad.fkie.fraunhofer.de/site/`, Accessed: 2018-07-24
[9] `https://www.farsightsecurity.com/`, Accessed: 2018-07-24
[10] `https://www.spamhaus.org/statistics/tlds/`, Accessed: 2018-07-18

- **Number of Tokens in SLD (tokens_sld)**: The number of tokens in the SLD. A token is a sequence of characters separated by "-".
- **Number of Tokens in 3LD (tokens_3ld)**: The number of tokens in the 3LD.
- **Length of SLD (sld_len)**: The length of the SLD, measured as the number of characters [3].
- **Length of 3LD (3ld_len)**: The length of the 3LD.
- **Length of TLD (tld_len)**: The length of the TLD.
- **Number of Unique Char (uni_domain)**: The number of unique characters in 3LD and SLD combined (excluding '.', '-').
- **Number of Unique Characters in SLD (uni_sld)**: The number of unique characters in SLD (excluding '.' and '-').
- **Number of Unique Characters in 3LD (uni_3ld)**: The number of unique characters in 3LD (excluding '.' and '-').
- **Longest Consonant Sequence in SLD (lng_con_seq)**: The length of the longest consonant sequence in the SLD, e.g. for the domain "google.com", "gl" is the longest consonant sequence and its value is 2 [4].
- **Consonant Ratio (con)**: The number of consonants in 3LD and SLD divided by their combined length. E.g. the domain "dfg.ca.gov" contains 4 consonants in the 3LD and SLD, namely 'd','f','g' and 'c', hence, the extracted feature value is 4/5.
- **Number Ratio (dig)**: The number of digits in 3LD and SLD divided by their combined length.
- **Number of Numerical Char in SLD (digits_sld)**: The number of numerical characters in SLD.
- **Number of Numerical Char in 3LD (digits_3ld)**: The number of numerical characters in 3LD.
- **Number of Dots (dots)**: The number of dots in the domain (not including the dot that separates the SLD from the TLD).
- **2-gram Circular Median (2gram_cmed)**: The domain string (excluding the TLD) 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 [17]) for the resulting string is computed.
- **3-gram Circular Median (3gram_cmed)**: The domain string (excluding the TLD) is duplicated and concatenated tail to head (e.g. "dfg.ca.gov" becomes "dfgcadfgca") and subsequently the 3-gram median (i.e. the nl3 feature mentioned in [17]) for the resulting string is computed.

Tree ensemble methods are among the most common algorithms of choice for supervised learning because of their general applicability and their state-of-the-art performance. A Random Forest (RF) is an ensemble of decision trees that are each separately trained on a different bootstrap sub-sample of the training data. During deployment, a majority vote among the prediction of all trees in the ensemble is taken to arrive at the target classification label for a new instance. This mechanism makes the ensemble less likely to overfit the training data. For both subtask 1 and subtask 2, we built RF classifiers using the 28 features

extracted from the domain names.

Random Forest Classifier for the Binary Classification Task. For subtask 1, we build a RF classifier for each of three different training datasets, leading to the first three models in Table 6. Each RF consists of 100 decision trees. Information gain is used as the selection criterion to select the best splitting attribute for each node in the trees, and all the features are considered during bootstrap sub-sampling to build the decision trees. A standard random seed is used for reproducibility of data and to compare the results. Each of these binary RF classifiers are trained to categorize the domains as *benign* or *malicious*.

In addition, for the fourth model in Table 6, we trained a RF classifier to categorize the domains as either *human readable* (HR, label 0) or *pseudo-random* (PR, label 1). Domain names in the PR category are immediately considered malicious, while domain names that are classified as HR are further passed to a separate binary RF classifier that is trained to distinguish between *benign* or *suppobox*. The latter is a DGA family containing human readable domains, hence domain names that are classified as *suppobox* are relabeled as 1 (*malicious*) and benign domains are labeled as 0.

Random Forest Classifier for the Multiclass Classification Task. For subtask 2, we build one binary RF classifier per DGA family. To do this, we begin by preparing the training dataset which is specific for each classifier. The training dataset is designed to be balanced with 50% domains that belong to the target family and 50% domains that belong to other families. For example, to build an RF classifier that classifies a domain as *banjori* (family label 1) or not, we create a training dataset that comprises of 50% domains belonging to family 1 (*banjori*) and 50% domains belonging to family 2 through 20, ensuring that the remaining 50% of non-target data has a stratified mix of the rest of the families. Once the individual training datasets are prepared, the corresponding binary RF classifiers are trained to identify if the domain belongs to the respective DGA family or not. We use two approaches to deploy these classifiers.

To deploy these one vs. rest RF classifiers, we directly pass the domains to each of the 20 DGA classifiers and compare the predicted probabilities. If the highest probability is greater than a threshold c, we simply assign the family label of the classifier that predicted it. However, if it is less than c, we make the final prediction as *benign*. The choice of the threshold c can be tuned (based on AUC score) to impact the predictions. In Table 7 and 10 we report results for $c = 0.5$ and $c = 0.9$.

In addition to the above, we also performed experiments with traditional multiclass RF classifiers trained on various datasets. The results of these experiments are consolidated in Table 7.

3.2 Featureless Approach

Deep learning techniques for detecting DGAs learn features automatically, thereby bypassing the human effort of feature engineering, and proved to be successful in

the task of DGA detection [6, 8, 10, 15, 18]. We trained a variety of deep neural networks that take as input the domain name string, which is *preprocessed* in the following way. Each domain name string is converted to lowercase and then represented as a sequence of ASCII values corresponding to its characters. We set the maximum length of a domain name as 75 characters [18]. If the original domain name is too short, we pad with zeroes on the left. If the original domain name is too long, then we truncate the domain name by removing characters from the right side of the SLD until the desired length is reached. All deep neural network architectures start with an embedding layer that learns to represent each character that can occur in a domain name by a 128-dimensional numerical vector, which is different from the original ASCII encoding. The embedding maps semantically similar characters to similar vectors, where the notion of similarity is automatically learned based on the classification task at hand.

Deep Learning for the Binary Classification Task. For this task, we trained five kinds of deep neural network models, referred to as Endgame (LSTM), Invincea (CNN), CMU (LSTM), MIT (CNN+LSTM), and NYU (CNN). These neural networks are based on previous work on the use of deep learning for character based text classification, as documented in Table 1. To optimize these neural networks for the task of classifying as a domain name as *benign* or *malicious*, we followed the same adaptations as in [18]. We refer to the latter for a detailed description of the architecture of all adapted models. When deploying these trained neural networks on a test dataset, we label a domain as *benign* if the probability is less than 0.5, and *malicious* if the probability is more than 0.5.

Deep Learning for the Multiclass Classification Task. In this task, we used a similar model architecture as used for the binary classification task. However, instead of two prediction classes, the models predict 1 out of 21 classes (one class corresponds to one family). Hence the output layer of the models from [18] is changed to use "softmax" as the activation function. This is to ensure that the output values are in the range of 0 and 1 and can be used as predicted probabilities. We performed one-hot encoding so that the output layer will create 21 output values, one for each class. The output value with the largest probability is taken as the final class predicted by the model.

4 Experimental Results

We performed various experiments using both featureful and featureless approaches. We set aside 10% from both cleaned training datasets provided by DMD to use as validation data. We refer to these test datasets as "DMD master test 1" and "DMD master test 2" (see Table 5). The remaining 90% of the DMD training datasets are referred to as "DMD master train 1" and "DMD master train 2". In addition, we use the datasets listed in Section 2.3 for training purposes as well, as indicated in Table 6 and 7.

Table 5: Data Statistics, DMD Master train and test datasets

Dataset	Benign	DGA	Total
DMD master train 1	546,211	121,440	667,651
DMD master test 1	60,691	13,493	74,184
DMD master train 2	89,039	267,727	356,766
DMD master test 2	9,893	29,748	39,641

4.1 Binary Classification

Table 6 contains the results of all models trained for the binary classification task of labeling domain names as *benign* or *malicious*, evaluated in terms of accuracy, F1-score, recall, and precision on the DMD master test 1 dataset. The models vary in terms of architecture (RF vs. DNN) as well as in terms of the data that was used for training.

Table 6: Experiments performed for binary classification (subtask 1). All models are evaluated on DMD master test 1.

Model name	Architecture	Train data	Accuracy	F1-score	Recall	Precision
RF_binary_1	RF	DMD master train 1	**96.98%**	**0.9155**	**0.9006**	**0.9308**
RF_binary_2	RF	Alexa-Bambenek	84.43%	0.6469	0.7847	0.5503
RF_binary_3	RF	DMD master train 1 + DMD master train 2	94.85%	0.8689	0.9384	0.8090
RF_binary_4	RF (HR vs PR)	DMD master train 1 + master train 2	95.11%	0.8707	0.9062	0.8379
Endgame_DMD	DNN	DMD master train 1	**98.65%**	**0.9632**	**0.9689**	**0.9577**
Invincea_1	DNN	Alexa-Bambenek	95.72%	0.8853	0.9083	0.8635
Endgame_1			96.05%	0.8904	0.8824	0.8986
NYU_1			93.97%	0.8425	0.8880	0.8015
CMU_1			95.77%	0.8837	0.8840	0.8835
MIT_1			94.08%	0.8468	0.8997	0.7998
Invincea_2	DNN	Pre-trained on Alexa-Bambenek and trained on DMD master train 1	**98.74%**	**0.9659**	**0.9828**	**0.9497**
Endgame_2			**98.70%**	**0.9650**	**0.9778**	**0.9525**
NYU_2			**98.70%**	**0.9647**	**0.9785**	**0.9512**
CMU_2			**98.67%**	**0.9637**	**0.9710**	**0.9564**
MIT_2			**98.70%**	**0.9649**	**0.9751**	**0.9548**
Endgame_Real	DNN	Real traffic data	81.92%	0.5994	0.7434	0.5021
Invincea_3	DNN	Pre-trained on Real traffic data and trained on Alexa-Bambenek	96.25%	0.8979	0.9073	0.8888
Endgame_3			96.47%	0.9017	0.8916	0.9120
NYU_3			95.32%	0.8732	0.8853	0.8615
CMU_3			97.17%	0.9217	0.9143	0.9292
MIT_3			96.55%	0.9049	0.9022	0.9076

The first four models correspond to *featureful RF classifiers*, trained using the features extracted from domains as mentioned in Section 3.1. Out of these, the highest accuracy and F1-score is obtained by an RF trained on DMD master train 1, i.e. the training data provided specifically for subtask 1. Augmenting the training data with DMD master train 2 (training data provided for subtask 2) or swapping it out for Alexa-Bambenek (an alternative ground truth dataset) did not improve the results.

The remaining models in Table 6 correspond to *featureless deep neural networks*, all trained on a workstation with an NVIDIA Titan Xp GPU and 12 GB RAM. The best results in terms of accuracy and F1-score are obtained through pre-training on Alexa-Bambenek data and post-training on DMD master train 1 data. This means that learning the weights of the neural network takes place in two stages: during the first stage, or pre-training, only examples of the Alexa-Bambenek training dataset are presented, while during the second stage, or post-training, only examples from the DMD master train 1 dataset are used. The results for the DNN classifiers confirm the observation already made for the RF classifiers that use of the DMD master train 1 dataset leads to the best results. This is not very surprising as the models in Table 6 are evaluated on DMD master test 1, which was drawn from the same distribution as DMD master train 1. Another interesting observation is that the five kinds of DNNs achieve a very similar best performance, despite of the vast differences in their architectures. These results are in line with what was reported in [18].

4.2 Multiclass Classification

Table 7 presents the results for the classifiers trained for malware family detection, evaluated on the DMD master test 2 dataset, using both featureful and featureless approaches. The reported F1-score, precision and recall are macro-averages, i.e. for each model, the F1-score, precision and recall are calculated for each of the 21 labels and an unweighted mean is taken (without considering label imbalance).

For testing the *featureful approach*, two types of RF models were built. One is the multilabel RF model where the classifier predicts the family (ranging between 0 and 20), given the features extracted from domain names. RF_multi_1 and RF_multi_2 from Table 7 are both such RF classifiers, different only in the data that was used for training. In the other technique, one binary "one vs. rest" RF classifier is developed to detect each family. Each classifier predicts the probability of the domain belonging to a particular family and the one with the highest likelihood is chosen as the final prediction, provided that this predicted probability reaches a predefined threshold c. Otherwise the domain is labeled as *benign*. The resulting model is called RF_multi_3 in Table 7, which we deployed with a threshold value $c = 0.9$.

As can be seen from Table 7, the best results in terms of accuracy and F1-score are achieved with a multilabel RF model trained on DMD master train 2 data. Table 8 shows a ranking of the importance of the features in the RF_multi_1 model, as compared to the RF_binary_1 model. An interesting observation from this table is that, while the relative ordering for RF_binary_1 and RF_multi_1

Table 7: Experiments performed for multiclass classification (subtask 2). All models are evaluated on DMD master test 2. F1-score, recall and precision are macro-averaged across all 21 labels.

Model name	**Architecture**	**Train data**	**Accuracy**	**F1-score**	**Recall**	**Precision**
RF_multi_1	RF	DMD master train 2	**89.04%**	**0.8676**	0.8707	0.8665
RF_multi_2	RF	DMD master train 2 + Alexa-Bambenek + DGArchive	73.45%	0.6568	0.6911	0.7875
RF_multi_3 $c = 0.9$	RF (one vs rest)	DMD master train 2 + Alexa-Bambenek + DGArchive	85.64%	0.8295	0.8361	0.8270
Endgame_multi	DNN	DMD master train 2 + Alexa-Bambenek + DGArchive	77.22%	0.6734	0.7192	0.7240
CMU_multi	DNN	DMD master train 2 + Alexa-Bambenek + DGArchive	78.05%	0.6922	0.7299	0.7286

is somewhat different, there is clear agreement among which features belong in the top half and which features belong in the bottom half. In particular, features extracted from the 3LD are considered less relevant for both binary classifcation and malware family detection.

Table 8: Ranking of features according to importance in the RF_binary_1 model from Table 6 and the RF_multi_1 model from Table 7.

Feature	RF_binary_1	RF_multi_1	Feature	RF_binary_1	RF_multi_1
sym	1	8	cer	15	13
lng_con_seq	2	5	uni_domain	16	14
tld_hash	3	2	digits_sld	17	18
sld_len	4	1	3ld_len	18	21
hex	5	7	flag_dig	19	17
domain_len	6	3	uni_3ld	20	22
uni_sld	7	15	2gram_med	21	23
tld_len	8	6	2gram_cmed	22	25
dig	9	4	digits_3ld	23	27
vow	10	9	3gram_cmed	24	24
con	11	10	3gram_med	25	26
ent	12	11	tokens_3ld	26	28
gni	13	12	dots	27	20
flag_dga	14	16	tokens_sld	28	19

For testing the *featureless approach*, we trained two deep neural network models, namely the Endgame (single LSTM layer) and the CMU (bidirectional LSTM layer) adapted with a softmax layer for multiclass classification. As is clear from Table 7, neither of these outperformed the RF approach.

5 Final Results

Based on the results from Section 4, we submitted a variety of trained classifiers to the DMD2018 competition. These models were evaluated by the DMD2018 organizers on the Test 1 and Test 2 datasets for both subtasks (see Section 2) in terms of accuracy, recall, precision, and F1-score. Table 9 and 10 show the results for all models and predictions that we submitted to DMD for the binary and multiclass classification tasks. The best results are highlighted in bold.

For the binary classification task, the results obtained with DNNs are better than those with RFs in Table 9, which is in line with our observation in Section 4.1. For Test 1, we obtain the best results with an ensemble (Invincea_2, Endgame_2, NYU_2) of deep neural network models, achieving an accuracy of 99% on Test 1. When deploying this ensemble, we use majority voting, i.e. we let each of the DNNs individually label the domain name, and subsequently select the most frequently predicted label as the final classification. Note in Table 9 that this ensemble also achieves a good result on Test 2, with an almost perfect recall of 0.999, meaning that it catches 99.9% of DGA domain names. It is still outperformed by the stand-alone Invincea_2 model, which achieves a higher precision for the same level of recall, leading to the best F1-score and accuracy of all our classifiers for Test 2.

Table 9: Final competition results for binary classification (subtask 1).

Model name	Architecture	Test data	Accuracy	F1-score	Recall	Precision
RF_binary_1	RF	Test 1	97.3%	0.708	0.683	0.736
		Test 2	59.4%	0.724	0.997	0.568
RF_binary_3	RF	Test 1	94.1%	0.584	0.424	0.941
		Test 2	65.8%	0.778	0.995	0.639
Endgame_DMD	DNN	Test 1	45.6%	0.081	0.044	0.548
		Test 2	63.9%	0.775	0.934	0.662
Invincea_2	DNN	Test 1	98.8%	0.876	0.808	0.956
		Test 2	**76.6%**	**0.858**	**0.999**	**0.751**
MIT_2	DNN	Test 1	98.9%	0.879	0.823	0.943
		Test 2	73.9%	0.838	0.999	0.722
Invincea_2	Ensemble	**Test 1**	**99.0%**	**0.892**	**0.828**	**0.966**
+ Endgame_2		Test 2	73.9%	0.839	0.999	0.723
+ NYU_2						

Regarding the malware family classification task, the results in Table 10 are in line with our observation from Section 4, in the sense that the DNNs that we trained for this task are outperformed by RFs. It is interesting to note that, while the best results for the multiclass classification task in Table 7 were achieved with the most straightforward multiclass random forest model (RF_multi_1), the best results in Table 10 stem from a one vs. rest RF model (RF_multi_3).

Table 10: Final competition results for multiclass classification (subtask 2).

Model name	Architecture	Test data	Accuracy	F1-score	Recall	Precision
RF_multi_1	RF	Test 1	63.1%	0.598	0.631	0.605
		Test 2	65.1%	0.616	0.651	0.652
RF_multi_2	RF	Test 1	57.5%	0.528	0.575	0.613
		Test 2	82.3%	0.827	0.823	0.885
RF_multi_3	RF	**Test 1**	**63.3%**	**0.602**	**0.633**	**0.618**
$c = 0.9$	(one vs rest)	**Test 2**	**88.7%**	**0.901**	**0.887**	**0.924**
RF_multi_3	RF	Test 1	61.9%	0.593	0.619	0.614
$c = 0.5$	(one vs rest)	Test 2	87.4%	0.890	0.874	0.919
Endgame_multi	DNN	Test 1	59.7%	0.559	0.597	0.654
		Test 2	80.2%	0.788	0.802	0.797
CMU_multi	DNN	Test 1	60.2%	0.566	0.602	0.696
		Test 2	79.7%	0.783	0.797	0.887

While both RF_multi_1 and RF_multi_3 have a comparable performance on Test 1, achieving an accuracy of 63%, it is especially on Test 2 that RF_multi_3 shines, with an accuracy of almost 89%. As indicated in Table 7, RF_multi_1 was trained using only training data provided explicitly for the competition, i.e. DMD master train 2, while for RF_multi_3 we used external training data. A plausible explanation for the good performance of RF_multi_3 on Test 2 is therefore that Test 2 contains domain names from a distribution/source that is quite different from the training data provided by DMD for subtask 2.

Table 11 shows the final ranking that we obtained in the competition for each of the four test datasets. We obtained first place for subtask 1 (binary classification), Test 1, with the ensemble model from Table 9, and first place for subtask 2 (multiclass classification), Test 2, with the RF_multi_3 model with deployment threshold $c = 0.9$ from Table 10.

Table 11: Final rankings for binary and multiclass classification tasks.

Task	Dataset	Accuracy	F1-score	Recall	Precision	Ranking
Binary Classification	Test 1	**99.0%**	0.892	0.966	0.828	**1**
	Test 2	**76.6%**	0.858	0.999	0.751	3
Multiclass Classification	Test 1	**63.3%**	0.602	0.633	0.618	5
	Test 2	**88.7%**	0.901	0.887	0.924	**1**

6 Conclusion

In this paper, we have investigated the performance of featureful (Random Forest) and featureless (Deep Neural Network) based classifiers for DGA detection, trained with various sources of publicly available and DMD provided data. For the binary classification task of determining whether a domain name is benign or malicious, we obtained the best results with a deep learning approach where the features are learned automatically from the data during the training process. For the multiclass classification task of determining which malware family a DGA domain name belongs to, we obtained the best results with a one vs. rest RF model trained on 28 features extracted from the domain names. The fact that the deep neural networks that we trained for malware family detection were outperformed by a RF is possibly due to the relatively small size of the dataset, with a limited number of training examples per malware family. An important take-away is thus that both featureful and featureless approaches have a valuable role to play in the defense against malware.

Acknowledgments

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

References

1. Does Alexa have a list of its top-ranked websites?, `https://support.alexa.com/hc/en-us/articles/200449834-Does-Alexa-have-a-list-of-its-top-ranked-websites-`, accessed: 2017-05-28
2. OSINT feeds from Bambenek Consulting, `http://osint.bambenekconsulting.com/feeds/`, accessed: 2017-05-28
3. Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., Dagon, D.: From throw-away traffic to bots: Detecting the rise of DGA-based malware. In: USENIX Security Symposium. vol. 12 (2012)
4. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: Finding malicious domains using passive DNS analysis. In: NDSS Symposium (2011)
5. Dhingra, B., Zhou, Z., Fitzpatrick, D., Muehl, M., Cohen, W.: Tweet2vec: Character-based distributed representations for social media. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. vol. 2, pp. 269–274 (2016)
6. Lison, P., Mavroeidis, V.: Automatic detection of malware-generated domains with recurrent neural models. preprint arXiv:1709.07102 (2017)
7. Plohmann, D., Yakdan, K., Klatt, M., Bader, J., Gerhards-Padilla, E.: A comprehensive measurement study of domain generating malware. In: USENIX Security Symposium. pp. 263–278 (2016)
8. Saxe, J., Berlin, K.: eXpose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys. preprint arXiv:1702.08568 (2017)

9. Schiavoni, S., Maggi, F., Cavallaro, L., Zanero, S.: Phoenix: DGA-based botnet tracking and intelligence. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 192–211. Springer (2014)
10. Tran, D., Mac, H., Tong, V., Tran, H.A., Nguyen, L.G.: A LSTM based framework for handling multiclass imbalance in DGA botnet detection. Neurocomputing 275, 2401–2413 (2018)
11. Vinayakumar, R., Poornachandran, P., Soman, K.: Scalable framework for cyber threat situational awareness based on domain name systems data analysis. In: Big Data in Engineering Applications, pp. 113–142. Springer (2018)
12. Vinayakumar, R., Soman, K., Poornachandran, P.: Detecting malicious domain names using deep learning approaches at scale. Journal of Intelligent & Fuzzy Systems 34(3), 1355–1367 (2018)
13. Vinayakumar, R., Soman, K., Poornachandran, P., Sachin Kumar, S.: Evaluating deep learning approaches to characterize and classify the DGAs at scale. Journal of Intelligent & Fuzzy Systems 34(3), 1265–1276 (2018)
14. Vosoughi, S., Vijayaraghavan, P., Roy, D.: 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. pp. 1041–1044 (2016)
15. Woodbridge, J., Anderson, H.S., Ahuja, A., Grant, D.: Predicting domain generation algorithms with long short-term memory networks. preprint arXiv:1611.00791 (2016)
16. Yadav, S., Reddy, A.K.K., Reddy, A.L.N., Ranjan, S.: Detecting algorithmically generated malicious domain names. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. pp. 48–61 (2010)
17. Yu, B., Gray, D., Pan, J., De Cock, M., Nascimento, A.: Inline DGA detection with deep networks. In: Data Mining for Cyber Security, Proceedings of International Conference on Data Mining (ICDM2017) Workshops. pp. 683–692 (2017)
18. Yu, B., Pan, J., Hu, J., Nascimento, A., De Cock, M.: Character level based detection of DGA domain names. In: Proc. of IJCNN at WCCI2018 (2018 IEEE World Congress on Computational Intelligence). pp. 4168–4175 (2018)
19. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in Neural Information Processing Systems. vol. 28, pp. 649–657 (2015)

Chapter 3

AN EVALUATION OF DGA CLASSIFIERS

This chapter focuses on evaluating the performance of state-of-the-art DGA detection classifiers in the literature based on two criteria. In the first experiment, we split the train and test data sets based on a) the seed used by the DGA generator b) the time when the DGA was generated. The purpose of this experiment is to compare the robustness of the classifier to changes in seed (time-invariant DGAs) and time (time-dependent DGAs) of the generator. In the second experiment we train the classifiers using publicly available data sets such as Alexa[1], Bambenek Consulting[2] and DGArchive[3] and real DNS traffic data collected from multiple ISPs, schools etc. The trained models are then exposed to one day of passive DNS traffic to measure if publicly available data sources are a good representation of domains observed in real DNS applications. My primary contributions in this chapter are:

- Lexical feature extraction from domain names.
- Trained B-RF, M-RF, OVA-RF [8], B-LSTM.MI and M-LSTM.MI [31] classifiers for DGA detection and family classification for experiments 1 and 2 mentioned above.
- Performed real traffic analysis to assess the quality of data sets used to train the classifiers.
- Submitted paper to CyberHunt IEEE BigData 2018[4] conference and presented the paper to peers.

---

[1] https://aws.amazon.com/alexa-top-sites/

[2] https://osint.bambenekconsulting.com/feeds/

[3] https://dgarchive.caad.fkie.fraunhofer.de/site/api.html

[4] https://securitylab.no/cyberhunt2018/

- Participated in Women in Cybersecurity (WiCyS) 2019[5] poster presentation competition.

Bibliographical details:

***Sivaguru, R.**, Choudhary, C., Yu, B., Tymchenko, V., Nascimento, A. and De Cock, M., 2018, December. "An evaluation of DGA classifiers". In 2018 IEEE International Conference on Big Data (Big Data) (pp. 5058-5067). IEEE.*

[5] https://www.wicys.org/conference-2019-home

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).

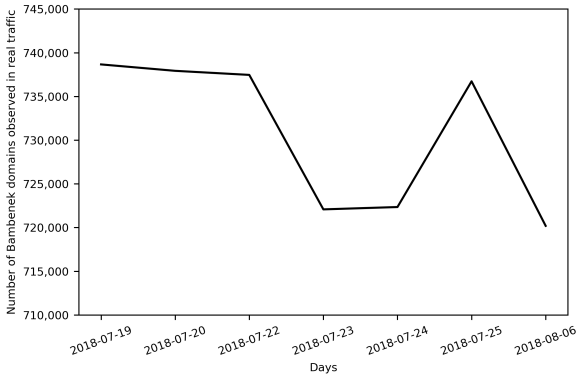


Fig. 1. Number of Bambenek domains observed in real traffic domains on the train and test days

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 ∩ Day7.

	Family	Day1-6	Day7	Day1-6 ∩ 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 ∩ 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
	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**	24,997	7,000	0
	tinba	42,212	12,882	2,093
	ramnit	9,907	2,639	0
	simda	4,006	2,008	22
	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
TI	**tinba**	112	28	140	42,212	12,882	55,094
TI	**ramnit**	50	13	63	9,907	2,639	12,546
TI	**simda**	12	3	15	4,006	2,008	6,014
TI	**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

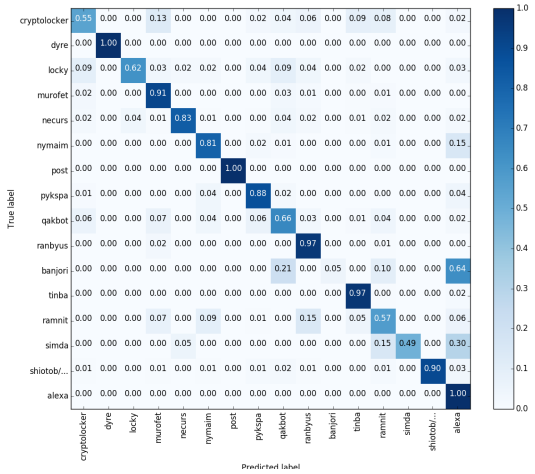


Fig. 2. Normalized confusion matrix for M-LSTM.MI

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.

Chapter 4

CHARBOT: A SIMPLE AND EFFECTIVE METHOD FOR EVADING DGA CLASSIFIERS

This chapter focuses on devising an adversarial attack algorithm to synthetically craft domain names that are intended to evade DGA classifiers. The primary goal in this work is to evaluate the robustness of state-of-the-art DGA classifiers to a simple adversarial attack such as CharBot. The CharBot algorithm takes in a list of benign second level domains (SLDs), a list of top level domains (TLDs), and a pseudo-random seed for reproducibility. It randomly chooses an SLD and replaces characters from two random indices with any valid DNS character. It then appends a randomly chosen TLD to form a domain name string. This domain name retains most of the linguistic characteristics of benign domains but can be used to perform malicious activities through command-and-control communication. These CharBot domains are then exposed to state-of-the-art DGA classifiers to assess its vulnerability against adversarial attacks. The contributions in this chapter include:

- Lexical feature extraction from domain names
- Trained B-RF [8, 29] and LSTM.MI [31] classifiers for DGA detection using real DNS traffic data
- Evaluate vulnerabilities of the above classifiers to adversarial attacks such as CharBot, DeepDGA [2] and DeceptionDGA [30]
- Performed simple adversarial training that includes re-training the model by appending evading instances to the original train data set.

Bibliographical details:

Peck, J., Nie, C., ***Sivaguru, R.****, Grumer, C., Olumofin, F., Yu, B., Nascimento, A. and De Cock, M. "CharBot: A Simple and Effective Method for Evading DGA Classifiers", IEEE Access, vol. 7, pp 91759 - 91771, 2019.*

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal
35

Received May 29, 2019, accepted June 26, 2019, date of publication July 5, 2019, date of current version July 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927075

CharBot: A Simple and Effective Method for Evading DGA Classifiers

JONATHAN PECK[1,2], CLAIRE NIE[3], RAAGHAVI SIVAGURU[3], CHARLES GRUMER[3], FEMI OLUMOFIN[4], BIN YU[4], ANDERSON NASCIMENTO[3], AND MARTINE DE COCK[1,3]

[1]Department of Applied Mathematics, Computer Science and Statistics, Ghent University, 9000 Ghent, Belgium
[2]Data Mining and Modeling for Biomedicine, VIB Inflammation Research Center, 9052 Ghent, Belgium
[3]School of Engineering and Technology, University of Washington at Tacoma, Tacoma, WA 98402, USA
[4]Infoblox, Santa Clara, CA 95054, USA

Corresponding author: Jonathan Peck (Jonathan.Peck@ugent.be)

The work of J. Peck was supported by the Research Foundation Flanders (FWO).

ABSTRACT Domain generation algorithms (DGAs) are commonly leveraged by malware to create lists of domain names, which can be used for command and control (C&C) purposes. Approaches based on machine learning have recently been developed to automatically detect generated domain names in real-time. In this paper, we present a novel DGA called CharBot, which is capable of producing large numbers of unregistered domain names that are not detected by state-of-the-art classifiers for real-time detection of the DGAs, including the recently published methods FANCI (a random forest based on human-engineered features) and LSTM.MI (a deep learning approach). The CharBot is very simple, effective, and requires no knowledge of the targeted DGA classifiers. We show that retraining the classifiers on CharBot samples is not a viable defense strategy. We believe these findings show that DGA classifiers are inherently vulnerable to adversarial attacks if they rely only on the domain name string to make a decision. Designing a robust DGA classifier may, therefore, necessitate the use of additional information besides the domain name alone. To the best of our knowledge, the CharBot is the simplest and most efficient black-box adversarial attack against DGA classifiers proposed to date.

INDEX TERMS Adversarial machine learning, domain generation algorithms, supervised learning.

I. INTRODUCTION

The purpose of distributing malware is often to extract sensitive information from victim machines or to use them for disseminating spam. To achieve this, botmasters need to be able to communicate with the infected machines, which is done via command-and-control (C&C) servers. The use of a fixed pool of C&C servers is not attractive, however, since these servers may be taken offline or blacklisted. Therefore, malware authors design *domain generation algorithms* or DGAs to automatically create many domain names that are likely to be unregistered and hence available for the malware to establish a communication channel [1]. A DGA makes use of a *seed*, i.e., some random number that is accessible to both the botmaster and the malware on the infected machines.

Possible seeds include the current date, trending topics on Twitter, weather forecasts, etc. Once this seed has been fixed, the botmaster, as well as all of the infected machines, can generate the same list of domains. The botmaster registers one of these domains and waits for the malware to successfully resolve a DNS query against it. From that point on, communication can take place. Should the C&C server ever be taken offline or have its domain blacklisted, this process can simply be restarted and a new C&C server can be established.

An extensive amount of research in the past decade has been devoted to the development of methods for detection of domains generated by DGAs [2]–[5]. These methods can be roughly divided into two classes: classifiers that detect DGAs based solely on the domain name itself; and classifiers that use some sort of context information, such as IP addresses of the source, traffic, and query patterns by the infected machines. Our focus in this paper is on the first kind of classifiers, i.e. techniques that can detect DGA domains in real-time based on the domain name string. These systems are particularly attractive since additional information beyond the domain name string can be expensive to acquire. It might

The associate editor coordinating the review of this manuscript and approving it for publication was Irene Amerini.

also simply not be available due to privacy concerns. Another significant advantage of systems that perform DGA detection based solely on the domain name is their potential use in real-time systems, blocking malicious domains before they are actually resolved. Accordingly, much research has been carried out to prevent this type of C&C communication using systems that can detect in real-time whether a domain name is likely generated by a DGA or not [2], [4]–[13].

Such DGA classifiers need to be sufficiently robust so that they can still reliably detect DGA domains even when the DGAs start generating lists from seeds that were not seen during training. Existing work in this area is comprised both of methods that make use of human-engineered features as well as deep learning techniques which learn to extract relevant features automatically. We show in this paper that both kinds of methods are inherently vulnerable to simple attacks and hence the use of side information may be crucial to developing robust DGA classifiers. Specifically, we introduce a new and effective DGA called *CharBot*. It is a simplistic character-based DGA (hence the name) that generates domain names by randomly modifying two characters in well known benign domains collected from the Alexa top domain names.[1] We find that the domains CharBot generates are almost always unregistered, hence available for C&C communication.

To demonstrate CharBot's capabilities, we attack two types of recently proposed prototypical DGA classifiers that are considered state-of-the-art at the time of this writing: a random forest (RF) model called *FANCI* based on human-engineered features extracted from the domain name [11] and a deep neural network (DNN) model called *LSTM.MI* [5]. We also test a RF approach called *B-RF* based on the features proposed in [12]. We train these models on data sets consisting of benign and malicious domain names. The benign names originate from the Alexa top domain names. For the malicious domains, we use the OSINT Bambenek Consulting feeds.[2] We find that the domain names generated by CharBot go largely undetected by all these state-of-the-art DGA classifiers.

We attempt to harden the classifiers against CharBot by incorporating samples from it in the training data sets and retraining the models. Although this strategy does increase the detection rates, they are still not high enough to be practical. We also try retraining using samples generated by DeepDGA — a state of the art generative model for malicious domain names [14] — as well as the DeceptionDGA by Spooren *et al.* [15], but we find that this does not adequately help with detecting CharBot. CharBot is much simpler than both DeepDGA and DeceptionDGA: DeepDGA is a deep learning approach, whereas CharBot performs only simple string manipulations; DeceptionDGA is designed to evade classifiers based on human-engineered features. By contrast, CharBot is fully black-box: it does not require any details of the models being attacked.

CharBot works by corrupting domain names from the Alexa top domains, so it is natural to ask whether the domains it generates can also be used to successfully attack DGA classifiers that do not depend on Alexa for training. To answer this question, we investigate whether the DGA classifiers can be hardened by replacing the Alexa data set by an alternative data set of benign domains during training. To this end, we use a data set of domain names that occurred in real DNS traffic, weakly labeled according to heuristic rules [7]. We find that training on this different data set yields approximately the same results as when training on Alexa. This supports the idea that CharBot attacks are transferable across models and data sets.

These findings expose a dangerous weakness in modern DGA classifiers: they can be circumvented using a simple algorithm and they cannot be easily trained to detect it well. We speculate that this weakness is inherent in any model that relies solely on domain name strings to perform DGA classification. CharBot works by introducing a small number of typographical errors in benign domain names from the Alexa data set. As such, the statistical properties of the names it generates will be almost identical to those of the Alexa domains. This makes it nearly impossible for a classifier to draw any significant distinction between Alexa names and CharBot names. Moreover, any other set of legitimate domains that should be accepted by a classifier with high probability could in principle be used instead of Alexa by a CharBot attack. Therefore, we do not believe these attacks can be mitigated without relying on additional side information. Such information might include the IP addresses the domains resolve to, how many times the domains were queried and when, etc. This has been explored in other works already [3], [10], [16]–[18]. To our knowledge, we are the first to expose this type of weakness in DGA classifiers that do not use side information. We would, therefore, recommend that the community focuses its research efforts on DGA classifiers that utilize side information and not just rely on the domain name string by itself.

The rest of this paper is structured as follows. Section II gives an overview of related work in the field of adversarial machine learning. Section III details the CharBot algorithm. Section IV describes the data sets we used for the experiments. Section V outlines our experiments and discusses their results, as well as several ways we could defend against CharBot attacks. Section VII concludes the work and lists some possibilities for future research.

II. RELATED WORK

Machine learning 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], [11], [16] and ''featureless'' methods in which the

[1] https://alexa.com/topsites. Accessed: 2019-02-10.
[2] http://osint.bambenekconsulting.com/feeds/. Accessed: 2019-02-10.

automatic discovery of good features is part of the overall machine learning model training process, as a form of representation learning [4]–[7], [9], [19]. 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 [4]–[8], [12], [13], [19], and the featureless approach is typically reported to yield better, more accurate results.

A natural response of malware authors to machine learning classifiers for DGA detection is to try to purposely craft domain names that will be mislabeled as benign by the classifiers. This kind of evasion attack is studied as part of the broader field of *adversarial machine learning* (AML) [20]. In this setting, an intelligent adversary aims to exploit weaknesses in a machine learning model in order to obtain desired (illegitimate) outcomes. A prototypical example is that of spam classification, where the adversary attempts to craft spam e-mails that evade detectors while still achieving the desired results. Seminal contributions in this area include the work of Dalvi *et al.* [21] as well as the papers by Lowd and Meek [22], [23], who study classical machine learning algorithms such as linear classifiers, naive Bayes, support vector machines and maximum entropy filters. More recent works primarily study AML for deep neural networks [24]–[26].

A recent innovation in the area of deep learning and generative modeling, is the *Generative Adversarial Network* or GAN, first proposed by Goodfellow *et al.* [27]. In the GAN framework, a generative model is trained by pitting it against an adversary. The adversary is a discriminative model whose goal is to discern whether a given sample came from the data generating distribution or from the generative model. The generator is trained to maximize the loss of the discriminator, so the GAN training procedure corresponds to a two-player minimax game. Ideally, when the training converges, the generator should recover the data generating distribution and the discriminator should not be able to do any better than random guessing.

GANs have found several uses in cybersecurity by now. Anderson *et al.* [14] proposed *DeepDGA*, which is a generative model for DGA domains trained using a GAN. They find that adding samples from DeepDGA to the training data of deep learning based DGA classifiers improves their performance against unseen malware families, aiding generalization of the models when insufficient training data is available. In the field of password security, Hitaj *et al.* [28] have proposed *PassGAN*, another generative model trained in the GAN framework. PassGAN learns to capture the distribution of human passwords and is able to surpass state of the art tools for password guessing. Hu and Tan [29] recently proposed *MalGAN*, a GAN with which they are able to construct malware samples that can bypass black-box machine learning methods. Their attack is particularly striking because they are able to reduce malware detection rates to almost zero without requiring direct access to the detectors they aim to evade. Moreover, they found that explicitly retraining the detectors on MalGAN samples is ineffective: MalGAN can easily be adapted to take this retraining into account, bypassing the retrained models again with almost 100% success. With CharBot, we achieve similar (and, in several cases, better) results with a much simpler approach that can actually be incorporated within a piece of malware, in contrast to deep-learning based methods which are usually too large or too computationally intensive.

Several authors have recently looked into the automatic generation of URLs for phishing. To this end, Bahnsen *et al.* [30] create a text consisting of known phishing URLs from PhishTank[3] and use it to train an LSTM for text generation, i.e. given a small seed sentence, predict the next characters iteratively. They report that this technique generates examples that are not detected by their own LSTM based phishing URL classifier [31]. Anand *et al.* [32] trained a GAN – containing a character based LSTM as part of its architecture – to generate synthetic phishing URLs to augment the training data for feature-based phishing URL detection classifiers. The problem they address is the class imbalance in typical training data sets, which contain many more examples of benign URLs than of phishing URLs. Instead of adding all generated phishing URLs as positive examples to their training data, they first map the generated URLs to their corresponding feature vectors, and select ''representative samples'' based on Euclidean distance in this feature space. In a similar vein to Anand et al., Burns and Heath [33] train a GAN on OpenPhish,[4] PhishTank and DNS-BH[5] data sources to develop synthetic phishing domains. They compare a random forest classifier trained on Alexa and Umbrella[6] data sets to models that were augmented with samples generated by the GAN. They find that the augmented models appear to have consistently higher test set accuracy than the original classifier.

URLs intended for phishing are quite different in nature than DGA domains for C&C purposes. Indeed, to be successful, phishing URLs need to deceive humans, which requires them to be as indistinguishable as possible from benign URLs to the human observer. DGA domain names used for C&C purposes are not intended at all to be read by human users. DGA domain names are successful if they can evade DGA classifiers and have not been previously registered, i.e. they should be available for the botmaster to register. To the best of our knowledge, so far Anderson et al. are the only ones who have looked into generative modeling of DGA domain names [14]. Although their results are significant, we show in this work that classifiers which have been adversarially

[3]https://www.phishtank.com/. Accessed: 2019-02-08.
[4]https://openphish.com/. Accessed: 2019-02-08.
[5]http://www.malwaredomains.com/. Accessed: 2019-02-08.
[6]http://s3-us-west-1.amazonaws.com/umbrella-static/index.html. Accessed: 2019-02-08.

trained using DeepDGA remain vulnerable to simple attacks such as the CharBot algorithm we propose in section III.

The CharBot algorithm is a *black-box targeted evasion attack* that works against tree ensembles and neural networks. ''Black-box'' refers to the fact that our CharBot DGA does not require details of the classifiers in order to work: it can attack any model trained on any data set and succeed with high probability. It is a targeted attack because we want the classifiers to output a specific class in response to our DGA samples, namely *benign*. Untargeted attacks, on the other hand, merely aim to change the classification to *any* class other than the original; for example, an untargeted attack would also count a change from benign to malicious as a success, whereas in our scenario that would be unacceptable. Finally, CharBot is an evasion attack because it occurs at test time, when the model is already trained and deployed. This is in contrast to *poisoning attacks* which occur at training time and work by corrupting samples in the training data set in order to deliberately introduce weaknesses into the model [20]. The CharBot attack itself is inspired by *typosquatting*, a well-known technique used by phishers and social engineers [34]. Typosquatting involves taking a legitimate domain and introducing a few typographical errors that are unlikely to be noticed by human users (e.g., changing *google.com* into *g0ogle.com*). Whereas *a priori* one would think that typosquatting exploits an inherently human vulnerability, we make the surprising discovery that state-of-the-art DGA classifiers are actually vulnerable to such techniques as well.

Similarly to our work here, Spooren *et al.* [15] developed *DeceptionDGA*, a novel DGA which incorporates knowledge of the features used by a DGA classifier in order to attack it. They report significant reductions in predictive accuracy for the FANCI model as well as the Endgame LSTM by Woodbridge *et al.* [4]. The DeceptionDGA algorithm is more complicated than CharBot, requiring knowledge of the underlying model in order to deploy it. Despite this difference in complexity, the detection rates we observe for CharBot in our experiments are comparable to those of DeceptionDGA.

We also wish to acknowledge the concurrent work of [35] who describe *MaskDGA*, a black-box technique for evading DGA classifiers that is similar to CharBot. MaskDGA makes use of a surrogate model as well as a list of DGA domains. It uses these data to craft character-level perturbations of the malicious domains such that they are no longer recognized by the surrogate model. Similarly to our own results, the authors find that such techniques are highly effective at reducing the accuracy of state-of-the-art DGA classifiers. They also make the recommendation that DGA classifiers should rely on additional side-information whenever this is possible in order to mitigate adversarial attacks.

III. CHARBOT

CharBot is a character-based DGA intended to show how successful a simplistic DGA based on small perturbations can be at evading detection by state-of-the-art classifiers. Without loss of generality, throughout this paper we consider domains consisting of a second-level domain (SLD) and a top-level domain (TLD), separated by a dot, as in e.g. *wikipedia.org*. CharBot requires the following inputs:

- A list of legitimate domain names. In our case, ten thousand Alexa domains with a second-level domain (SLD) length of six or greater are used.
- A list of top-level domains (TLDs).
- A date to be used as a seed for pseudorandomization.

With these inputs, CharBot (1) selects a domain from the provided list, (2) selects two characters from the SLD, and (3) selects two replacement characters. The replacement characters are chosen from an equal distribution of DNS-valid characters — the alphanumeric characters and the dash — and the algorithm ensures the characters selected from the SLD are different from the replacement characters. Finally, CharBot (4) appends a TLD to the new domain by selecting one of the following: *com, at, uk, pl, be, biz, co, jp, cz, de, eu, fr, info, it, ru, lv, me, name, net, nz, org, us*. Pseudocode is given in algorithm 1.

Algorithm 1 CharBot

Data: a list of SLDs D, a list of TLDs T, a seed s
Result: a DGA domain
1 Initialize the pseudorandom generator with the seed s.
2 Randomly select a SLD d from D.
3 Randomly select two indices i and j so that $1 \leq i, j \leq |d|$.
4 Randomly select two replacement characters c_1 and c_2 from the set of DNS-valid characters.
5 Set $d[i] \leftarrow c_1$ and $d[j] \leftarrow c_2$.
6 Randomly select a TLD t from T.
7 **return** $d.t$

A DGA is successful if it can generate many unique domains that have not yet been registered and which are not flagged by DGA classifiers as malicious. CharBot draws its replacement characters from a uniform distribution. Therefore, the more characters we replace, the more the generated domains resemble random strings. This increases the detection rate by DGA classifiers, so we aim to keep the number of replacement characters minimal. We tested several choices for the number of characters to be replaced. We found that two characters strike an appropriate balance between the rate of detection by DGA classifiers and the probability that a domain is already registered: with two characters, domains are flagged slightly more often but almost all domains are unregistered (see table 1); when replacing only a single character, detection rates go down but more domains turn out to be registered already.

Adversarial attacks such as CharBot are always accompanied by an *adversarial cost function* $c(x, \tilde{x})$ which describes the cost associated with perturbing an ''ideal'' sample x into a sample $\tilde{x}$ that the adversary can actually use. For image classification, it is common to use ℓ_p distances for this

purpose [24], [26], [36]. However, in our context, the cost of perturbing a correctly classified benign domain x into a malicious domain $\tilde{x}$ that is classified as benign must be measured differently, as we are working in a discrete input space (text) instead of a continuous one (images). Specifically, it makes sense to define our cost function as follows:

$$c(x, \tilde{x}) = \begin{cases} d_L(x, \tilde{x}) & \text{if } \tilde{x} \text{ is unregistered,} \\ \infty & \text{otherwise.} \end{cases}$$

Here, d_L denotes the *Levenshtein distance* or *edit distance* [37]. The cost function $c(x, \tilde{x})$ increases with the number of edits (insertions, deletions, and substitutions) required to transform x into $\tilde{x}$, as each edit makes the attack more detectable by DGA classifiers. However, there is an infinitely large cost associated with generating a domain that is already registered, since such domains cannot be used by the attacker at all and may cause the malware to malfunction. CharBot was designed to minimize this cost function efficiently and as simply as possible.

We note that obvious extensions of the CharBot algorithm are possible. For example, we could additionally implement insertions of new characters and deletions of existing characters. The number of characters might also be chosen adaptively based on some heuristic instead of fixed in advance. However, we chose to limit ourselves to only substituting a fixed number of characters since this simple strategy already gives us very good results. Moreover, simpler attacks are likely to be preferred by attackers and therefore constitute a greater security concern.

The only obstacle to the deployment of CharBot in real malware might be its size. We implemented CharBot in 17,983 bytes of Python code. The Alexa data set it requires takes up 145,008 additional bytes, although the public availability of this data set means it could be downloaded on the fly. Therefore, we would need at most 162,991 bytes for a full implementation of CharBot with Alexa included. By comparison, the DeepDGA algorithm [14] requires to embed in the malware a trained machine learning model that takes up at least 6,539,192 bytes. This is about 40× larger than CharBot. We therefore feel that file size is no obstacle to deploying CharBot in real malware.

IV. DATA SETS

We use three different kinds of data in our experiments:

A. ALEXA

The top 1 million unique domain names from Alexa.[7] Alexa ranks websites based on their popularity in terms of the number of page views and number of unique visitors. It only retains the websites' SLD and TLD, aggregating across any subdomains. For example, according to Alexa, the five highest ranked domain names in terms of popularity on 2019-02-06 are *google.com*, *youtube.com*, *facebook.com*, *baidu.com* and *wikipedia.org*. It is generally assumed that the top 1 million domain names in the Alexa ranking are "benign" domain names in the sense that they were not created by a DGA. Of course, this does not mean that the domain is "benign" in the sense of not being used for malicious activity. Indeed, there is reason to believe that a significant number of Alexa top ranked domains are used for malicious purposes [38], but this is not the problem we are considering here. In our setting, we only consider a domain to be "malicious" if it was generated by a DGA.

B. BAMBENEK

1 million unique DGA domain names from the Bambenek Consulting feeds[8] for 3 different days, namely Jun 24, Jul 22, Jul 23, 2017. These feeds contain DGA domain names from specific malware families that were observed in real traffic on those days. Such domain names can be collected by reverse engineering a known malware family, generating lists of domain names with the reverse engineered malware, and checking which of these domain names also occur in real traffic.

C. QNAME

1 million unique domain names originating from a real-time stream of passive DNS data that consists of roughly 10-12 billion DNS queries per day collected from subscribers including ISPs (Internet Service Providers), schools, and businesses. We annotated this stream based on a set of heuristic filtering rules following [7]. Specifically, we labeled as benign all domains that have been resolved at least twice, never resulted in an NXDomain response and span more than 30 days. Here, span is defined as the number of days between the first and last successfully resolved query for a given domain. We randomly sampled 1 million such domains that appeared in DNS traffic between September 2015 and August 2018. This data set is *weakly labeled* since the heuristic filtering rules do not guarantee that the domains are actually benign or malicious; however, we believe it to be a useful approximation.

The Alexa and Qname data sets serve as our negative (benign) examples, whereas Bambenek serves as our set of positive (malicious) examples. Alexa and Qname have precisely 537 domains in common, which is a negligible number compared to the total sizes of the data sets, therefore making Qname a good data set to test transferability of CharBot.

We refer to the combination of Alexa and Bambenek data as *AlexaBamb* and similarly for *QnameBamb*. These data sets consist of 2 million samples each, 1 million per class.

V. EXPERIMENTS

We perform experiments on two DGA classifiers that are considered state of the art at the time of this writing: FANCI [11] and LSTM.MI [5], as well as a third model we call B-RF based on the work by [12]. All classifiers are trained to label a domain name as either benign (negative) or

[7] https://www.alexa.com, Accessed 2019-02-08.

[8] http://osint.bambenekconsulting.com/feeds/, Accessed 2019-02-08

TABLE 1. Adversarial data sets.

DGA	Data Set	Seeds Used	# Unique Synthetic Domains	# Unregistered Domains (out of 500 sampled)
CharBot	Training	2018-12-04	100,000	500 (100%)
	Testing	2019-01-01	10,000	500 (100%)
DeepDGA	Training	2018-12-04	100,000	499 (99.8%)
	Testing	2019-01-01	10,000	499 (99.8%)
DeceptionDGA	Training	N/A	100,000	494 (98.8%)
	Testing	N/A	10,000	494 (98.8%)

malicious (positive). We find that the best results overall are achieved with the deep learning based LSTM.MI approach, followed by the random forest-based B-RF approach, and finally the random forest-based FANCI method. The difference in predictive accuracy between the various approaches is substantial. The results hold across the AlexaBamb and QnameBamb data sets (see table 4, and a more detailed discussion in section V-D).

To arrive at the results, we train on the AlexaBamb data set as well as on the QnameBamb data set with a 80%/20% train/test split for each, reporting the true positive rate (TPR), the partial area under the ROC curve (AUC) and the fraction of samples from CharBot, DeepDGA and DeceptionDGA which the models were able to detect (see table 4 and table 5). All of these metrics are reported at FPRs of 0.1% and 1%.[9] The AUC@0.1%FPR is the integral of the ROC curve from FPR $= 0$ to FPR $= 0.001$ on the test data, and similarly for the AUC@1%FPR. We repeat all experiments on the original models as well as the models after adversarial retraining.

To perform the adversarial retraining, we utilized the data sets shown in table 1. Specifically, we used CharBot and DeepDGA with different seeds to generate training and testing data sets. The training data sets were used to augment the original training data of the classifiers; the testing data sets were used to verify their performance. For DeceptionDGA, Spooren et al. [15] supplied a list of 150,000 domains generated by their algorithm from which we sampled our training and testing data. Note that, based on a random sample of 500 domains,[10] CharBot has the highest fraction of unregistered domains (100%), followed by DeepDGA (99.8%) and DeceptionDGA (98.8%).

The experiments on the QnameBamb data set are intended to investigate the transferability of CharBot. All CharBot domain names used in the experiments (see table 1) are created by CharBot by corrupting domain names from the Alexa data set. This might leave DGA classifiers that are trained on AlexaBamb extra vulnerable to CharBot attacks. A natural question to ask is whether CharBot can also successfully bypass DGA classifiers that were trained on a data set different from Alexa, one CharBot has no access to. To test this, we trained LSTM.MI, FANCI, and B-RF on the QnameBamb data and reported the same statistics as for AlexaBamb.

Below we give a brief description of the LSTM.MI, FANCI, and B-RF classifiers, followed by detailed results (section V-D) and a discussion of possible countermeasures for defending against small perturbations attacks such as CharBot (section VI).

A. LSTM.MI

Woodbridge *et al.* [4] were the first to propose deep learning for DGA domain name detection. Their DGA classifier is a neural network consisting of an embedding layer, an LSTM layer, and a single node output layer with sigmoid activation. In this paper, we use the LSTM.MI model that was proposed recently by Tran *et al.* [5]. Its architecture is very similar to that of Woodbridge *et al.* [4]; the main distinction is that the LSTM.MI model is trained with a cost-sensitive learning algorithm that takes class imbalances into account. This allows the LSTM.MI approach to achieve slightly better results than the original LSTM approach (see [5], [12]). The code for training the LSTM.MI model is publicly available.[11]

B. FANCI

The FANCI classifier recently proposed by Schüppen *et al.* [11] is a random forest (RF) classifier designed to classify NXDomains as benign (bNXD) or malicious (mAGD). NXDomains, or Non-Existent Domains, are domains that can not be resolved. DGAs generate hundreds or even thousands of domains every day, only very few of which are actually registered by the botmaster. That means that almost all queries for DGA generated domains by infected machines will result in an NXDomain response by the local DNS server, so it is reasonable to attempt to detect DGA activity by analyzing NXDomains.

To this end, the FANCI classifier leverages 21 manually defined features, extracted from the domain name string. The 21 features can be divided into structural, linguistic, and statistical categories (see table 2). The FANCI RF model is comprised of 9 decision trees, of which 7 use the Gini coefficient as the measure of impurity and the other 2 use

[9] A low false positive rate is very important in deployed DGA detection systems because blocking legitimate traffic is highly undesirable. The threshold of 0.1% FPR was chosen because this rate is often used by real-world models in practice, whereas 1% is the largest FPR that could still be useful.

[10] We limited ourselves to a random sample of 500 domains to avoid getting blocked by ISPs.

[11] https://github.com/bkcs-hust/lstm-mi. Accessed: 2019-02-08.

TABLE 2. Features used by FANCI and B-RF. (*) For these features, FANCI uses dot free public-suffix-free domain. (**) For these features, FANCI uses public-suffix-free domain.

#	Feature	FANCI	B-RF
1	Domain name length	✓	✓
2	Second level domain length	✗	✓
3	Top level domain length	✗	✓
4	Domain Unique Characters length	✗	✓
5	SLD Unique Characters length	✗	✓
6	TLD Unique Characters length	✗	✓
7	Has malicious TLD	✗	✓
8	Has Valid TLD	✓	✗
9	TLD Hash	✗	✓
10	Contains Digits	✓	✗
11	Starts with Digit	✗	✓
12	Underscore Ratio*	✓	✗
13	Symbol ratio	✗	✓
14	Hex ratio	✗	✓
15	Digit Ratio*	✓	✓
16	Vowel Ratio*	✓	✓
17	Consonant Ratio	✗	✓
18	Ratio of Repeated Characters*	✓	✓
19	Ratio of Consecutive Consonants*	✓	✓
20	Ratio of Consecutive Digits*	✓	✓
21	Number of tokens in SLD	✗	✓
22	Number of digits in SLD	✗	✓
23	Entropy*	✓	✓
24	Gini Index	✗	✓
25	Classification error of characters	✗	✓
26	N-Gram Distribution*	✓	✗
27	2-Gram Median	✗	✓
28	3-Gram Median	✗	✓
29	2-Gram Circle Median	✗	✓
30	3-Gram Circle Median	✗	✓
31	Number of Subdomains**	✓	✗
32	Subdomain Length Mean**	✓	✗
33	Has www Prefix	✓	✗
34	Contains Single-Character Subdomain**	✓	✗
35	Is Exclusive Prefix Repetition	✓	✗
36	Contains TLD as Subdomain**	✓	✗
37	Ratio of Digit-Exclusive Subdomains**	✓	✗
38	Ratio of Hexadecimal-Exclusive Subdomains**	✓	✗
39	Contains IP Address**	✓	✗
40	Alphabet Cardinality*	✓	✗

TABLE 3. FANCI features not expected to have any effect in our experiments.

#	Feature
31	Number of Subdomains
33	Has www Prefix
34	Contains Single-Character Subdomain
35	Is Exclusive Prefix Repetition
36	Contains TLD as Subdomain

entropy. Each tree takes between 2 to 18 features. The source code of the FANCI classifier is available on GitHub.[12]

The domain names used in our experiments contain only SLDs and TLDs (see section IV). As such, it is expected that a number of features used in the FANCI model would not make a distinction between malicious and benign examples. Table 3 lists the FANCI features that are not expected to have any effect.

[12]https://github.com/fanci-dga-detection/fanci. Accessed: 2019-02-08.

C. B-RF

B-RF [12] is a random based DGA detection classifier that is trained on 26 manually engineered features as indicated in table 2. There is some overlap between the features used by FANCI and those used by B-RF. For instance, both make use of the domain name length, digit and vowel ratio, ratio of repeated characters, etc. Some features are used by FANCI but not by B-RF, such as whether the domains have valid TLDs or whether they contain digits. Other features like 2-gram median and 3-gram median are only used by B-RF.

B-RF consists of 100 trees and each tree is trained using a subset with a maximum of 20 features. Entropy is used as the criterion to decide the split attribute while growing the trees in the random forest.

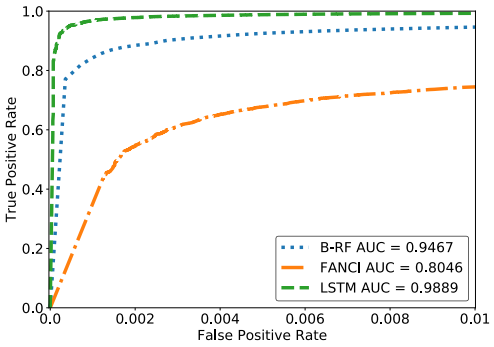


FIGURE 1. ROC curves for the classifiers trained on the AlexaBamb data set.

D. RESULTS

The predictive performance metrics are summarized in table 4 for false positive rates[13] of 0.1% and 1%. Figure 1 shows ROC curves for the different models on the AlexaBamb data. We plot the ROC curve only for FPRs between 0 and 0.01, as higher FPRs are meaningless in practice. We conclude from these results that the deep learning approach does better than the RF approaches, which is in line with what has been reported before in the literature [4]–[7], [13]. Among the RF models, B-RF outperforms FANCI significantly. We found that this improvement was not due to the number of trees, as decreasing the number of trees used by B-RF from 100 to 9 (as in FANCI) still yielded superior performance for B-RF. We therefore believe this difference in performance is caused by the different feature sets.

We were unable to establish a classification threshold that achieves 0.1% FPR for FANCI. Therefore, in reporting FANCI results, we only consider FPR $=$ 1%. We believe this is due to the fact that [11] used proprietary data to filter classification outcomes which increased accuracy. Our use of a different post-filtering data set may be the

[13]One can argue that even a FPR of 0.1% is still too high to be useful in practice. While this can certainly be true depending on the application, note that lower FPRs can only make our results better as the models will necessarily have lower TPR and lower detection rates for CharBot.

TABLE 4. Performance metrics of LSTM.MI, FANCI and B-RF on the different data sets.

Classifier	Data set	FPR=0.001		FPR=0.01	
		TPR	AUC	TPR	AUC
LSTM.MI	AlexaBamb	96.79%	94.91%	99.27%	98.89%
	AlexaBamb + CharBot	95.50%	95.35%	98.89%	98.67%
	AlexaBamb + DeepDGA	96.65%	96.44%	99.20%	99.00%
	AlexaBamb + DeceptionDGA	95.17%	95.05%	98.54%	98.46%
	QnameBamb	81.98%	83.37%	98.98%	96.68%
	QnameBamb + CharBot	82.91%	84.98%	98.51%	96.48%
	QnameBamb + DeepDGA	83.22%	83.98%	98.85%	96.50%
	QnameBamb + DeceptionDGA	84.66%	85.57%	98.61%	96.82%
FANCI	AlexaBamb	—	—	74.46%	80.46%
	AlexaBamb + CharBot	—	—	72.49%	78.71%
	AlexaBamb + DeepDGA	—	—	73.98%	80.02%
	AlexaBamb + DeceptionDGA	—	—	73.84%	80.18%
	QnameBamb	—	—	74.13%	79.06%
	QnameBamb + CharBot	—	—	72.13%	77.80%
	QnameBamb + DeepDGA	—	—	72.89%	78.19%
	QnameBamb + DeceptionDGA	—	—	73.65%	78.86%
B-RF	AlexaBamb	85.72%	82.93%	94.72%	94.67%
	AlexaBamb + CharBot	84.62%	79.30%	93.89%	93.75%
	AlexaBamb + DeepDGA	85.81%	80.94%	94.39%	94.35%
	AlexaBamb + DeceptionDGA	84.51%	78.62%	93.78%	93.73%
	QnameBamb	82.75%	73.85%	96.88%	94.52%
	QnameBamb + CharBot	83.03%	74.98%	96.37%	94.29%
	QnameBamb + DeepDGA	82.64%	73.87%	96.39%	94.26%
	QnameBamb + DeceptionDGA	82.98%	76.26%	96.26%	94.56%

TABLE 5. Detection rates of the different DGAs.

Classifier	Data set	FPR=0.001			FPR=0.01		
		CharBot	DeepDGA	DeceptionDGA	CharBot	DeepDGA	DeceptionDGA
LSTM.MI	AlexaBamb	5.58%	33.98%	4.02%	15.50%	39.53%	12.74%
	AlexaBamb + CharBot	55.19%	92.54%	19.69%	81.08%	98.44%	47.67%
	AlexaBamb + DeepDGA	12.39%	98.35%	7.34%	12.39%	98.35%	7.34%
	AlexaBamb + DeceptionDGA	23.59%	88.71%	40.29%	52.18%	96.66%	71.52%
	QnameBamb	15.25%	6.18%	16.61%	31.90%	19.51%	37.73%
	QnameBamb + CharBot	52.67%	42.48%	34.45%	81.96%	85.90%	66.27%
	QnameBamb + DeepDGA	27.84%	94.51%	24.25%	43.28%	99.61%	47.33%
	QnameBamb + DeceptionDGA	30.45%	15.97%	37.74%	53.31%	37.97%	24.25%
FANCI	AlexaBamb	—	—	—	3.05%	6.33%	1.66%
	AlexaBamb + CharBot	—	—	—	22.26%	12.12%	2.64%
	AlexaBamb + DeepDGA	—	—	—	6.45%	83.17%	2.08%
	AlexaBamb + DeceptionDGA	—	—	—	4.27%	6.57%	2.33%
	QnameBamb	—	—	—	21.43%	5.37%	46.85%
	QnameBamb + CharBot	—	—	—	48.44%	14.20%	49.62%
	QnameBamb + DeepDGA	—	—	—	45.13%	77.88%	50.11%
	QnameBamb + DeceptionDGA	—	—	—	44.75%	13.77%	50.45%
B-RF	AlexaBamb	1.69%	6.61%	1.38%	27.59%	23.97%	22.37%
	AlexaBamb + CharBot	1.84%	9.19%	1.20%	64.33%	34.06%	41.22%
	AlexaBamb + DeepDGA	4.54%	46.55%	2.86%	31.80%	84.12%	26.86%
	AlexaBamb + DeceptionDGA	2.00%	8.23%	1.34%	33.14%	24.51%	32.53%
	QnameBamb	18.80%	2.99%	41.67%	61.05%	22.57%	62.31%
	QnameBamb + CharBot	43.47%	12.54%	49.18%	85.82%	60.70%	79.33%
	QnameBamb + DeepDGA	44.04%	33.08%	49.80%	65.97%	98.84%	67.79%
	QnameBamb + DeceptionDGA	39.97%	10.53%	47.86%	62.29%	23.30%	67.74%

reason for the difference (note that the authors of the FANCI paper use accuracy whereas we use AUC). For AlexaBamb and its augmented datasets, we obtained between 90.7% and 91.4% accuracy. For QnameBamb and its augmented datasets, we obtained between 92.5% and 93.16% accuracy. This is not too far from the 93.7% that was reported in [15]

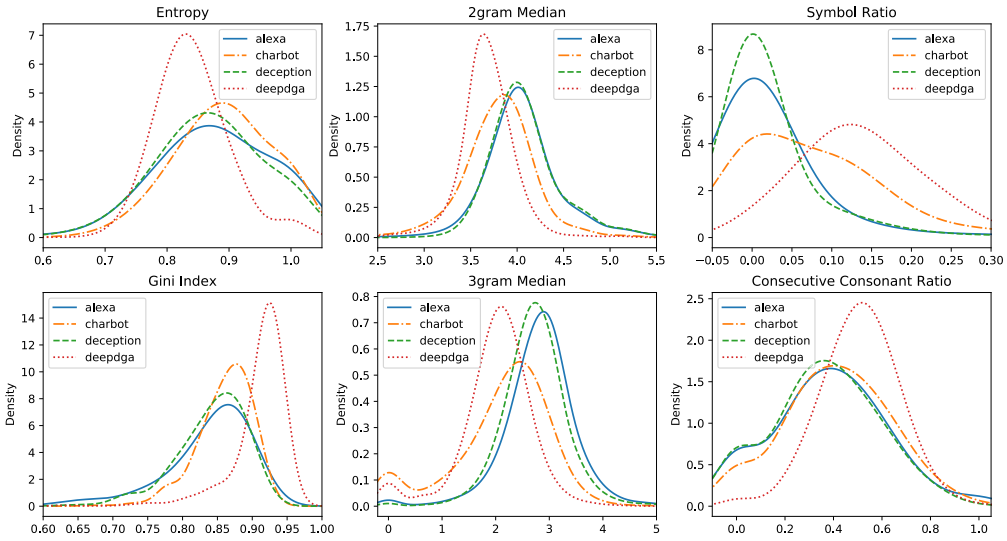


FIGURE 2. Kernel density estimation of the feature distributions of Alexa and adversarial domain names.

where the authors also attempted to replicate the FANCI results.

All models fail to adequately detect CharBot and DeceptionDGA domains even when explicitly trained on them. The LSTM.MI model succeeds in detecting DeepDGA close to 99% of the time with adversarial training, but the other models generally fail at detecting DeepDGA as well. Training on Qname instead of Alexa makes a significant difference, both in predictive performance as well as detection rate: the models have lower predictive accuracy when trained on Qname, but they are better able to detect CharBot domains. At the 0.1% FPR, however, these detection rates are nowhere near high enough to be useful in practice. FANCI is unable to properly detect CharBot at 1% FPR, whereas LSTM.MI and B-RF sometimes manage to obtain over 80% detection rate here. This is not a very useful result, however, since 1% FPR is considered too high to be practical. Therefore, at a low FPR, the domains generated by CharBot can be said to be *transferable* across different models and data sets in the sense that CharBot can fool models that have vastly different architectures and are not trained on Alexa. Combined with its simplicity, speed and small size, this makes CharBot an ideal DGA for use in malware in the wild.

The success of CharBot may be explained as follows. The algorithm works by taking the Alexa list of benign domains — which most DGA classifiers would overwhelmingly classify as such — and introduces a small number of typographical errors. The statistical properties of the domains generated by CharBot are therefore likely almost identical to those of Alexa, causing a low detection rate. The transferability may be explained by noting that even though Alexa and Qname are different data sets, they still capture the same underlying distribution: namely, that of benign domains. This closeness in distribution is most likely shared among all sufficiently large corpora of benign domains, allowing CharBot to fool any DGA classifier that only takes the domain name string into account. We test this hypothesis by performing kernel density estimation on the feature distributions of the Alexa domains and the adversarial domains. The results are plotted in figure 2. The Entropy and Gini index features are standard impurity measures for decision trees. The other features are:

- 2gram Median. This feature takes the median frequency from the list of 2gram frequencies for the given SLD. Bigram frequencies are collected from the Python package called wordfreq. [14]
- 3gram Median. 3gram median is similar to 2gram median except that it returns the median frequency from the list of trigram frequencies for the given SLD.
- Symbol ratio. This feature defines the ratio of non-alphabetical characters in the SLD, which includes digits and special characters.
- Consecutive Consonant Ratio. This feature defines the ratio of consecutive consonants in the SLD.

From the plots, we observe that the feature distributions of CharBot domains are much closer to those of Alexa than the distributions of DeepDGA are. However, DeceptionDGA is more similar to Alexa than CharBot is, although the difference is very small in some cases. Nevertheless, CharBot gets quite close to Alexa, which explains why it is so successful in fooling DGA classifiers. It also shows that defending against CharBot may be very difficult, potentially requiring a very high FPR. Figure 2 also provides insights into what

[14]https://pypi.org/project/wordfreq/1.1/. Accessed: 2019-02-14.

parts of CharBot may be improved to yield an even more effective DGA:

- The entropy curve of CharBot can be made more similar to that of Alexa domains by using a different replacement character distribution. Currently, we are using the uniform distribution, which has the highest possible entropy. Switching to a lower entropy distribution may improve the performance of CharBot, although this would need to be carefully balanced against the probability of a generated domain already being registered.
- The random replacement of two characters caused the 2-gram distributions of CharBot to differ from those of the Alexa domains. We can overcome this weakness by replacing neighboring characters with 2-grams that occur frequently in Alexa. A similar line of reasoning applies to 3-grams.
- The symbol ratio distributions can be made more similar by drawing c_1 and c_2 from the same letters or digit sets of the original domain. For example, replace a digit with another random digit and not with a letter.

TABLE 6. Statistics of the domain name lengths for each data set.

Data Set	Mean	Standard Deviation
Alexa	14.30	4.70
Bambenek	21.80	6.42
Qname	23.99	7.97
CharBot	14.27	4.02
DeepDGA	28.52	9.06
DeceptionDGA	10.20	3.90

Investigating the lengths of the domain names that were generated vs. those that are present in the Alexa and Qname data sets (see table 6), we find that CharBot names are close in length to Alexa names (which is to be expected), but Qname, Bambenek and DeepDGA domains are significantly longer on average, whereas DeceptionDGA are significantly shorter. This difference in lengths may contribute to the detection rates: when training on Alexa, CharBot domains are similar in length whereas DeepDGA domains are longer like the Bambenek domains. By contrast, when training on Qname, domains are longer on average, which aids detection of CharBot (although the difference is not very large).

VI. COUNTERMEASURES

We consider a few options for defending against attacks such as CharBot:

A. COMPARING INCOMING DOMAINS TO ALEXA

The simplest defense against CharBot would be to take the domain in question and compare it to the full Alexa list. If the domain is equal to one found in the Alexa list save for one or two replaced characters, the domain is flagged as malicious. However, the Alexa data set contains one million samples, so this approach of computing the Hamming distance of input domains on the fly may not be practical. We can make this computation even harder by modifying CharBot to perform deletions and insertions, forcing the use of the edit distance [37] rather than the Hamming one. Practical implementations can reduce lookup time by pre-computing noisy versions of the Alexa list into a compact data structure such as a Bloom filter [39]. However, this approach is marred by a combinatorial explosion of possible corrupted domain names based on the Alexa data set: if we let n be the size of the CharBot data set, ℓ be the average length of a domain name, k be the number of edits CharBot introduces and m be the size of the replacement alphabet, then the number of possible domains CharBot can generate is given approximately by

$$n\binom{\ell}{k}(m-1)^k.$$

For $n = 10,000$, $\ell = 16$, $m = 40$ and $k = 2$ this yields 1,825,200,000 possible domains. Besides, this defense can also easily be defeated by simply using a different legitimate data set instead of Alexa for generating domain names.

B. INCREASING THE CAPACITY OF THE MODELS

Using more complicated classification models may allow them to find a meaningful separation between Alexa and CharBot domains. However, this would require careful feature engineering for featureful models and increase the computational burden of both model training and inference. Given that practical DGA classifiers need to be regularly retrained to keep up with new malware and they need to process many domains in real-time, this may not be feasible. Nevertheless, this may be an option worth exploring in future work.

C. WHITE-BOX ADVERSARIAL TRAINING

Our adversarial training procedure in this paper has consisted of generating a list of adversarial domains once and then augmenting the training data with them. However, adversarial training is usually done iteratively: at every iteration of training, the current batch of training samples is augmented with adversarially generated set specifically for the model at that particular stage [24], [25]. This requires a *white-box* attack which is able to take the model parameters into account. Adversarial attacks have mostly been considered in the image domain, although there is some work on text classification [40], [41]. Making use of this recent body of work on white-box adversarial training for text classification may allow us to improve the detection rate of CharBot.

D. USING SIDE INFORMATION

Perhaps the most realistic defense against attacks like CharBot would be to use additional information besides the domain name string alone. For instance, the IP addresses the domain maps to, how often the domain was queried and when, etc. There have been several works investigating the use of such information in DGA classification [3], [10], [16]–[18]. A fruitful avenue for future work could be to test whether these classifiers are more resilient to CharBot.

VII. CONCLUSION

We have proposed CharBot, a simple and efficient DGA. We have shown CharBot to be effective at both generating large amounts of unregistered domain names as well as fooling three DGA classifiers: FANCI, LSTM.MI and B-RF. We also compared CharBot to DeepDGA and DeceptionDGA, two state-of-the-art domain generation algorithms. The domain names generated by CharBot were more likely to be unregistered than those generated by DeepDGA or DeceptionDGA. Moreover, adversarial retraining using CharBot, DeepDGA or DeceptionDGA did not result in adequate detection of CharBot domains names.

Our DGA is the very first example of a black-box adversarial machine learning attack against DGA classifiers that is not based on Generative Adversarial Networks. We show that simply introducing small perturbations to a set of legitimate domains is good enough and such advanced techniques are unnecessary. We believe this highlights a dangerous weakness of modern DGA classifiers, namely their vulnerability to extremely simple attacks that make no use of sophisticated machine learning techniques. CharBot is an algorithm that could be realistically used in malware in the wild to circumvent state of the art DGA classifiers, making it a real threat. We speculate that this vulnerability is actually *inherent* to any classifier that relies only on the domain name string to perform DGA classification. The CharBot DGA is similar to dictionary DGAs: both have a list of strings embedded as part of the DGA code. In the case of dictionary DGAs this list is a dictionary of words that are combined in various ways to generate a domain name, while in the case of CharBot the list contains benign domain names that are altered slightly to generate a new domain name for malicious purposes. In both cases, the generated domain names exhibit properties that are very close to natural language, which makes them extremely difficult to distinguish from benign domain names.

Machine learning models that attempt to do DGA classification based only on the domain name itself, such as the ones considered in this paper, might not be sufficient to detect a DGA like CharBot. The result highlights the need for 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) [3], [16]–[18], as was done successfully for dictionary DGAs [10].

For future work, we focus on defending DGA classifiers against simple attacks such as CharBot. The avenues we are investigating to achieve this include performing white-box adversarial training as well as augmenting the model inputs with side information that is more difficult to manipulate.

REPRODUCIBILITY

To foster reproducibility of our results, we are open to sharing all of our code as well as data sets of CharBot samples upon request.

ACKNOWLEDGEMENT

The authors would like to thank the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. They thank Bobby Filar for making the code of the original DeepDGA algorithm available to then [14] and Jan Spooren for providing then with domain names generated by DeceptionDGA [15].

REFERENCES

[1] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *Proc. USENIX Security Symp.*, 2016, pp. 263–278. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_plohmann.pdf

[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 *Proc. USENIX Security Symp.*, vol. 12, 2012, pp. 491–506. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final127.pdf

[3] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with DNS traffic analysis," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1663–1677, Oct. 2012. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6151233

[4] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," 2016, *arXiv:1611.00791*. [Online]. Available: https://arxiv.org/abs/1611.00791

[5] 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, Jan. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217317320

[6] P. Lison and V. Mavroeidis, "Automatic detection of malware-generated domains with recurrent neural models," 2017, *arXiv:1709.07102*. [Online]. Available: https://arxiv.org/abs/1709.07102

[7] B. Yu, D. L. Gray, J. Pan, M. D. Cock, and A. C. A. Nascimento, "Inline DGA detection with deep networks," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 683–692. [Online]. Available: https://ieeexplore.ieee.org/document/8215728

[8] C. Choudhary, R. Sivaguru, M. Pereira, B. Yu, A. C. Nascimento, and M. De Cock, "Algorithmically generated domain detection and malware family classification," in *Proc. 6th Int. Symp. Secur. Comput. Commun. (SSCC)*, in Communications in Computer and Information Science. Singapore: Springer, 2018, pp. 640–655. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-5826-5_50

[9] J. J. Koh and B. Rhodes, "Inline detection of domain generation algorithms with context-sensitive word embeddings," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2018, pp. 2966–2971. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8622066

[10] 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 *Proc. 21st Int. Symp. Res. Attacks, Intrusions, Defenses (RAID)*, 2018, pp. 295–314. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-00470-5_14

[11] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "FANCI : Feature-based automated nxdomain classification and intelligence," in *Proc. USENIX Secur. Symp.*, 2018, pp. 1165–1181. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/schuppen

[12] R. Sivaguru, C. Choudhary, B. Yu, V. Tymchenko, A. Nascimento, and M. De Cock, "An evaluation of DGA classifiers," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 5051–5060. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8621875

[13] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character level based detection of DGA domain names," in *Proc. IEEE World Congr. Comput. Intell.*, Jul. 2018, pp. 4168–4175. [Online]. Available: http://faculty.washington.edu/mdecock/papers/byu2018a.pdf

[14] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-tuned domain generation and detection," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2016, pp. 13–21. [Online]. Available: https://dl.acm.org/citation.cfm?id=2996767

[15] J. Spooren, D. Preuveneers, L. Desmet, P. Janssen, and W. Joosen, "Detection of algorithmically generated domain names used by botnets: A dual arms race," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, 2019, pp. 1902–1910. [Online]. Available: https://lirias.kuleuven.be/2361222?limo=0

[16] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based botnet tracking and intelligence," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*, 2014, pp. 192–211. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-08509-8_11

[17] J. Kwon, J. Lee, H. Lee, and A. Perrig, "PsyBoG: A scalable botnet detection method for large-scale DNS traffic," *Comput. Netw.*, vol. 97, pp. 48–73, Mar. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1389128615004843

[18] P. Lison and V. Mavroeidis, "Neural reputation models learned from passive DNS data," in *Proc. IEEE Int. Conf. Big Data*, 2017, pp. 3662–3671. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8258361

[19] J. Saxe and K. Berlin, "eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys," 2017, *arXiv:1702.08568*. [Online]. Available: https://arxiv.org/abs/1702.08568

[20] Y. Vorobeychik and M. Kantarcioglu, *Adversarial Machine Learning* (Synthesis Lectures on Artificial Intelligence and Machine Learning), vol. 12, no. 3. San Rafael, CA, USA: Morgan & Claypool, 2018, pp. 1–169.

[21] N. Dalvi, P. Domingos, S. Sanghai, and D. Verma, "Adversarial classification," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. 2004, pp. 99–108. [Online]. Available: http://s33043.gridserver.com/files/teaching/stanford/2008/readings/AdversarialClassification%20DalviEtAl%20KDD04.pdf

[22] D. Lowd and C. Meek, "Adversarial learning," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 641–647. [Online]. Available: https://www.researchgate.net/profile/Daniel_Lowd/publication/221654486_Adversarial_learning/links/00b7d522999ea43eb4000000.pdf

[23] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," in *Proc. CEAS*, 2005. [Online]. Available: http://www.utdallas.edu/~muratk/courses/dmsec_files/125.pdf

[24] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*. [Online]. Available: https://arxiv.org/abs/1412.6572

[25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*. [Online]. Available: https://arxiv.org/abs/1706.06083

[26] A. Raghunathan, J. Steinhardt, and P. S. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10877–10887. [Online]. Available: https://papers.nips.cc/paper/8285-semidefinite-relaxations-for-certifying-robustness-to-adversarial-examples.pdf

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets

[28] B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz, "PassGAN: A deep learning approach for password guessing," 2017, *arXiv:1709.00440*. [Online]. Available: https://arxiv.org/abs/1709.00440

[29] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, *arXiv:1702.05983*. [Online]. Available: https://arxiv.org/abs/1702.05983

[30] A. C. Bahnsen, I. Torroledo, L. D. Camacho, and S. Villegas, "DeepPhish: Simulating malicious AI," in *Proc. IEEE APWG Symp. Electron. Crime Res.*, 2018, pp. 15–17. [Online]. Available: https://albahnsen.com/wp-content/uploads/2018/05/deepphish-simulating-malicious-ai_submitted.pdf

[31] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, "Classifying phishing URLs using recurrent neural networks," in *Proc. IEEE APWG Symp. Electron. Crime Res.*, Apr. 2017, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7945048

[32] A. Anand, K. Gorde, J. R. A. Moniz, N. Park, T. Chakraborty, and B.-T. Chu, "Phishing URL detection with oversampling based on text generative adversarial networks," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2018, pp. 1167–1176. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8622547

[33] J. Burns and E. Heath, "Using generative adversarial networks to harden phishing classifiers," in *Proc. FloCon*, New Orleans, LA, USA, 2019. [Online]. Available: https://flocon2019.sched.com/event/GXW1/using-generative-adversarial-networks-to-harden-phishing-class

[34] J. Szurdi, B. Kocso, G. Cseh, J. Spring, M. Felegyhazi, and C. Kanich, "The long 'taile' of typosquatting domain names," in *Proc. 23rd USENIX Secur. Symp. (USENIX Security)*, 2014, pp. 191–206. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-szurdi.pdf

[35] L. Sidi, A. Nadler, and A. Shabtai, "MaskDGA: A black-box evasion technique against DGA classifiers and adversarial defenses," 2019, *arXiv:1902.08909*. [Online]. Available: https://arxiv.org/abs/1902.08909

[36] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2574–2582. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Moosavi-Dezfooli_DeepFool_A_Simple_CVPR_2016_paper.pdf

[37] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys.-Dokl.*, vol. 10, no. 8, pp. 707–710, 1966. [Online]. Available: https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf

[38] P. Royal, "Quantifying maliciousness in alexa top-ranked domains," in *Proc. BlackHat*, Seattle, WA, USA, 2012. [Online]. Available: https://media.blackhat.com/ad-12/Royal/bh-ad-12-quanitfying-royal-slide.pdf

[39] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970. [Online]. Available: http://crystal.uta.edu/ mcguigan/cse6350/papers/Bloom.pdf

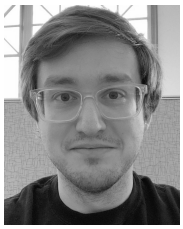
[40] Z. Gong, W. Wang, B. Li, D. Song, and W.-S. Ku, "Adversarial texts with gradient methods," 2018, *arXiv:1801.07175*. [Online]. Available: https://arxiv.org/abs/1801.07175

[41] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2018, pp. 31–36. [Online]. Available: http://www.aclweb.org/anthology/P18-2006

JONATHAN PECK received the B.Sc. degree in computer science and the M.Sc. degree in mathematical informatics from Ghent University, Belgium, in 2015 and 2017, respectively. He is currently pursuing the joint Ph.D. degree with the Department of Applied Mathematics, Computer Science and Statistics and the VIB Inflammation Research Center, Ghent, Belgium. His research is sponsored by a fellowship of the Research Foundation Flanders (FWO) and focuses improving the robustness of machine learning models to adversarial manipulations.

CLAIRE NIE received the B.Eng. degree in chemical engineering from McGill University, Canada, in 2012. She is currently pursuing the M.S. degree in computer science and systems with the University of Washington Tacoma, USA. She was a Financial Auditor of Technology Companies, Silicon Valley. During her graduate studies, she carried out research on training random forest classifiers for the detection of algorithmically generated domain names.

RAAGHAVI SIVAGURU received the B.Tech. degree in information technology from Anna University, India, in 2014. She is currently pursuing the M.S. degree in computer science and systems with the University of Washington Tacoma, USA. Her research interests include improving the predictive performance of DGA detection classifiers and conducting an evaluation of hardening techniques that use side information to defend against adversaries.



CHARLES GRUMER received the B.A. degree in psychology from the University of Washington Seattle, in 2015. He is currently pursuing the M.S. degree in computer science and systems with a specialty in data science, the University of Washington Tacoma. He is slated to graduate at the end of 2019, pending the completion and successful defense of his thesis, which focuses on white box adversarial example generation for DGA classifiers. He is also a Data Science Intern with Infoblox.

FEMI OLUMOFIN received the Ph.D. degree in computer science from the University of Waterloo, Canada. He is currently a Senior Member of the data science and analytics team with Infoblox in the San Francisco Bay Area. He has made contributions to research and development in the areas of privacy enhancing technologies, security, applied cryptography, big data analytics, and machine learning.

BIN YU received the Ph.D. degree in electronic engineering from Tsinghua University, China. He was a Postdoctoral Fellow with the Pattern Recognition and Image Processing Lab, Michigan State University, USA, and an Associate Professor with Beijing Jiaotong University, China. He was with many high tech companies in Silicon Valley at a senior leadership positions and a led projects of machine learning and artificial intelligence for internet search, medical imaging, computer vision, and e-commerce. He is currently a Chief Data Scientist, pioneered big data analytics with Infoblox, Santa Clara, USA, to detect malicious DNS traffic, using deep learning, and artificial intelligence techniques to keep pace with fast changing malware evolution. He has a rich experience in both academia and industry for more than 25 years and has published more than 50 peer-reviewed papers and patents in artificial intelligence, deep learning, machine learning, image processing, and cybersecurity. He served as a Senior Member for the IEEE Computer Society.

ANDERSON NASCIMENTO received the B.S. degree in electrical engineering from the University of Brasilia, Brazil, in 1998, and the M.S. and Ph.D. degrees in information and communication engineering from the University of Tokyo, Japan, in 2001 and 2004, respectively. He was a Permanent Member with the Nippon Telegraph and Telecom cryptography Research Group, Japan, and a Faculty Member with the University of Brasilia, Brazil. He is currently the Endowed Associate Professor of information security and information technology with the School of Engineering and Technology, University of Washington Tacoma, USA. His research interests include cryptography, information security, privacy, and machine learning applications in these areas.

MARTINE DE COCK received the M.S. and Ph.D. degrees in computer science from Ghent University, Belgium, in 1998 and 2002, respectively. She was a Research Assistant and a Postdoctoral Fellow with the Scientific Research - Flanders, a Visiting Scholar with the BISC Group, University of California, Berkeley, USA, a Visiting Scholar with the Knowledge Systems Laboratory, Stanford University, USA, and an Associate Professor with the Department of Applied Mathematics, Computer Science and Statistics, Ghent University. She is currently a Professor with the School of Engineering and Technology, University of Washington Tacoma, USA, and a Guest Professor with Ghent University. She has over 150 peer-reviewed publications in international journals and conferences on artificial intelligence, data mining, machine learning, information retrieval, web intelligence, and logic programming. Her current research interests include privacy-preserving machine learning, cybersecurity, and data analytics to improve the quality of healthcare. She is also a Program Committee Member of numerous international conferences. She co-organized the KDDCup2013. She has served as an Associate Editor for the IEEE TRANSACTIONS ON FUZZY SYSTEMS.

Chapter 5

INLINE DETECTION OF DGA DOMAINS USING SIDE INFORMATION

This chapter focuses on hardening DGA classifiers against adversarial attacks using side information features in addition to the domain name string. We carefully selected the side information features that the attacker cannot manipulate. Additionally, the information is gathered from just the DNS query/response and does not rely on any external sources such as WHOIS queries, thereby enabling inline detection of DGA domain names. The contributions in this chapter are:

- Lexical feature extraction for domain names and side information feature extraction from DNS query and response resource records.
- Train B-RF [29] classifier using a) lexical features only, b) side information features only, and c) a combination of both lexical and side information features.
- Train state-of-the-art LSTM.MI [31] to compare the performance with B-RF trained using side information
- Real traffic analysis to evaluate practical usefulness of the DGA classifiers trained using side information
- Measure robustness against adversarial attacks by assessing the proportion of CharBot [21] domains detected as DGA by the classifier

Inline Detection of DGA Domains Using Side Information

Raaghavi Sivaguru, Jonathan Peck, Femi Olumofin,
Anderson Nascimento, Martine De Cock

***Abstract*—Malware applications typically use a command and control (C&C) server to manage bots to perform malicious activities. Domain Generation Algorithms (DGAs) are popular methods for generating pseudo-random domain names that can be used to establish a communication between an infected bot and the C&C server. In recent years, machine learning based systems have been widely used to detect DGAs. There are several well known state-of-the-art classifiers in the literature that can detect DGA domain names in real-time applications with high predictive performance. However, these DGA classifiers are highly vulnerable to adversarial attacks in which adversaries purposely craft domain names to evade DGA detection classifiers.**

In our work, we focus on hardening DGA classifiers against adversarial attacks. To this end, we train and evaluate state-of-the-art deep learning and random forest (RF) classifier for DGA detection using additional side information that are harder for adversaries to manipulate than the domain name itself. Additionally, the side information features are selected such that they are easily obtainable in practice to perform inline DGA detection. The performance and robustness of these models are assessed by exposing them to one day of real-traffic data and domains generated by adversarial attack algorithm such as CharBot. We found that the DGA classifiers that rely on both domain name and side information have high performance and are more robust against adversaries.

I. Introduction

Domain Generation Algorithms (DGAs) are subroutines that generate pseudo-random combinations of characters or words, and output domain name strings [1]. DGAs often use a seed input such as a number, which is embedded as part of the code, or a time-based element such as the system date, time etc., or a combination of both, to generate random strings. These strings are then concatenated with an available top level domain (TLD) to form domain names. The key idea behind DGAs is to generate the same set of domain names when executed by two different machines, such as by a botmaster and on an infected machine, at a given time. The botmaster registers one of the generated domain names, while the infected machines systematically queries the domains from the generated list until one of them is resolved. The domains from the list that have not been registered by the botmaster will typically result in an NXDomain (non-existent domain) response when queried, and can be discarded by the infected machine. This technique is often used by a command and control (C&C) center and an infected bot to establish communication and perform malicious activities as instructed by the C&C server.

The C&C server then issues commands to the infected bots to perform malicious activities such as distributed denial of service (DDoS) attacks, spamming, stealing sensitive information from the compromised machines etc. In the past, malware authors used a predefined list of domain names, which are embedded in the malware, to communicate with the bots. This technique made it easy for the defenders to blacklist the malicious domain names and block further communication. To overcome this, modern C&Cs use DGAs to randomly generate

domain names that are registered on the go, making them harder to detect. It is therefore important to identify the domains generated by DGAs and block them before they can be used to establish communication between the bot and the C&C center. There are several machine learning approaches proposed in the literature to address this issue including [2], [3], [4], [5], [6], [7], [8], [9] and other work that we cite later in this paper. These well known state-of-the-art classifiers can be deployed in real-world DNS applications to detect DGA domain names and block them. While some work focuses on detecting DGAs from NXDomains[4], our work aims to detect DGAs from resolved traffic.

Commonly used approaches for DGA detection can be categorized according to how fast they are able to flag malicious activity in DNS traffic. As illustrated in Table I, some techniques work in a *retrospective* manner, in which past DNS traffic, which is logged over a certain window, is analyzed in batches to detect anomalies. Other techniques work *inline*, meaning that they can detect DGA domains as soon as they are queried. There are two ways in which inline DGA detection can happen namely: (a) the domain first reaches the DGA classifier and if the classifier flags the domain as benign, then the query is passed to the DNS resolver to fetch the resolved IP address of the domain. However, if the classifier flags the domain as DGA, then the query will not be forwarded to the DNS resolver and it simply blocks the communication with that domain (b) the domain first queries through the DNS resolver and the DGA classifier uses the features learned from the DNS response to make a decision on whether the domain is DGA or not. Our work fits into the second category of inline detection, where both the domain name and the side information features learned from the DNS query/response are used by the classifier for DGA detection. The side information features are carefully selected to allow *inline DGA detection in the broader sense*. In the strict sense, inline DGA detection means that the information required to determine whether a domain name is DGA or not is available from the DNS query data alone. A DNS resolver can use the strictly inline DGA classifier's decision to determine if it is safe or not to resolve the query. A less conventional version of inline detection, which we refer to as "inline DGA detection in the broader sense", is one where data attributes from DNS responses are required (in addition to DNS queries). This means that the DNS resolver must resolve the query first, feed the information obtained to the DGA classifier, and then use the DGA classifier's decision to determine if it is safe to get the DNS response to the client or not. As we observe in our experimental results, taking information from DNS responses into account improves the ability of DGA classifiers to correctly detect DGA domains among resolvable traffic. We note that any dependence on data requiring queries to additional sources, such as the WHOIS database (as used for instance in [10], [11], [12]), would disqualify the approach from inline detection, even in the broader sense.

Machine learning based approaches to detect DGA domain names in practice can also be categorized according to the information they leverage. One way is to train classifiers to detect DGA domain names using only the domain name string itself, see e.g. [4], [5], [6], [8], [13], [14], [15]. The alternative is to train the classifiers using context information such as the IP address of the domain, its geographic location, attributes from DNS response records etc. in addition to the domain name [3], [10], [16], [17], [18], [12]. In our work, we combine both approaches. The advantage of the former approach is that it does not require gathering of additional information, which may be expensive to collect in real time, and that it allows the defenders to detect the DGA domain names and block them even before they can be resolved. The advantage of the latter approach is that side information is a lot harder for the attacker to manipulate than the domain name string itself, making machine learning models trained on side information potentially more robust against adversarial attacks.

Adversarial machine learning is a research area focused on problems introduced by the use of machine learning techniques in adversarial environments in which an intelligent adversary attempts to exploit the weaknesses in such techniques [19]. The *adversarial attacks* of interest in this paper are *evasion attacks* in which an adversary uses artificially crafted instances, called *adversarial samples*, that are intentionally used to mislead a machine learning system and produce erroneous results. The goal of evasion attacks in the context of DGA detection is to generate domains that will be labeled as benign by the DGA classifier. The vulnerability of a classifier against evasion attacks is measured in terms of DGA detection rate, which is the proportion of the adversarial samples predicted as malicious by the classifier. Lower DGA detection rates indicate high vulnerability of the classifier to the attack. There exists several evasion attacks against DGA classifiers such as CharBot[20], DeepDGA[21], DeceptionDGA[22], MaskDGA[23] and the DGAs (HMM & PCFG-based) proposed by [24]. CharBot and MaskDGAs are black-box targeted evasion attacks that do not require any knowledge about the DGA classifier and are intended to generate samples that can evade from being detected as DGA by the classifier. On the other hand DeceptionDGA is a white-box attack algorithm that uses the knowledge of features used by the DGA classifier to generate evading instances. Both types of attacks are found to be extremely powerful in generating domains that can evade detection by the DGA classifiers with high probability.

The main contributions of our work are:

- A comprehensive survey of lexical and side information features proposed in the literature on DGA detection.
- An experimental evaluation of the feasibility in collecting the features and their effectiveness when deployed for inline detection of DGAs in real stream of passive DNS traffic, which leads to a shortlist of features that are actually beneficial in practice.
- Experimental results that show how the side information features can make DGA classifiers more robust against adversarial attacks.

II. Related Work

Given the importance of being able to detect and block DGA domain related traffic, it comes as no surprise that the problem of automatic DGA detection has received a considerable amount of attention over the last decade. There are various ways in which existing DGA detection approaches differ from each other. As illustrated in Table I, DGA detection can be categorized according to the kind of input they require. Some techniques require just the *domain name string*, while other techniques require *side information*, or a combination of both. Both kinds of input have their own advantages and disadvantages. Methods that rely only on the domain name string are popular because side information is typically harder to obtain. On the other hand, features extracted from side information are harder to manipulate, making methods based on them more robust against adversarial attacks. All the approaches presented in our paper performs *inline* DGA detection using domain name only, side information only and a combination of both domain name & side information features.

Furthermore, the classifiers can be trained in two ways to detect if a given domain name is generated by a DGA or not. The first technique is the *featureful approach*, where the classifier relies on human engineered features extracted from the domain names. The second technique is the *featureless approach*, where the classifier learns the features automatically during the training process. Classifiers that are based on deep learning architectures like Long Short-Term Memory (LSTM) [5], [8] and Convolutional Neural Network (CNN) models [6], [9] leverage the featureless approach, whereas models such as random forests (RFs) adopt the featureful approach. In our work, we will be using both featureful and featureless approaches to

train random forest and deep learning classifiers for DGA detection.

III. side information Features

```
[{'name':'junebugweddings.com.,',
  'ttl': 300,
  'type': 1,
  'class': 1,
  'data': '104.27.191.148'},
 {'name': 'junebugweddings.com.,',
  'ttl': 300,
  'type': 1,
  'class': 1,
  'data': '104.27.190.148'}]
```

Fig. 1. An example DNS resource record

In this section we provide a detailed overview of side information features that can be extracted from DNS traffic to aid in the detection of DGA domains. An overview of all features is presented in Table II, accompanied with a list of citations that illustrates the popularity of each kind of feature in the literature. The order of the side information features listed in Table II indicates the importance of those features in DGA detection as ranked by the Random Forest model (see Section V). Not all features are equally easy to obtain in practice, and their contribution to the predictive accuracy of DGA classifiers varies. The last column of Table II indicates whether we retained the feature in our DGA-classifiers. Figure 1 shows a sample resource record data from which the side information features are extracted. In Figure 1, the attribute "name" represents the fully qualified domain name (FQDN), "ttl" represents the time-to-live of the DNS query, "type" represents the resource record type, "class" represents the class of resource record and "data" represents the resolved IP address. Below we give a more in-depth description of each kind of feature, and its typical use in the the literature on DGA detection. Figure 2 shows a comparison of density plots for some of the side information features extracted from benign and DGA domain names. The different side information features are listed below:

- **rrlength:** This feature measures the length of the RData field, which is extracted directly from the DNS response resource record. The RData in a DNS response encompasses a list of resolved IP addresses, the time-to-live value of the query and the type of resource record.
- **country:** This feature refers to the geographic location that the resolved IP address maps to. If the DNS resource record contains multiple IP addresses, the country for each of the IP addresses is first identified. If all of the IP addresses belong to the same country, then this feature takes up that name. On the other hand, if any of the IP addresses map to a different location, then the value of this feature would be "multi-valued". Alternatively, if the location could not be identified, then this feature takes the value "unknown". This feature is then converted to categorical values that range between 0 and 185, which means that the domains in our data set map to 184 different countries plus the values "multi-valued" and "unknown".
- **ttl:** This feature represents the time-to-live value of the DNS query, which is the time interval that the resource record can be cached by the DNS resolver, and is directly obtained from the DNS response resource record. Table III compares the distribution of TTL values (in seconds), in terms of mean, standard deviation and median, for benign and DGA domains in our data set (see Section VI-A). It can be seen that DGA domains are in general far more short-lived than benign domains. For better visibility in Figure 2, the density plot for TTL values are shown in hours instead of seconds.
- **n_ip:** This feature indicates the number of distinct IP addresses that are returned for the DNS domain lookup. It is manip-

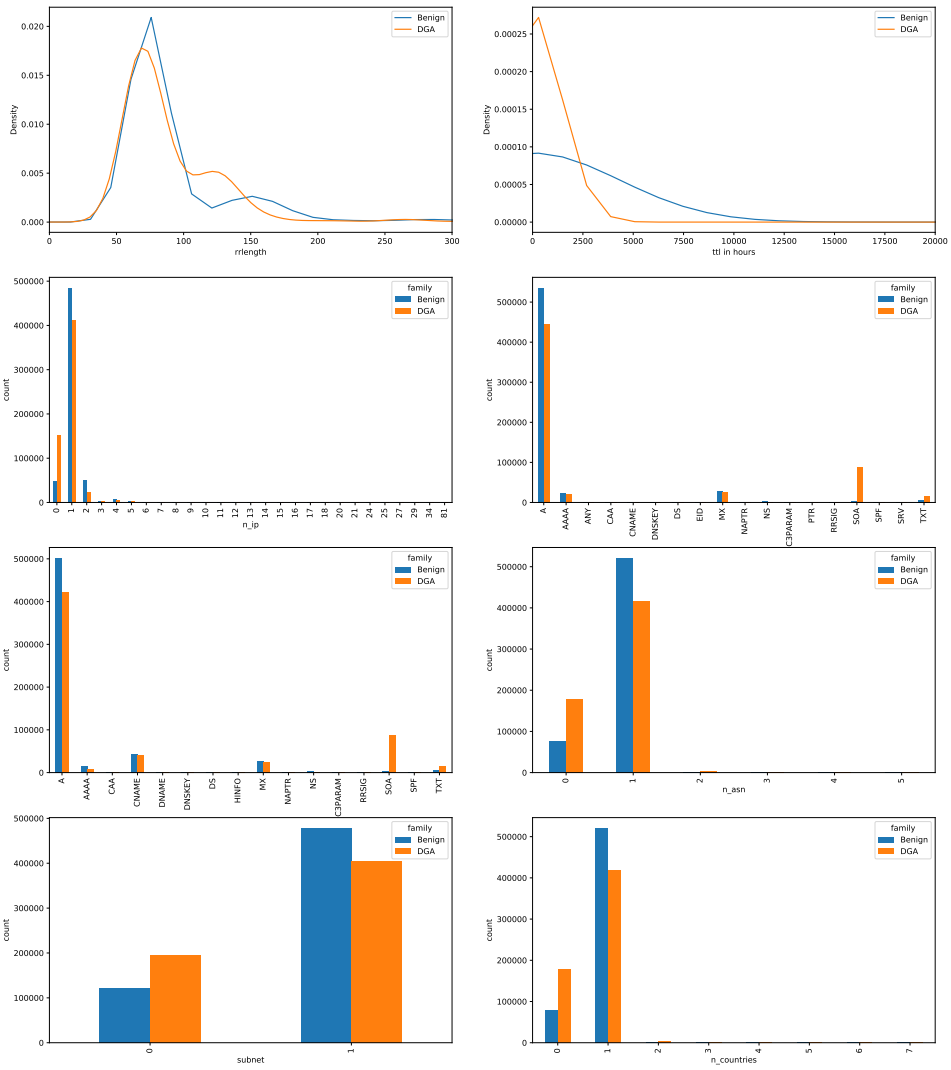


Fig. 2. Comparison of values for side information features extracted from benign and DGA domains

Input	Retrospective	Inline
Domain name string	[2], [25]	[8], [5], [4], [15], [26] [27], [6], [14], [13], [28]
Side information features	[29], [30], [16], [31], [32]	our work
Domain name string + side information features	[18], [33], [3], [34], [35] [17], [36], [10], [11], [12]	our work

TABLE I
OVERVIEW OF EXISTING WORK ON DGA DETECTION

Feature	Description	Reference	Retained
rrlength	Resource record length	[31]	✓
country	Country name that the domain maps to	[11], [12], [18], [33]	✓
ttl	Time-to-live of the DNS query	[33], [31]	✓
n_ip	Number of distinct IP addresses the domain maps to	[36], [17], [11], [12], [18]	✓
qtype	Type of DNS packet requested	[31]	✓
rtype	Record type of the DNS response	[31]	✓
n_asn	Number of distinct ASNs the domain maps to	[32]	✓
subnet	Do all IPs belong to same subnet	[12], [11]	✓
n_countries	Number of distinct countries the domain maps to	[36], [17], [11], [12], [18], [34]	✓
timestamp	Features derived from timestamp of the DNS query	[29], [36], [17], [18]	✗
opcode	Kind of DNS query	[31]	✗
AA	Authoritative answer	[31]	✗
QDCOUNT	Number of entries in question section	[31]	✗
ANCOUNT	Number of resource records in answer section	[31]	✗
NSCOUNT	Number of name servers in authoritative section	[31]	✗
ARCOUNT	Number of resource records in additional record section	[31]	✗
RCODE	Response code	[31]	✗
rDNS	Reverse DNS query results	[36], [17], [11], [12]	✗
TTL statistics	Mean, standard deviation etc. of time-to-live	[36], [17], [11], [12], [18]	✗
n_domains	Number of distinct domains associated with the IP	[36], [17], [11], [12], [18]	✗
n_queries	Number of queries for the domain and (domain, IP) pair	[18]	✗
WHOIS features	Registrar, domain creation/expiration date etc.	[10], [11], [12], [33]	✗

TABLE II
SIDE INFORMATION FEATURES

Type	Mean TTL	SD TTL	Median TTL
Benign	109,447	1,421,829	3,600
DGA	29,255	4,701,205	900

TABLE III
TTL DISTRIBUTION (IN SECONDS) FOR BENIGN AND DGA DOMAINS

ulated directly by accessing the list of IPs contained in the RData field of the DNS response resource record.

- **qtype:** This feature represents the DNS query type that can be extracted from the question section of the DNS query. Figure 2 shows the different values for this features in our data set.
- **rtype:** This feature represents the resource record type that can be extracted directly from the RData field in the DNS response resource record. Figure 2 shows the different values for this features in our data set.
- **n_asn:** This feature indicates the number of distinct autonomous system numbers that the IP addresses map to. The ASN for a given IP address is obtained by using Python Geolite2 Maxmind API.[1]
- **subnet**: This feature is a boolean value that represents if all the IP addresses belong to the same subnet. A value of 0 indicates that one or more of the IP addresses, returned in the DNS response, belong to a different

[1]https://geoip2.readthedocs.io/en/latest/

subnet and value of 1 indicates that all the IP addresses map to the same subnet.

- **n_countries**: This feature represents the distinct number of countries that the resolved IP addresses map to. This feature has a very similar distribution when compared to the "n_asn" feature, which can be observed in Figure 2.
- **timestamp:** The timestamp denotes the time at which the DNS query was issued by a host. This feature in itself may not be useful in detecting DGAs. Some of the past studies record all of the timestamps at which a particular domain name was queried and construct time-series data to analyze the periodicity at which the benign and DGA domains are queried [36], [17], [29], whereas [18] computes the lifespan of a domain by subtracting the first and last seen timestamps of the domain name. Such approaches require access to past DNS traffic and hence are regarded as "retrospective". Since we only focus on performing inline DGA detection in our work, we do not use the timestamp feature to perform DGA classification.
- **opcode:** This feature represents the kind of query such as standard query, inverse query, request for server status etc. In our data set, all the domains being queried belong to *standard query* type and hence using this feature does not contribute in the prediction of DGA domain names.
- **aa:** This feature is a boolean flag which represents if the responding name server is an authority for the domain name being queried. The AA flag for all DNS responses in our data set has the same value "True" and hence we do not leverage the AA flag information while training our DGA classifiers.
- **qdcount, ancount, nscount, arcount:** At this time, our DNS traffic collector do not capture this information & hence we do not use these features to train our model. However, it can be easily obtained from the DNS query and resource records.
- **rcode:** Since our data set comprises of resolved domain names only, the rcode remains "0" for all the samples and hence we discard this information.
- **TTL statistics:** This refers to a collection of features such as standard deviation, mean, minimum, maximum etc. of all time-to-live values extracted from the DNS response. While these features are relevant in a retrospective approach that investigates a domain based on *all* DNS resource records related to it say during the past 24 hours, it is not meaningful for fast inline DGA detection. Indeed, since all of the TTL values in a single response record have constant values, it would not add value to include these statistics as features.
- **n_domains:** This feature represents the number of distinct domain names that are mapped to a given IP address. In order to use this feature, one needs to maintain a bipartite graph that depicts the mapping for each (domain, IP) pair. Again, this method of performing graph inference is computationally intensive and does not contribute towards inline detection of DGA domains. Therefore we refrain from using this side information feature while training our DGA classifiers.
- **n_queries:** Similar to "timestamp" and "n_domains", this feature also requires storing and fetching of information from past DNS traffic and hence n_queries cannot be used for inline detection of DGAs.
- **WHOIS features:** Extracting WHOIS features such as registrar, domain creation/expiration date etc. involves very expensive WHOIS queries. This affects the capability of the classifier to perform inline DGA detection on-the-go and hence we do not use any feature that require WHOIS queries.

IV. Lexical Features

In this section we list the 26 human engineered lexical features that are extracted manually from the domain name string in order to train the RF classifier for DGA detection. Table IV shows a list of the lexical features used in B-RF and details on how the feature values are calculated are given below:

- **domain_len:** This feature represents the length of the domain name, which is the number of characters in the SLD.TLD pair. For example, we refer "google.com" as the domain name, where "google" indicates the SLD (second level domain) and "com" indicates the TLD (top level domain). The value of the feature domain_len for the domain name "google.com" is 10.
- **sld_len:** This feature represents the number of characters in the second level domain.
- **tld_len:** This feature represents the number of characters in the top level domain.
- **uni_domain:** This feature represents the number of unique characters in the domain name, after removing special characters such as '.' & '-' from the domain name.
- **uni_sld:** This feature represents the number of unique characters in the second level domain, after removing special characters such as '.' & '-' from the SLD.
- **uni_tld:** This feature represents the number of unique characters in the top level domain, after removing special characters such as '.' & '-' from the TLD.
- **flag_dga:** This feature represents a boolean value (0 or 1) that indicates if the domain name contains any of the following TLDs, which are known to be frequently associated with malicious activities[2]: "study", "party", "click", "top", "gdn", "gq", "asia", "cricket", "biz", "cf".
- **tld_hash:** This feature represents the hash value of top level domain.
- **flag_dig:** This feature represents a boolean value that indicates if the domain name starts with a digit/number (0-9).
- **sym:** This feature represents the ratio of number of special characters in the SLD to the total number of characters in SLD (sld_len).
- **hex:** This feature represents the ratio of number of hexadecimal characters (0-9 & a-f) in the SLD to the total number of characters in the SLD.
- **dig:** This feature represents the ratio of number of digits (0-9) in the SLD to the total number of characters in the SLD.
- **vow:** This feature represents the ratio of number of vowels ('a', 'e', 'i', 'o', 'u') in the SLD to the total number of characters in the SLD.
- **con:** This feature represents the ratio of number of consonants in the SLD to the total number of characters in the SLD.
- **rep_char_ratio:** This feature represents the ratio of number of characters that occurs more than once in the SLD to the total number of unique characters in the SLD.
- **cons_con_ratio:** This feature represents the ratio of consecutive consonants (such as "ct", "fk", "ns" etc.) to the length of the domain (domain_len).
- **cons_dig_ratio:** This feature represents the ratio of consecutive digits (such as "92", "24", "75" etc.) to the length of the domain (domain_len).
- **tokens_sld:** This feature represents the number of tokens in the SLD, where a token indicates sequence of characters separated by '-'.
- **digits_sld:** This feature represents the total number of digits in the SLD.
- **ent:** This feature represents the normalized entropy value of the characters in SLD and is calculated using the formula:

$$ent = \frac{\sum_{i=1}^{n} p_i * log_2(p_i)}{log_2(sld\_len)}$$

where 'n' represents the number of unique characters in the SLD and p_i represents the proportion between the frequency of the

[2]https://www.spamhaus.org/statistics/tlds/

Feature	Description	Reference	Retained
domain_len	Domain name length	[20], [14], [6], [13], [4], [26], [28], [11], [12]	✓
sld_len	Second level domain length	[20], [14], [13]	✓
tld_len	Top level domain length	[20], [14], [13]	✓
uni_domain	Domain Unique Characters length	[20], [14], [13]	✓
uni_sld	SLD Unique Characters length	[20], [14], [13]	✓
uni_tld	TLD Unique Characters length	[20], [14], [13]	✓
flag_dga	Has malicious TLD	[20], [14], [13], [26]	✓
tld_hash	TLD Hash	[20], [14], [13], [6]	✓
flag_dig	Starts with Digit	[20], [14], [13], [6]	✓
sym	Symbol ratio	[20], [14], [13], [6]	✓
hex	Hex ratio	[20], [14], [13], [6]	✓
dig	Digit Ratio	[20], [14], [13], [4], [28], [36], [17], [11], [12]	✓
vow	Vowel Ratio	[20], [14], [13], [6], [4], [28]	✓
con	Consonant Ratio	[20], [14], [13]	✓
rep_char_ratio	Ratio of Repeated Characters	[20], [14], [4]	✓
cons_con_ratio	Ratio of Consecutive Consonants	[20], [14], [4], [28]	✓
cons_dig_ratio	Ratio of Consecutive Digits	[20], [14], [4]	✓
tokens_sld	Number of tokens in SLD	[20], [14], [13], [26]	✓
digits_sld	Number of digits in SLD	[20], [14], [13], [26]	✓
ent	Entropy of characters in SLD	[20], [14], [13], [6], [4], [28]	✓
gni	Gini Index of characters in SLD	[20], [14], [13], [6]	✓
cer	Classification error of characters in SLD	[20], [14], [13], [6]	✓
2gram_med	2-Gram Median of characters in SLD	[20], [14], [13], [6]	✓
3gram_med	3-Gram Median of characters in SLD	[20], [14], [13], [6]	✓
2gram_cmed	2-Gram Circle Median of characters in SLD	[20], [14], [13]	✓
3gram_cmed	3-Gram Circle Median of characters in SLD	[20], [14], [13]	✓

TABLE IV
LEXICAL FEATURES USED BY B-RF

unique character 'c_i' in the SLD to the total number of unique characters in the SLD.

- **gni:** This feature represents the gini value of the characters in SLD and is calculated using the formula:

$$gni = 1 - \sum_{i=1}^{n} p_i^2$$

where 'n' represents the number of unique characters in the SLD and p_i represents the proportion between the frequency of the unique character 'c_i' in the SLD to the total number of unique characters in the SLD.

- **cer:** This feature represents the classification of error of characters in SLD, which is computed using the formula:

$$cer = 1 - \max_{i=1}^{n}(p_i)$$

where p_i represents the proportion between the frequency of the unique character 'c_i' in the SLD to the total number of unique characters in the SLD.

- **2gram_med:** This feature represents the median of 2-gram frequencies in SLD.
- **3gram_med:** This feature represents the median of 3-gram frequencies in SLD.
- **2gram_cmed:** In order to compute this feature, the SLD of the domain is concatenated again with the SLD. (i.e) For example, if "google" is the SLD, a string such as "googlegoogle" is formed. The 2gram_med is then calculated on this newly formed string "googlegoogle" to obtain the value of this feature.
- **3gram_cmed:** In order to compute this feature, the SLD of the domain is concatenated again with the SLD. (i.e) For example, if "yahoo" is the SLD, a string such as "yahooyahoo" is formed. The 3gram_med is then calculated on this newly formed string "yahooyahoo" to obtain the value of this feature.

V. DGA Classifiers

We consider three different DGA classifiers in this work, which we detail below. We chose one model representative of the featureful approach (B-RF), one deep learning model which represents the featureless approach (LSTM.MI) and finally a hybrid model which combines both approaches (LSTM.MI+B-RF).

A. B-RF

B-RF is a random forest based DGA detection classifier that categorizes domain names as benign domains or DGA domains. It consists of 100 trees and each tree is trained using a subset of the feature space to avoid overfitting. Entropy is used as the criterion to decide the split attribute while growing the trees in the random forest. There are 3 variants of B-RF classifier, each trained either on lexical features (as the RF classifier in [14]) or DNS features, or a combination of both lexical and DNS features. The performance of these variants of the B-RF classifier are listed in the first 3 rows of Table VI.

B. LSTM.MI

Woodbridge et al. [5] were the first to propose deep learning for DGA domain name detection. Their DGA classifier is a neural network consisting of an embedding layer, an LSTM layer, and a single node output layer with sigmoid activation. In this paper, we use the LSTM.MI model that was proposed recently by Tran et al. [8]. Its architecture is very similar to that of Woodbridge et al. [5]; the main distinction is that the LSTM.MI model is trained with a cost-sensitive learning algorithm that takes class imbalances into account. This allows the LSTM.MI approach to achieve slightly better results than the original LSTM approach (see [8], [14]). The 4th row in Table VI shows the performance of the LSTM.MI classifier. It operates directly on the domain name string, instead of on lexical features extracted from it. Characters in the domain name are converted to lower case and are encoded with categorical values, ranging from 1 to 38, to represent '.', '-', digits from 0 to 9 & characters from 'a' to 'z'. All the domains in our data are fixed to a length of 77 characters, which is the length of the longest domain name in our data set. Hence, the domains that are shorter than 77 characters are padded with zeroes in the left.

C. LSTM.MI+B-RF

The hybrid LSTM.MI+B-RF classifier combines both LSTM.MI and B-RF architectures by training a B-RF classifier with features listed in Table II and IV, in addition to the confidence score obtained from the LSTM.MI model for that domain name. The confidence score ranges between 0 and 1, signifying the probability of the domain being a DGA as predicted by the LSTM.MI classifier. The above workflow of DGA detection using LSTM.MI+B-RF setup is depicted in Figure 3. The last two rows in Table VI represent the performance of this DGA classifier.

VI. Experimental results

A. Dataset

In the first experiment, we train and evaluate the DGA classifiers from Section V on a dataset with 600,000 DGAs (positive) and 600,000 benign (negative) samples. Table V shows some examples of DGA & benign domains. The training data points originate from a real-time stream of passive DNS data, consisting of roughly 10-12 billion DNS queries per day collected from subscribers including ISPs (Internet Service Providers), schools, and businesses. From this traffic, the positive samples are collected by retaining resolved domain names that are listed in DGArchive[3], a blacklist with DGA domains [1]. Dictionary DGAs, which are human-readable DGA domains belonging to malware families such as suppobox, gozi, matsnu and nymaim2 are discarded from

[3]https://dgarchive.caad.fkie.fraunhofer.de/

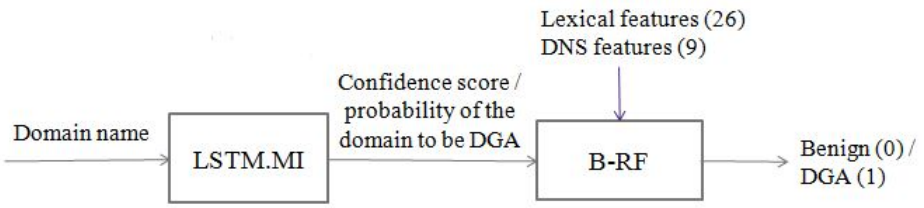


Fig. 3. DGA detection using LSTM.MI + RF model

the training set. This is because these DGAs look more like benign domains and confuse the DGA classifiers[25]. Since this work is primarily aimed at measuring the impact of adversarial instances such as CharBot, we exclude samples from Dictionary DGAs. The benign samples are collected based on a predefined set of heuristics as listed below:

- Domain name should have valid DNS characters only (digits, letters, dot and hyphen)
- Domain has to be resolved at least once for every day between June 01, 2019 and July 31, 2019.
- Domain name should have a valid public suffix
- Characters in the domain name are not all digits (after removing '.' and '-')
- Domain should have at most four labels (Labels are sequence of characters separated by a dot)
- Length of the domain name is at most 255 characters
- Longest label is between 7 and 64 characters
- Longest label is more than twice the length of the TLD
- Longest label is more than 70% of the combined length of all labels
- Excludes IDN (International Distribution Network) domains (such as domains starting with `xn--`)
- Domain must not exist in DGArchive

Both the DGA and benign domains in the data set are collected from real-time passive DNS traffic that was observed in February 2019. The domains in the data set are then preprocessed by following the two steps mentioned below:

- Retain only the SLD & TLD of the domain name and discard any 3LD (third level domain) or any other label if present. For example, for the domain name "www.google.com", the 3LD which is "www" is removed and the SLD.TLD which "google.com" is retained.
- All the alphabetical characters in the domain name are converted to its corresponding lower case characters.

B. Performance evaluation of DGA classifiers

The true positive rate (TPR) and false positive rate (FPR) for the DGA classifiers are calculated as follows:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

where TP, TN, FP & FN represent the number of true positives, true negatives, false positives and false negatives respectively. The predictive performance of the classifiers is evaluated using 5-fold cross-validation for metrics such as TPR and Area Under the ROC Curve (AUC) as tabulated in Table VI. In cybersecurity applications, it is important to achieve high TPR for a very low FPR. This is because it is undesirable to block a large number of benign domains in real-world traffic. Hence all the reported metrics are

Benign domains (labeled 0)	DGA domains (labeled 1)
7ft4.com	vocom.eu
sgtobel.ch	leadhelp.net
intimvoronezh.net	1b6a95e6b5d4.com
essc-tabriz.com	korpncyeajsgeatkopoqs.info
konsaltbezopasnost.ru	kndydusmrlrofrcmfuayfmswrkytl.biz

TABLE V
SOME EXAMPLES FO BENIGN VS DGA DOMAIN NAMES

thresholded at a very low FPR of 0.1%. We also obtain the ROC (Receiver Operating Characteristic) curve by plotting the TPR against the FPR of the classifiers. The AUC is obtained by taking the integral of the ROC curve. It is a measure of how well the trained classifier can distinguish between the classes. Higher values of AUC indicate better models. An ideal classifier is the one which has an AUC score of 1; a classifier with an AUC score of 0.5 indicates that it is randomly guessing; and a classifier with AUC 0 means that the predictions are inverted, i.e. samples labeled as 0 are predicted as 1 by the classifier and vice versa. In addition to the AUC score, the AUC at a fixed FPR of 0.1% is also reported. This thresholded AUC represents the integral of the ROC curve for a FPR of 0 to 0.001.

There are several interesting observations to be made based on Table VI. First, looking at the AUC@1%FPR column, one can see that the predictive performance for inline DGA detection based on DNS features alone is not good: the B-RF/DNS based model achieves an AUC@1%FPR of only 53.23%. Second, when it comes to DGA detection based on the domain string alone, the deep learning approach (LSTM.MI) clearly outperforms the random forest approach (B-RF/Lexical) at 94.47% vs. 89.78%. This is fully in line with previous findings [8], [27]. Third, the most interesting and novel result from Table VI is that the DGA classifiers, when trained with *both* lexical and side information features, have the best overall performance in terms of AUC score and TPR, namely 99.17% for the architecture from Fig. 3.

C. Real Traffic Analysis

Next, we apply the best performing classifiers in Table VI on one day of real traffic DNS traffic to evaluate their predictive performance in real-time. We collected a set of resolved domains that were observed on March 26, 2019 to perform this analysis. As part of pre-processing, the fully qualified domain names are validated against the heuristics mentioned in Section VI-A, in order to maintain consistency with the training data set. The domains that satisfy the heuristics are then retained in this experiment after discarding the third level domain (3LD/subdomain) from the domain name, if present. This resulted in a set consisting of 66,440,681 domains (contains duplicate domains with SLD.TLD pairs), out of which 1,159,662 domains were found in DGArchive and 14,653,217 domains were found in Alexa. There is also an overlap of 1,124,467 domains between the Alexa whitelist and DGArchive blacklist.

Table VII shows a comparison of the number of domains that were flagged as DGA by the LSTM.MI, B-RF and LSTM.MI+B-RF classifier. The B-RF model (in Table VII) has the highest true positive rate among the 3 models being compared (i.e.) Out of the 1.87M domains flagged as DGA by the classifier, approximately 61% were found in DGArchive. Although the LSTM.MI classifier catches the highest number of DGAs in real-traffic, the true positive rate is 34% which is 27% lower than the B-RF classifier. However, as seen in the last row of Table VII, the B-RF also has the highest number of false positives. This could largely be due to the fact that there is a large number of overlapping domains between Alexa and DGArchive as men-

Model	Features	Performance metrics	
		AUC@ 0.1%FPR	TPR@ 0.1%FPR
B-RF	DNS	53.23%	16.21%
	Lexical	89.78%	97.44%
	DNS + Lexical	**98.19%**	**99.42%**
LSTM.MI	Domain name string	94.47%	98.80%
LSTM.MI + B-RF	Domain name string + DNS	96.51%	99.89%
	Domain name string + DNS + Lexical	**99.17%**	**99.91%**

TABLE VI
PERFORMANCE EVALUATION OF DGA CLASSIFIERS USING 5-FOLD CROSS-VALIDATION

tioned earlier in this section. A good workaround to reduce the number of false positives during the deployment is to check the flagged domains against Alexa before making the final decision.

D. Defense against Adversarial ML

The use of side information is important in the context of adversarial ML because side information is a lot harder to manipulate than the domain name string itself [10]. In order to test this, we generated 1,000 DGA domains with CharBot[20], a simple DGA algorithm that was written specifically to evade existing DGA classifiers. Since, to the best of our knowledge, CharBot has not been deployed yet in the wild, we can not collect side information for CharBot domains from real traffic. Instead, we pair up the CharBot domains with the DNS features obtained from 1,000 randomly sampled DGA domains in real traffic. To avoid any bias in the selection of DNS features for CharBot domains, we perform the random sampling for 5 trials and create 5 sets of CharBot DNS features. The lexical features extracted for CharBot are appended with the DNS features, which can then be exposed to DGA classifiers for detection of malicious domains. The idea here is to test if the DGA classifiers trained on side information features are successful in detecting CharBot domains.

Table VIII shows the CharBot detection rate, which is the average proportion of CharBot domains that were flagged as DGA by the classifiers over the 5 randomized trials. Higher values of CharBot detection rate indicates that the classifier is more robust against new DGAs or adversarial attacks. As expected, the B-RF model trained on both lexical and side information features detects 20% of CharBot domains as DGA/malicious, which is 12% more than the LSTM.MI model. This clearly indicates that the use of side information features to train the DGA classifier makes it more robust against adversarial samples like CharBot domains, when compared to classifiers that rely only on the domain name for DGA detection.

VII. CONCLUSION

In this paper, we proposed and evaluated state-of-the-art classifiers for inline DGA detection using side information features that are easily obtained from DNS query and response. Results from Table VI & VIII shows that using side information in addition to the domain name to train classifiers not only improves the predictive performance, but also makes it more robust against adversaries like CharBot, when compared to the classifiers that use just the domain name to detect DGAs. Additionally, the side information features in our approach are carefully chosen to perform lightweight *inline* detection of DGA domains, and do not rely on external sources such as WHOIS for feature extraction.

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

Model	LSTM.MI	B-RF	LSTM.MI+B-RF
Features	Domain name	DNS + Lexical	Domain name + DNS + Lexical
Out of the 66M domains in real-traffic, number of domains flagged as DGA by the classifier	3,400,017	1,877,784	2,170,056
Out of the domains flagged as DGA by the classifier, number of domains found in DGArchive	1,151,750	1,149,689	1,150,116
Out of the domains flagged as DGA by the classifier, number of domains found in Alexa	1,626,232	1,717,638	1,420,319

TABLE VII
REAL TRAFFIC ANALYSIS OF DGA CLASSIFIERS ON 66,440,662 (66M) DOMAINS

Classifier	Features	DGA (CharBot) detection rate
B-RF	DNS	1.70 % ± 0.24 %
	Lexical	3.80 % ± 0.0 %
	Lexical + DNS	20.06 % ± 0.56 %
LSTM.MI	Domain name string	8.00 % ± 0.0 %
LSTM.MI+B-RF	Domain name string + DNS	14.98 % ± 0.63 %
	Domain name string + Lexical + DNS	15.76 % ± 0.53 %

TABLE VIII
DETECTION RATE OF CHARBOT DOMAINS AS DGA

REFERENCES

[1] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A Comprehensive Measurement Study of Domain Generating Malware," in *25th USENIX Security Symposium*, 2016, pp. 263–278.

[2] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, 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, pp. 491–506.

[3] 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*. Springer, 2014, pp. 192–211.

[4] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "FANCI: Feature-based Automated NX-Domain Classification and Intelligence," in *27th USENIX Security Symposium*, 2018, pp. 1165–1181.

[5] 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.

[6] B. Yu, D. L. Gray, J. Pan, M. De Cock, and A. C. Nascimento, "Inline DGA Detection with Deep Networks," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 683–692.

[7] J. Koh and B. Rhodes, "Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings," in *Proceedings of 2018 IEEE International Conference on Big Data*, 2018, pp. 2965–2970.

[8] 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.

[9] J. Saxe and K. Berlin, "eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys," *preprint arXiv:1702.08568*, 2017.

[10] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, "Detecting DGA Domains with Recurrent Neural Networks and Side Information," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*. ACM, 2019.

[11] T. Chin, K. Xiong, C. Hu, and Y. Li, "A Machine Learning Framework for Studying Domain Generation Algorithm DGA-based Malware," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2018, pp. 433–448.

[12] Y. Li, K. Xiong, T. Chin, and C. Hu, "A Machine Learning Framework for Domain Generation Algorithm DGA-Based Malware Detection," *IEEE Access*, vol. 7, pp. 32 765–32 782, 2019.

[13] C. Choudhary, R. Sivaguru, M. Pereira, B. Yu, A. C. Nascimento, and M. De Cock, "Algorithmically Generated Domain Detection and Malware Family classification," in *International Symposium on Security in Computing and Communication*. Springer, 2018, pp. 640–655.

[14] R. Sivaguru, C. Choudhary, B. Yu, V. Tymchenko, A. Nascimento, and M. De Cock, "An Evaluation of DGA Classifiers," in *2018 IEEE International Conference on Big Data*, 2018, pp. 5058–5067.

[15] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character Level Based Detection of DGA Domain Names," in *Proc. WCCI*, 2018, pp. 4168–4175.

[16] 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.

[17] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains," *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 4, 2014.

[18] P. Lison and V. Mavroeidis, "Neural Reputation Models learned from Passive DNS Data," in *2017 IEEE International Conference on Big Data*, 2017, pp. 3662–3671.

[19] Y. Vorobeychik and M. Kantarcioglu, "Adversarial Machine Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–169, 2018.

[20] J. Peck, C. Nie, R. Sivaguru, C. Grumer, F. Olumofin, B. Yu, A. Nascimento, and M. De Cock, "CharBot: A Simple and Effective Method for Evading DGA Classifiers," *IEEE Access*, vol. 7, pp. 91 759–91 771, 2019.

[21] 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.

[22] J. Spooren, D. Preuveneers, L. Desmet, P. Janssen, and W. Joosen, "Detection of Algorithmically Generated Domain Names used by Botnets: A Dual Arms Race." in *Proceedings of the 34th ACM/SIGAPP Symposium On Applied Computing*. Association for Computing Machinery, 2019, pp. 1902–1910.

[23] L. Sidi, A. Nadler, and A. Shabtai, "MaskDGA: A Black-box Evasion Technique Against DGA Classifiers and Adversarial Defenses," *arXiv preprint arXiv:1902.08909*, 2019.

[24] Y. Fu, L. Yu, O. Hambolu, I. Ozcelik, B. Husain, J. Sun, K. Sapra, D. Du, C. T. Beasley, and R. R. Brooks, "Stealthy Domain Generation Algorithms," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1430–1443, 2017.

[25] 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 *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 295–314.

[26] A. Joshi, L. Lloyd, P. Westin, and S. Seethapathy, "Using Lexical Features for Malicious URL Detection–A Machine Learning Approach," *arXiv preprint arXiv:1910.06277*, 2019.

[27] B. Yu, J. Pan, D. Gray, J. Hu, C. Choudhary, A. C. Nascimento, and M. De Cock, "Weakly Supervised Deep Learning for the Detection of Domain Generation Algorithms," *IEEE Access*, vol. 7, pp. 51 542–51 556, 2019.

[28] Z. Wang, Z. Jia, and B. Zhang, "A Detection Scheme for DGA Domain Names Based on SVM," in *2018 International Conference on Mathematics, Modelling, Simulation and Algorithms (MMSA 2018)*. Atlantis Press, 2018.

[29] 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.

[30] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, "Detecting Malware Domains at the Upper DNS Hierarchy." in *USENIX Security Symposium*, vol. 11, 2011, pp. 1–16.

[31] L. Watkins, S. Beck, J. Zook, A. Buczak, J. Chavis, W. H. Robinson, J. A. Morales, and S. Mishra, "Using Semi-supervised Machine Learning to Address the Big Data Problem in DNS Networks," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, 2017.

[32] I. Khalil, T. Yu, and B. Guan, "Discovering Malicious Domains through Passive DNS Data Graph Analysis," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 663–674.

[33] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 1245–1254.

[34] M. Singh, M. Singh, and S. Kaur, "Detecting bot-infected machines using DNS fingerprinting," *Digital Investigation*, vol. 28, pp. 14–33, 2019.

[35] J. Abbink and C. Doerr, "Popularity-based Detection of Domain Generation Algorithms," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, no. 79. ACM, 2017.

[36] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis," in *Ndss*, 2011, pp. 1–17.

Chapter 6

CONCLUSION

In our thesis, we trained and compared the performance of state-of-the-art DGA classifiers that use LSTM and Random Forest algorithms for DGA detection and DGA malware family classification tasks. We also evaluated the performance of the classifiers on time-dependent DGAs vs. time-invariant DGAs and found that the classifiers were more robust to change in seed than the change in timestamp used by the DGA generator. We also measured the quality of our training data sets by building models using samples from publicly available data sources and the ones that are collected from a real stream of passive DNS traffic. We observed that the domains available in public data sources like Alexa are not a good representation of the benign domains observed in real traffic data. This finding is particularly useful as many studies in the literature train their DGA classifiers using samples from Alexa to represent benign domains. However, this does not result in good performance of the DGA classifiers when exposed to real-time DNS traffic data.

Although there exist powerful DGA classifiers with high predictive performance, studies show that they are still vulnerable to adversarial attacks. Adversarial attacks are often devised by cybercriminals who wish to degrade the performance of the ML systems or to evade from being detected to perform malicious activities. In order to assess the robustness of our DGA classifiers against such attacks, we exposed samples generated by algorithms such as Charbot [21], DeceptionDGA [30] and DeepDGA [2] and found that most of the adversarial samples are successful in evading state-of-the-art DGA classifiers that rely only on the domain name to detect DGAs.

Several research findings suggest the use of adversarial training as an effective strategy to make the classifiers robust against adversarial attacks[2, 27]. However, this technique does

not generalize well for new attack algorithms. This led to the idea of using side information features that may be a lot harder for the adversary to manipulate, in addition to the domain name, for training DGA detection classifiers. Additionally, we also proposed to perform *inline* DGA detection using side information features that can be easily obtained from the DNS query and response. "Inline" detection means that we do not gather information from any additional resources such as WHOIS nor perform analysis on past DNS traffic to derive values for the features. Instead, we extract features on a per-domain basis as it is seen in DNS traffic to perform DGA detection. We found that the classifiers that rely on features derived from side information apart from the domain name are more robust to adversarial attacks such as CharBot and have high predictive performance when deployed in real-time DNS applications.

Appendix A

A.1 Code, data and model for Chapter 2

File	**Location on UWT MCS1 machine**
Code	/home/raaghavi/thesis/chapter2/code
Data	/home/raaghavi/thesis/chapter2/data
Models	/home/raaghavi/thesis/chapter2/models
Feature Extraction	Infoblox Github repository

Table A.1: Location of files used in Chapter 2

A.2 Slides for Chapter 2

Algorithmically Generated Domain Detection and Malware Family Classification

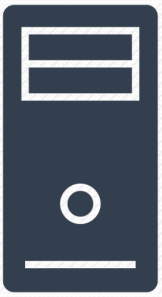
Chhaya Choudhary, Raaghavi Sivaguru, Mayana Pereira,
Bin Yu, Anderson C. Nascimento, Martine De Cock

Infoblox
CONTROL YOUR NETWORK

Domain Generation Algorithm (DGA)

Infected Machine

0-0v.net →
← NXDomain
0-2-60.org →
← NXDomain
xjqvqfjwxhue.com →
← Success (IP address)
0-experimental.com →
← NXDomain

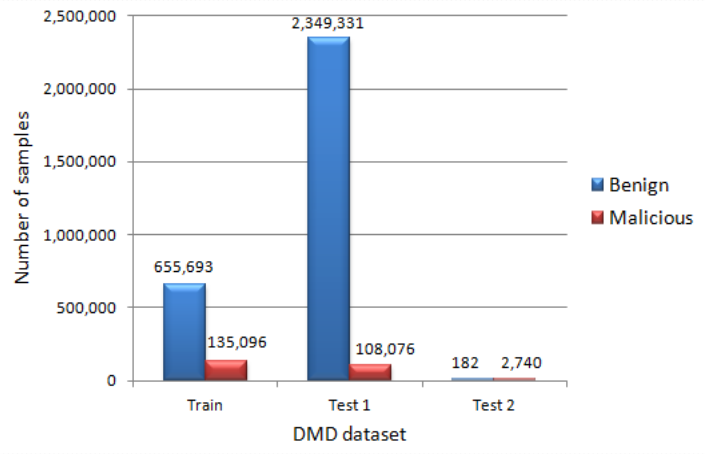


DNS Server

Success (IP address)

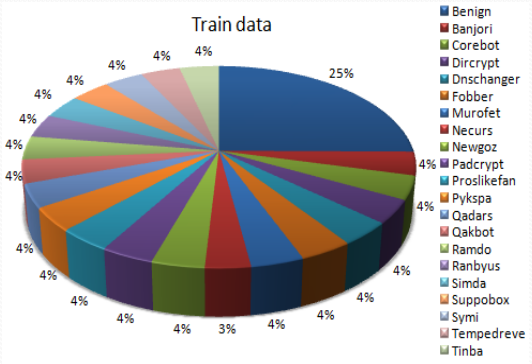
Command and Control Center

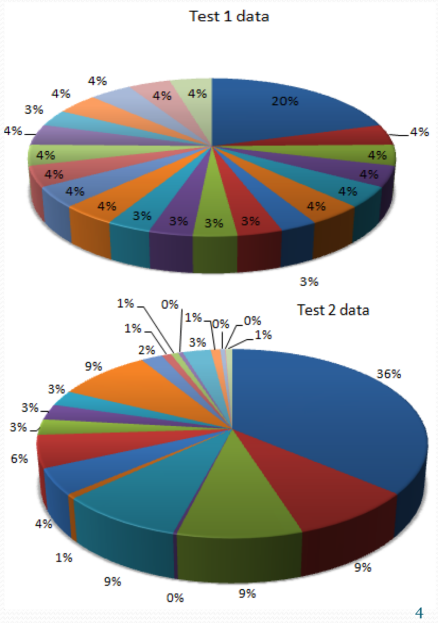
Dataset - Binary Classification



3

Dataset - Multiclass Classification





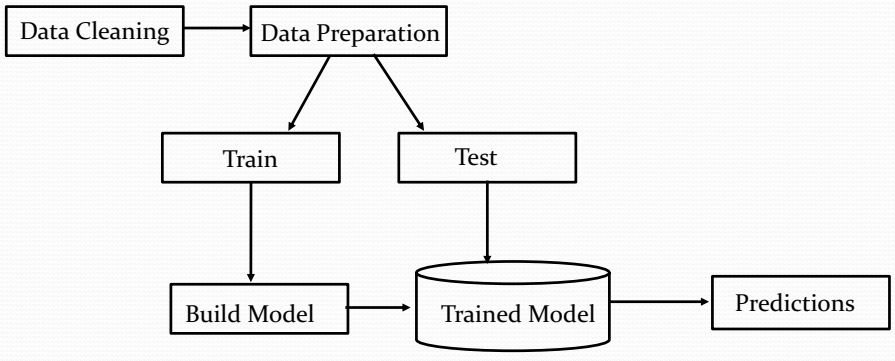
4

Results - Approach

	Binary Classification (Benign vs Malicious)	Multiclass Classification (Categorize among 21 families)
Featureless approach (Deep Learning)	Ranked #1 on Test-1 data	Please see paper
Featureful approach (Random Forest)	Please see paper	Ranked #1 on Test-2 data

5

Methodology



6

Additional Datasets

- **Alexa-Bambenek.** The AlexaBambenek dataset consists of the top 1M domains from Alexa[1] (considered benign) and 1M DGA domains from the OSINT feeds[2].
- **DGArchive**. The DGArchive dataset[3] is a repository of known DGA domains.
- **Real-Traffic.** The real-traffic dataset originates from a real-time stream of passive DNS data obtained from Farsight Security[4].

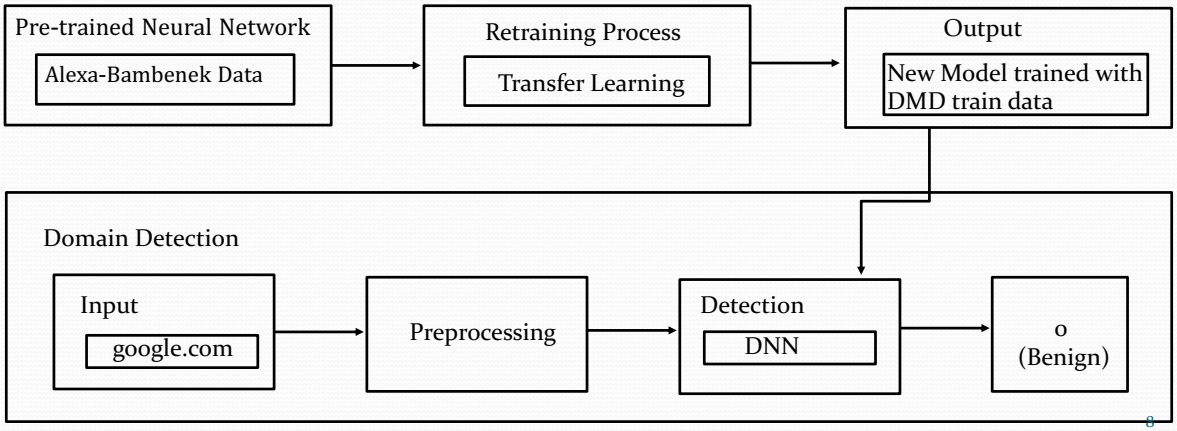
Dataset	Benign	DGA	Total
Alexa-Bambenek	1M	1M	2M
DGArchive	NA	15,772,535	15,772,535
Real-Traffic	15,534,803	18,247,899	33,782,702

[1] https: //support.alexa.com/hc/en-us/articles/200449834-Does-Alexa- have-a-list-of-its-top-ranked-websites-
[2] http://osint. bambenekconsulting.com/feeds/
[3] https://dgarchive.caad.fkie.fraunhofer.de/site/
[4] https://www.farsightsecurity.com/

7

Binary Classification

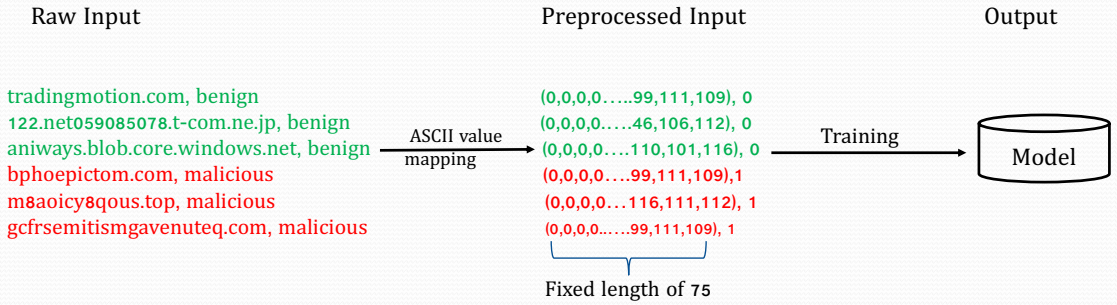
- Featureless



8

Binary Classification

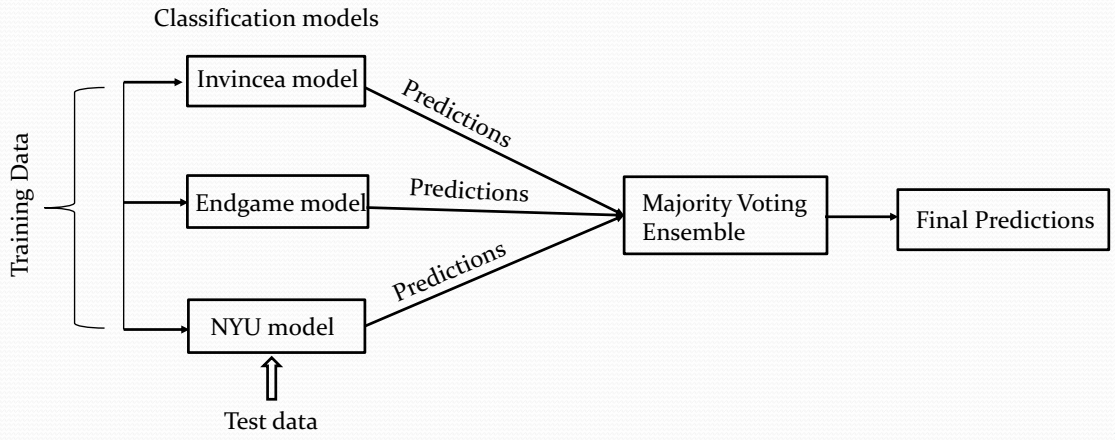
- Preprocessing



9

Binary Classification

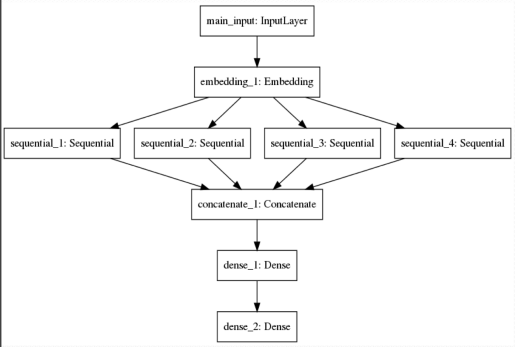
- Ensemble Method



10

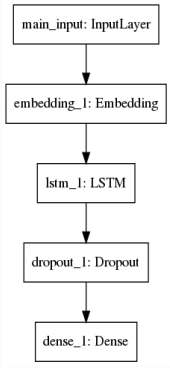
Binary Classification - Model Architectures

Invincea model



4 Parallel CNN layers each with 256 filters of size 2,3,4 & 5

Endgame model



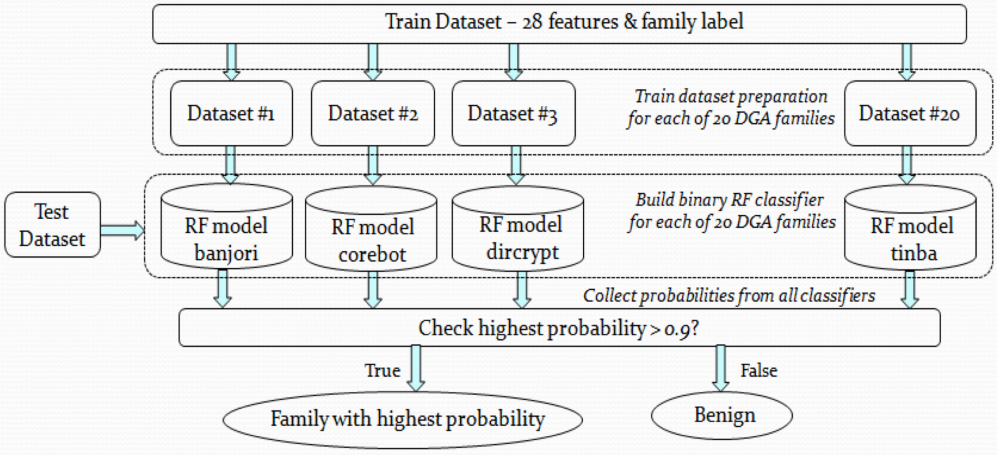
Single LSTM layer

NYU model

2 Stacked CNN layers

11

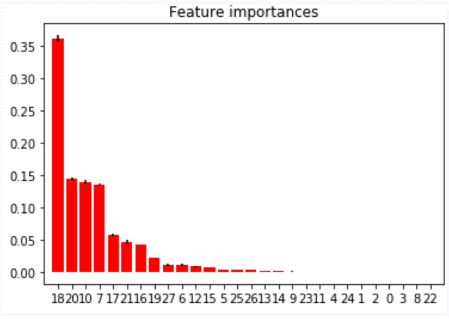
Multiclass Classification - One vs Rest Architecture



12

Multiclass Classification

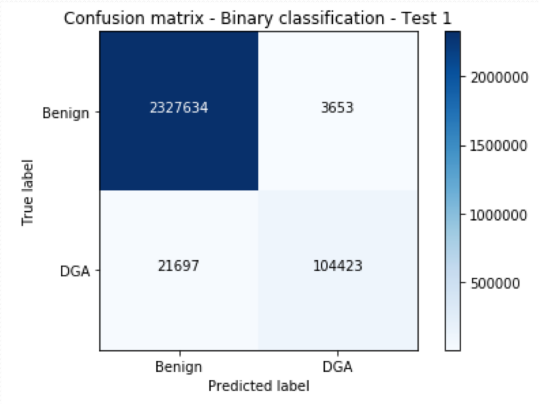
Feature Importance Ranking



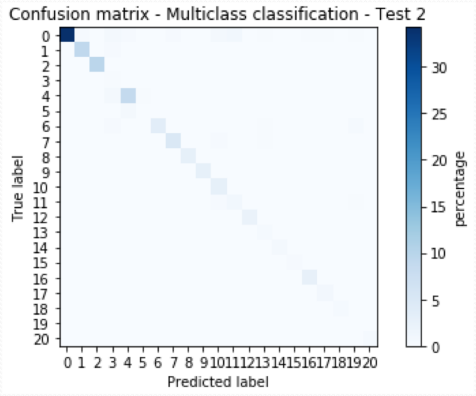

```
Feature ranking:
1. feature 18 (0.360991) sld_len          15. feature 26 (0.003795) uni_sld
2. feature 20 (0.144251) tld_hash         16. feature 13 (0.001105) flag_dga
3. feature 10 (0.139556) domain_len       17. feature 14 (0.000924) flag_dig
4. feature 7 (0.135614) dig               18. feature 9 (0.000734) digits_sld
5. feature 17 (0.056352) lng_con_seq      19. feature 23 (0.000455) tokens_sld
6. feature 21 (0.046999) tld_len          20. feature 11 (0.000009) dots
7. feature 16 (0.042132) hex              21. feature 4 (0.000008) 3ld_len
8. feature 19 (0.021789) sym              22. feature 24 (0.000007) uni_3ld
9. feature 27 (0.011048) vow              23. feature 1 (0.000005) 2gram_med
10. feature 6 (0.010904) con              24. feature 2 (0.000004) 3gram_cmed
11. feature 12 (0.008481) ent             25. feature 0 (0.000002) 2gram_cmed
12. feature 15 (0.006902) gni             26. feature 3 (0.000001) 3gram_med
13. feature 5 (0.004044) cer              27. feature 8 (0.000000) digits_3ld
14. feature 25 (0.003888) uni_domain      28. feature 22 (0.000000) tokens_3ld
```

13

Confusion Matrix for best models



Binary Classification



Multiclass Classification

14

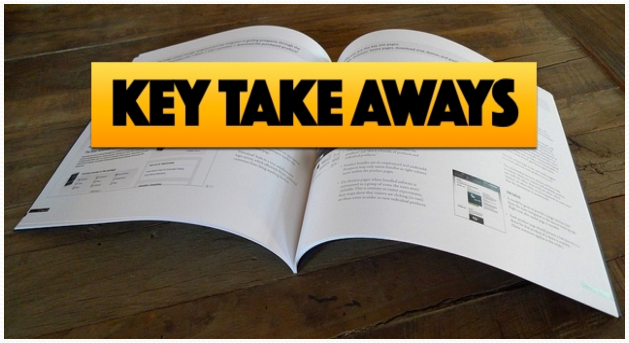
Results - DMD

Task	Dataset	Accuracy	F1-score	Recall	Precision	Ranking
Binary Classification	Test1	**99.0%**	0.892	0.966	0.828	1
	Test2	**76.6%**	0.858	0.999	0.751	3
Multiclass Classification	Test1	**63.3%**	0.602	0.633	0.618	5
	Test2	**88.7%**	0.901	0.887	0.924	1

15

Key Takeaways

- DGAs can be effectively detected by machine learning models
- Both featureful and featureless approaches have a valuable role to play in the defense against malware
- More data helps
- Transfer Learning is helpful



16

Thank you!

Questions?

Chhaya Choudhary :: chhayc@uw.edu
Raaghavi Sivaguru :: raaghavi@uw.edu

Appendix B

B.1 Code, data & model for Chapter 3

File	**Location on UWT MCS1 machine**
Code	/home/raaghavi/thesis/chapter3/code
Data	/home/raaghavi/thesis/chapter3/data
Models	/home/raaghavi/thesis/chapter3/models

Table B.1: Location of files used in Chapter 3

B.2 Slides for Chapter 3

AN EVALUATION OF DGA CLASSIFIERS

RAAGHAVI SIVAGURU[1], CHHAYA CHOUDHARY[1], BIN YU[2], VADYM TYMCHENKO[2], ANDERSON NASCIMENTO[1], MARTINE DE COCK[1]

UNIVERSITY OF WASHINGTON TACOMA[1], INFOBLOX[2]

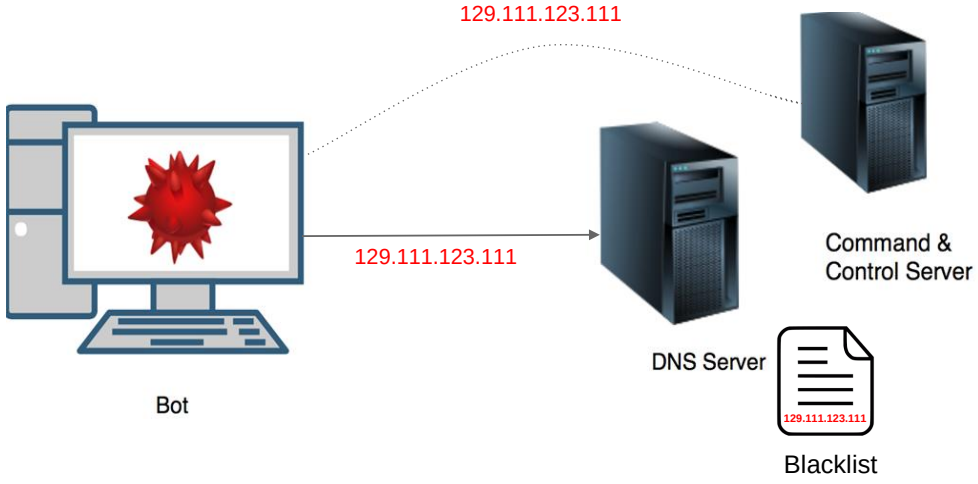
MOTIVATION

Percentage increase in average annual number of security breaches **27.4%**

Cyber crime damage costs to hit $6 trillion annually by 2021.

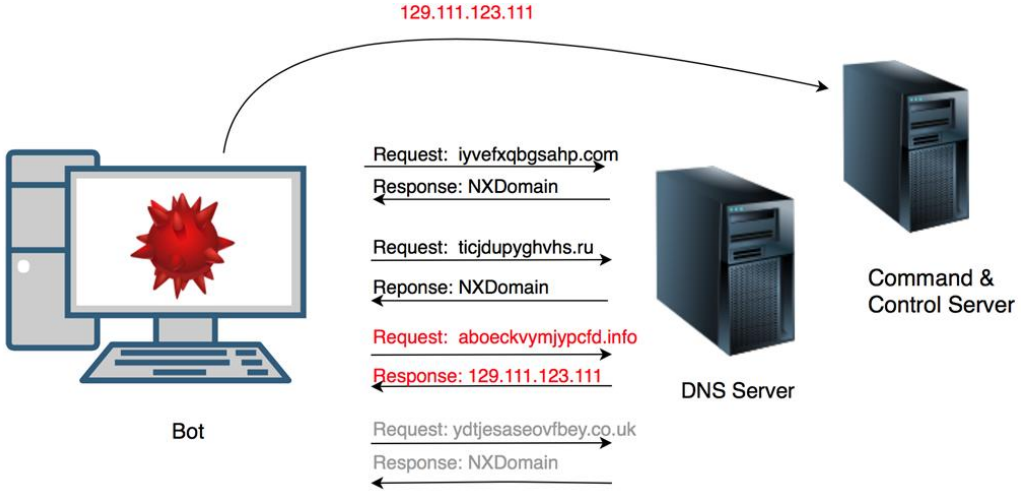
Ref: https://www.accenture.com/t20170926T072837Z__w__/us-en/_acnmedia/PDF-61/Accenture-2017-CostCyberCrimeStudy.pdf

TRADITIONAL BOT COMMUNICATION



3

DOMAIN GENERATION ALGORITHM - **DGA**



4

EXAMPLE: CRYPTOLOCKER DGA


```
def generate_domain (year, month, day):
  """ Generates a domain name for the given date. """

  domain = ""

  for i in range (16):
     year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
     month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
     day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
     domain += chr(((year ^ month ^ day) % 25) + 97)

  domain = domain + '.org'
  return domain

Generated domains:
generate_domain(2018, 8, 2)   - evtddtqrfchttjcl.org
generate_domain(2018, 9, 27)  - iqqvqbnbhjedlway.org
generate_domain(2018, 10, 10) - wlsjpnmuouwfivmh.org
generate_domain(2018, 11, 21) - hlxclhpmcajwaquf.org
```

This DGA algorithm uses -

- Current date as seed
- A Pseudo-Random Generator (PRNG)

5

EXAMPLE: DGA VS BENIGN

DGA - zvwoardenslavetusul.com

vs

Benign - maluliarmarinhos.com.br

6

TIME-DEPENDENT & TIME-INVARIANT DGA

DGA domain	Time-dependent (TD) / Time-invariant DGA (TI)	DGA Family
kmlhxjymatn.com	TI	ramnit
hntbxoqybxhqjmcqwl.mx	TD	necurs
tlgypartbulkyf.com	TI	banjori
vrlnorirnnlyrn.com	TD	murofet
lvocyxgffeux.com	TI	tinba

Time-dependent DGA - Generates different domains at different times

Time-invariant DGA - Generates same domains all the time when it runs

Time-invariant DGA and Time-dependent DGA - From DGArchive

7

CONTRIBUTIONS

- ❏ Evaluate how change in seed of the DGA generator affects classifier performance
- ❏ Investigate the behavior of the state-of-the-art classifiers when exposed to real traffic
- ❏ Classification based on domain names only - no additional information is required

8

STATE OF THE ART CLASSIFIERS

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

[**CMU**] *"Tweet2vec: Character-based distributed representations for social media", B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. Cohen, 2016.*

[MIT] *Tweet2vec:Learning tweet embeddings using character-level CNN-LSTM encoder-decoder", S.Vosoughi, P.Vijayaraghavan, and D.Roy, 2016.*

[**LSTM.MI**] *"A LSTM based framework for handling multiclass imbalance in DGA botnet detection", Tran, D., Mac, H., Tong, V., Tran, H.A. and Nguyen, L.G., 2018.*

9

DATASETS

- **Source:** Alexa, Bambenek, DGArchive, real traffic
- **Benign:** Intersection of Alexa with real traffic
- **DGA:** **TD:** Intersection of Bambenek with real traffic
  TI: Collection of seeds from DGArchive

	Family	No. of domains: Day1-6	No. of domains: Day7
TD	**Cryptolocker-Flashback**	7,469	2,609
	dyre	7,766	2,181
	locky	9,234	3,883
	murofet	49,284	22,383
	necurs	58,799	28,684
	nymaim	12,496	6,048
	Post Tovar GOZ	98,873	32,851
	pykspa	19,964	14,261
	qakbot	38,446	18,184
	ranbyus	14,367	13,806
	Total	316,698	144,890
TI	**banjori**	439,207	439,205
	tinba	59,564	53,995
	ramnit	56,138	56,140
	simda	14,729	14,730
	shiotob/urlzone/bebloh	12,521	12,519
	Total	582,159	576,589
	Alexa	884,752	792,278

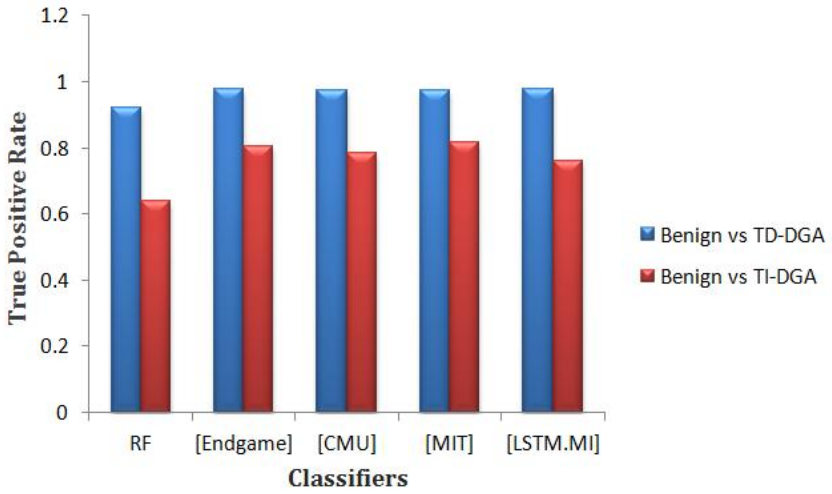
10

CLASSIFIERS PERFORMANCE

FEATURES	MODEL	ARCHITECTURE	TPR*
28 features	RF	Random Forest	0.8843
Featureless	[Endgame]	Single LSTM layer	0.9469
	[CMU]	Bidirectional LSTM layer	0.9432
	[MIT]	Stacked CNN layer + LSTM layer	0.9478
	[LSTM.MI]	Cost sensitive LSTM layer	0.9524

11 *All True Positive Rates listed in the table correspond to a False Positive Rate of 0.1%

CLASSIFIER PERFORMANCE



12

REAL TRAFFIC ANALYSIS

- **Classifier** LSTM.MI
- **Test Data** 57M resolved domains from real traffic (Aug 27, 2018)
- **Result** 1.45% of the domains were flagged as DGA
- **Analysis** Most of the flagged DGAs were benign

13

BENIGNS FLAGGED AS DGA

80776r432554442211122138247243.blogspot.com
goroskopkjvlkxl.appspot.com
wkdtjsgur100.github.io
eee3c5a49ca4b8c645d6b62830c6db6c.netlify.com
dfszgdfhgh.ddns.net
s10fgx928q3pu.cloudfront.net
ezs3f2a3de0adb3506ae3d28f8f25edfa9c8.s3.amazonaws.com
pi5254614543fgbsjhfghddsd44.herokuapp.com
f17f9711cc6764abe2f102a25ca970a6.bitballoon.com

14

REAL TRAFFIC ANALYSIS - EXTENDED

- ❏ Modify existing training data set
- ❏ Replace Alexa with 1M resolved domains from real traffic (Aug 10)
- ❏ Re-train LSTM.MI classifier
- ❏ Re-test the classifier on real traffic - Aug 27, 2018

15

COMPARISON OF RESULTS

	LSTM.MI TRAINED USING ALEXA	LSTM.MI TRAINED USING REAL TRAFFIC
Flagged as DGA	839,212 (1.45%)	146,067 (0.2%)
Found in Bambenek and DGArchive	6,068	5,623
Found in Alexa	774	674

16

KEY TAKEAWAYS

- Time invariant (TI) DGAs are harder to detect than time dependent (TD) DGAs
- Alexa* represents website statistics which is different from real DNS traffic
- Classifiers trained on real traffic domains substantially reduced false positives

17

*Alexa: https://www.alexa.com/topsites

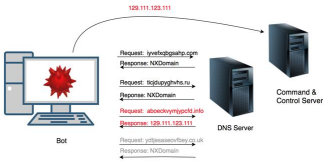
B.3 Poster for Chapter 3

An Evaluation of DGA Classifiers

Raaghavi Sivaguru, Chhaya Choudhary, Bin Yu, Vadym Tymchenko, Anderson Nascimento, Martine De Cock
raaghavi@uw.edu, chhayc@uw.edu, biny@infoblox.com, vtymchecko@infoblox.com, andclay@uw.edu, mdecock@uw.edu

Motivation

Domain Generation Algorithms (DGA)



"DGAs are widely used by malware to establish a communication link between the infected machine and command-and-control (C&C) server to perform malicious activities"

State-of-the-art classifiers for DGA detection

Endgame	Single LSTM layer	*"Predicting domain generation algorithms with long short-term memory networks", J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, 2016.*
CMU	Bi-directional LSTM layer	*"Tweet2vec: Character-based distributed representations for social media", B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. Cohen, 2016.*
MIT	Stacked CNN + LSTM layer	*Tweet2vec:Learning tweet embeddings using character-level CNN-LSTM encoder-decoder", S.Vosoughi, P.Vijayaraghavan, and D.Roy, 2016.*
LSTM.MI	Cost sensitive LSTM layer	*"A LSTM based framework for handling multiclass imbalance in DGA botnet detection", Tran, D., Mac, H., Tong, V., Tran, H.A. and Nguyen, L.G., 2018.*

Our Contributions

- Evaluate how change in seed of DGA generator affects classifier performance - i.e. detection of **TD-DGA** (time dependent DGA) and **TI-DGA** (time invariant DGA)
- Investigate the performance of state-of-the-art classifiers when they are exposed to real traffic

Methodology

Dataset preparation

- **Source** Alexa, Bambenek, DGArchive, Real traffic (passive DNS)
- **Benign** Domains that are found both in Alexa and real traffic (negative examples)
- **DGA** Domains that are found both in Bambenek and real traffic (positive examples)

Experiment 1: Evaluation of state-of-the-art classifiers in DGA detection

- **Train:** Real traffic domains from day 1 through day 6 (July 19 – July 25 2018)
- **Test:** Real traffic domains from day 7 (Aug 6)
- **Task:** Train classifiers for DGA detection (binary) & DGA family classification (multiclass)

Experiment 2: Evaluation of classifier robustness against change in seed of the DGAs

- **Benign vs TD-DGA** **Train:** Real traffic domains from day 1- day 6 **Test:** Real traffic domains from day 7
- **Benign vs TI-DGA** Seed information of the domains are obtained from DGArchive **Train:** Domains belonging to 80% of seeds **Test:** Domains belonging to 20% of seeds

Experiment 3: Performance evaluation of classifier against real traffic

- **Classifier** LSTM.MI DGA detection classifier from Experiment 1
- **Test data** 57 million resolved real traffic domains on Aug 27, 2018
- **Result** 1.45% of the domains in test data were flagged as DGA
- **Analysis** Most of the flagged DGAs were benign (This included content distribution networks, wild card domains etc.)

Experiment 4: Performance evaluation after retraining the classifier with real traffic data set

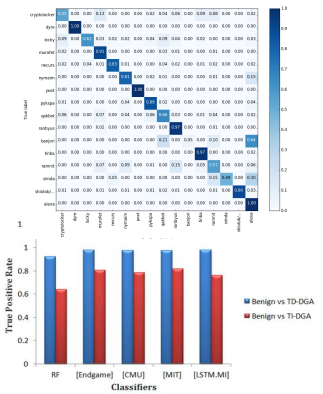
- Modify train data by replacing **Alexa** with 1M resolved domains from real traffic on Aug 10, 2018
- Retrain **LSTM.MI** classifier and test it against 57M resolved real traffic domains on Aug 27, 2018

Results

Metrics used for evaluation

- **True Positive Rate (TPR):** % of DGA domains detected correctly
- **False Positive Rate (FPR):** % of benign domains flagged as DGA
- **Area Under ROC Curve (AUC)**

Experiment 1 & 2: Evaluation of classifier robustness



Key Observation:
"Classifiers are more robust to change in time than change in seed of the DGAs"

Experiment 3 & 4: Evaluation of classifier in real traffic

	LSTM.MI trained using Alexa	LSTM.MI trained using real traffic
# flagged as DGA	839,212 (1.45%)	146,067 (0.2%)
# found in DGArchive and Bambenek	6,068	5,623
#found in Alexa	774	674

Key Takeaway:
"Performance of DGA detection classifiers can be improved by training with real datasets since synthetic datasets (Alexa) may not be a sufficient representation of real DNS traffic"

References

Sivaguru, R., Choudhary, C., Yu, B., Tymchenko, V., Nascimento, A. and De Cock, M., 2018, December. An Evaluation of DGA Classifiers. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 5058-5067). IEEE.

CENTER FOR DATA SCIENCE
UNIVERSITY of WASHINGTON | TACOMA

Infoblox
CONTROL YOUR NETWORK

GHENT UNIVERSITY

Appendix C

C.1 Code, data & model for Chapter 4

File	Location on UWT MCS1 machine
Code	/home/raaghavi/thesis/chapter4/code
Data	/home/raaghavi/thesis/chapter4/data
Models	/home/raaghavi/thesis/chapter4/models

Table C.1: Location of files used in Chapter 4

Appendix D

D.1 Code, data & model for Chapter 5

File	Location on UWT MCS1 machine
Code	/home/raaghavi/thesis/chapter5/code
Data	/home/raaghavi/thesis/chapter5/data
Models	/home/raaghavi/thesis/chapter5/models

Table D.1: Location of files used in Chapter 5

D.2 Slides for Chapter 5

Hardening Inline DGA Classifiers Against Adversarial Attacks

Raaghavi Sivaguru

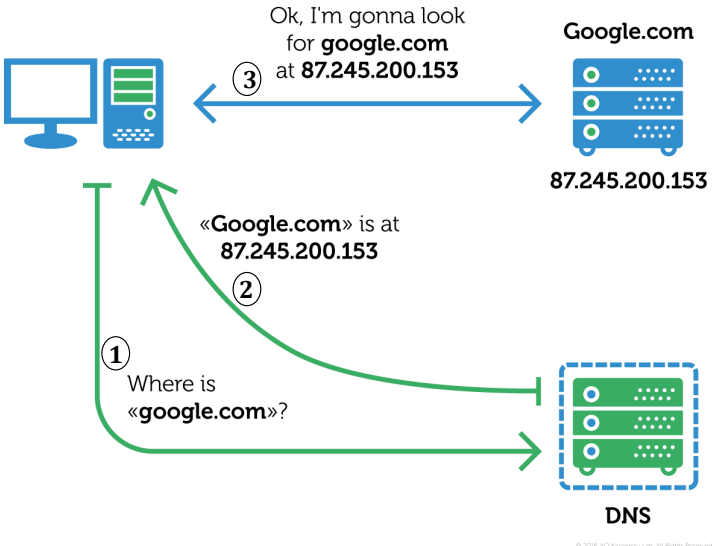
Committee Chair: Dr. Martine De Cock (UWT)
Committee Members: Dr. Anderson (UWT), Dr. Femi (Infoblox), Jonathan (Ghent University)

December 04, 2019

UNIVERSITY of WASHINGTON

Infoblox
CONTROL YOUR NETWORK

Domain Name System (DNS)



Reference URL: https://media.kasperskydaily.com/wp-content/uploads/sites/92/2016/12/06015704/dns-normal-en-1024x791.png

Bot & Botmaster

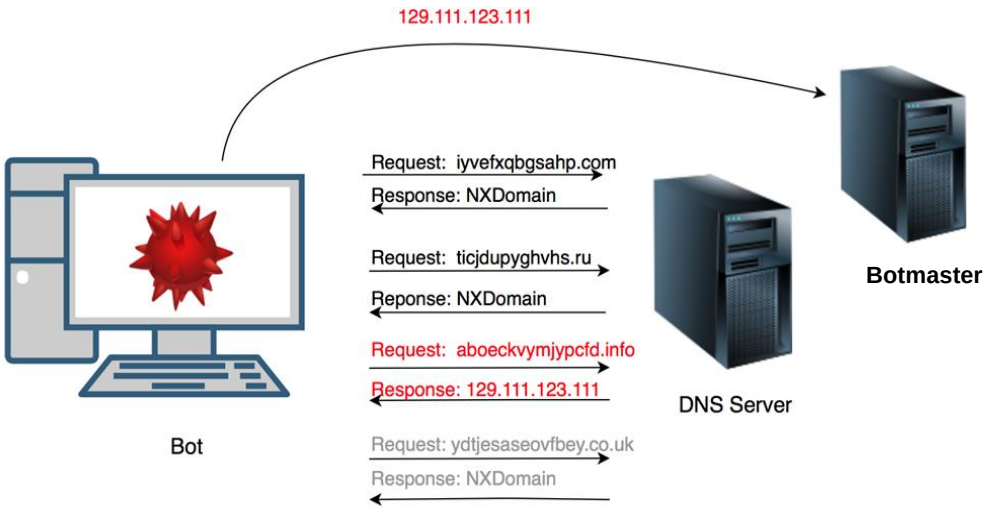
- **Bot:** A compromised machine that performs malicious activities.
- **Botmaster:** Instructs bots connected to it to perform malicious activities.
- **How do they communicate?:** Domain Generation Algorithms (DGAs)

Domain Generation Algorithm (DGA)

- **What are DGAs?:** Algorithms that generate large number of pseudo-random sequence of characters/words.
- **Why are they used?:** To establish on-the-go communication between bot & botmaster.
- **Problem:** Usually short-lived, hence detecting/blacklisting is hard.

```
pkptulovkvsnoqs.net.
3e51a09b273b.com.
fmepzyjkjaseijprw.net.
lsxjaproznyimcx.net.
bsdmxulibezsrsnq.com.
lmyqkrquloza.net.
ovuhsftqngzqgyr.net.
vpkovkupwqlnowc.biz.
rexlpmdttquoxzru.com.
qlccnywxnodpmqdhoco.net.
beautifulplaste5xtt8l.com.
rczcvkpqlepcz.com.
xofwzjqmnryl.net.
nozmjcbsvyzixej.com.
1844a4xe50dm6v0o0maloa7c.net.
tlkpqporqtofmjv.biz.
mausaqcseu.org.
tkzsgruvgyhpttol.net.
```

Communication Using DGAs



Examples – DGAs vs Non-DGAs

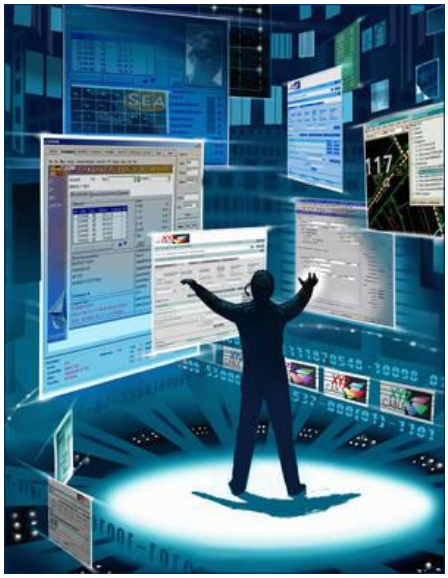
DGAs (malicious/positive samples)	Non-DGAs (benign/negative samples)
diezee.com	7ft4.com
kljtuyhgerwxlrvwzlpqfanz.org	bobbyssupermarket.com
lcmwdypbxqwnbrwfmnjbfqlsg.info	artisticflowers-decor.com
leadhelp.net	nextgeneration-homesolutions.com
deeptouch.net	essc-tabriz.com
spasms.com	sgtobel.ch
marknamibiacaptured.com	pedaltrain.com

bobbyssupermarket.com

SLD TLD

Machine Learning for DGA detection

- **Featureful approach:**
  - Involves manual extraction of features from domain names.
  - Use supervised ML algorithms such as SVM, RF etc.
  - Effective baselines
- **Featureless approach:**
  - ML algorithms automatically learn features from domain names.
  - Includes methods such as CNN, LSTM etc.
  - State-of-the-art techniques in DGA detection



Adversarial Attacks

- Attacks that are intended to fool ML classifiers
- An example: CharBot*

Input:
D = {google, facebook, …, yahoo} → List of SLDs from non-DGAs
T = {com, edu, co.in, …, net} → List of TLDs
s = 1575299615 → seed for reproducibility

Domain generation:
d = google; i=2; j=5; $c1$=z; $c2$=9; t=edu → Randomly chosen values

Output: gzog9e.edu

*Peck, Jonathan, Claire Nie, **Raaghavi Sivaguru**, Charles Grumer, Femi Olumofin, Bin Yu, Anderson Nascimento, and Martine De Cock. **"CharBot: A Simple and Effective Method for Evading DGA Classifiers."** *arXiv preprint arXiv:1905.01078* (2019).

Current Methods

- **Adversarial training:**
  - Append adversarial samples to training set & train the model iteratively
  - Lack of generalizability for new attacks
- **Graph inference:**
  - Analyzes/derives patterns from past traffic
  - Requires high storage & computation
- **Methods using external databases:**
  - Gathers additional features from WHOIS
  - Often unreliable

Proposed Method

- **Inline DGA detection using side-information:**
  - Perform inline DGA detection using domain name & additional information that can be easily obtained from DNS query/response
  - Use features that cannot be manipulated by the attacker
  - No compromise on performance of existing state-of-the-art DGA classifiers

Experiments

1. Train DGA classifiers using side-information & compare it's performance against state-of-the-art DGA classifier.

2. Expose the above classifiers to one day of real-traffic domains.

3. Measure classifier robustness against adversarial samples.

Experimental Details

- **Data:**
  - Real DNS traffic data from Feb 2019 – provided by Infoblox
  - Data size – 1.2M domains (600k DGAs & 600k non-DGAs)

- **Features:**
  - Domain name string (1)
  - Lexical features (26)
  - DNS features (9)

- **Classifiers:**
  - LSTM.MI*
  - B-RF

*Tran, Duc, et al. "A LSTM based framework for handling multiclass imbalance in DGA botnet detection." *Neurocomputing* 275 (2018): 2401-2413.

Features used in B-RF

#	Lexical Feature
1	Domain name length
2	Second level domain length
3	Top level domain length
4	Domain Unique Characters length
5	SLD Unique Characters length
6	TLD Unique Characters length
7	Has malicious TLD
9	TLD Hash
9	Starts with Digit
10	Symbol ratio
11	Hex ratio
12	Digit Ratio
13	Vowel Ratio
14	Consonant Ratio
15	Ratio of Repeated Characters
16	Ratio of Consecutive Consonants
17	Ratio of Consecutive Digits
18	Number of tokens in SLD
19	Number of digits in SLD
20	Entropy
21	Gini Index
22	Classification error of characters
23	2-Gram Median
24	3-Gram Median
25	2-Gram Circle Median
26	3-Gram Circle Median

Lexical Features

Feature	Description
rrlength	Resource record length
country	Country name that the domain maps to
ttl	Time-to-live of the DNS query
n_ip	Number of distinct IP addresses the domain maps to
qtype	Type of DNS packet requested
rtype	Record type of the DNS response
n_asn	Number of distinct ASNs the domain maps to
subnet	Do all IPs belong to same subnet
n_countries	Number of distinct countries the domain maps to

Side-information Features

Step 1: Training & Evaluation

- **Evaluation strategy:** 5-fold cross-validation
- **Performance Metrics:** Area Under Curve (AUC) & True Positive Rate (TPR)

Model	Features	Performance Metrics		
		AUC	AUC@ 0.1%FPR	TPR@ 0.1%FPR
LSTM.MI	Domain name	99.98%	94.47%	98.80%
B-RF	DNS	97.92%	53.23%	16.21%
	Lexical	99.91%	89.78%	97.44%
	Lexical + DNS	**99.98%**	**98.19%**	**99.42%**
LSTM.MI + B-RF	Domain + DNS	99.98%	96.51%	99.89%
	Domain name + Lexical + DNS	**99.99%**	**99.17%**	**99.91%**

Step 2: Evaluate on One Day DNS Traffic

- ❑ Expose best DGA classifiers to one day of real traffic (March 26, 2019)
- ❑ Number of domains – 66,440,681 (~66M)
- ❑ Number of domains in DGArchive* – 1,159,662 (~1.1M)

Classifier	**LSTM.MI**	**B-RF**	**LSTM.MI + B-RF**
Features	Domain name string	Lexical + DNS	Domain + Lexical + DNS
Number of domains flagged as DGA	3,400,017	**1,877,784**	2,170,056
Number of domains found in DGArchive*	1,151,750	**1,149,689**	1,150,116

***DGArchive** – Public blacklist (DGAs);

Step 3: Robustness against Adversaries

- ❑ **Adversarial sample: CharBot (1000 domains)**
  - Extract lexical features for CharBot domains.
  - CharBot has no DNS data – hence DNS features are randomly sampled (5 trials) from known DGAs.
  - Measure detection rate – proportion of CharBot domains detected as DGA by the classifier.

Model	**Features**	**CharBot detection rate**
LSTM.MI	Domain name	8.00 ± 0.00%
B-RF	Lexical + DNS	**20.06 ± 0.56%**
LSTM.MI + B-RF	Domain + Lexical + DNS	15.76 ± 0.53%

Summary

- DGA classifiers trained using both lexical & side information (DNS) features outperforms state-of-the-art LSTM classifier.
- Achieves higher TPR (True Positive Rate).
- Performs inline detection.
- Robust against adversarial samples.
- No knowledge on attack algorithm is needed.

Publications

- Choudhary, Chhaya, **Raaghavi Sivaguru**, Mayana Pereira, Bin Yu, Anderson C. Nascimento, and Martine De Cock. **"Algorithmically generated domain detection and malware family classification."** In *International Symposium on Security in Computing and Communication*, pp. 640-655. Springer, Singapore, 2018.

- **Raaghavi Sivaguru**, Chhaya Choudhary, Bin Yu, Vadym Tymchenko, Anderson Nascimento, and Martine De Cock. **"An evaluation of DGA classifiers."** In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 5058-5067. IEEE, 2018.

- Peck, Jonathan, Claire Nie, **Raaghavi Sivaguru**, Charles Grumer, Femi Olumofin, Bin Yu, Anderson Nascimento, and Martine De Cock. **"CharBot: A Simple and Effective Method for Evading DGA Classifiers."** *arXiv preprint arXiv:1905.01078* (2019).

- Journal paper of thesis to be submitted soon.

BIBLIOGRAPHY

[1] Jasper Abbink and Christian Doerr. Popularity-based Detection of Domain Generation Algorithms. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, number 79. ACM, 2017.

[2] Hyrum S Anderson, Jonathan Woodbridge, and Bobby Filar. DeepDGA: Adversarially-Tuned Domain Generation and Detection. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, pages 13–21, 2016.

[3] Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou, and David Dagon. Detecting Malware Domains at the Upper DNS Hierarchy. In *USENIX Security Symposium*, volume 11, pages 1–16, 2011.

[4] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In *USENIX Security Symposium*, volume 12, pages 491–506, 2012.

[5] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In *Ndss*, pages 1–17, 2011.

[6] Leyla Bilge, Sevil Sen, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains. *ACM Transactions on Information and System Security (TISSEC)*, 16(4), 2014.

[7] Tommy Chin, Kaiqi Xiong, Chengbin Hu, and Yi Li. A Machine Learning Framework for Studying Domain Generation Algorithm DGA-based Malware. In *International Conference on Security and Privacy in Communication Systems*, pages 433–448. Springer, 2018.

[8] Chhaya Choudhary, Raaghavi Sivaguru, Mayana Pereira, Bin Yu, Anderson C Nascimento, and Martine De Cock. Algorithmically Generated Domain Detection and Malware Family classification. In *International Symposium on Security in Computing and Communication*, pages 640–655. Springer, 2018.

[9] Ryan R Curtin, Andrew B Gardner, Slawomir Grzonkowski, Alexey Kleymenov, and Alejandro Mosquera. Detecting DGA Domains with Recurrent Neural Networks and Side Information. *arXiv preprint arXiv:1810.02023*, 2018.

[10] Ryan R Curtin, Andrew B Gardner, Slawomir Grzonkowski, Alexey Kleymenov, and Alejandro Mosquera. Detecting DGA Domains with Recurrent Neural Networks and Side Information. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*. ACM, 2019.

[11] Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. Tweet2vec: Character-based Distributed Representations for Social Media. *arXiv preprint arXiv:1605.03481*, 2016.

[12] Yu Fu, Lu Yu, Oluwakemi Hambolu, Ilker Ozcelik, Benafsh Husain, Jingxuan Sun, Karan Sapra, Dan Du, Christopher Tate Beasley, and Richard R Brooks. Stealthy Domain Generation Algorithms. *IEEE Transactions on Information Forensics and Security*, 12(6):1430–1443, 2017.

[13] Apoorva Joshi, Levi Lloyd, Paul Westin, and Srini Seethapathy. Using Lexical Features for Malicious URL Detection–A Machine Learning Approach. *arXiv preprint arXiv:1910.06277*, 2019.

[14] Issa Khalil, Ting Yu, and Bei Guan. Discovering Malicious Domains through Passive DNS Data Graph Analysis. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 663–674, 2016.

[15] Joewie Koh and Barton Rhodes. Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings. In *Proceedings of 2018 IEEE International Conference on Big Data*, pages 2965–2970, 2018.

[16] Jonghoon Kwon, Jehyun Lee, Heejo Lee, and Adrian Perrig. PsyBoG: A Scalable Botnet Detection Method for Large-Scale DNS Traffic. *Computer Networks*, 97:48–73, 2016.

[17] Yi Li, Kaiqi Xiong, Tommy Chin, and Chengbin Hu. A Machine Learning Framework for Domain Generation Algorithm DGA-Based Malware Detection. *IEEE Access*, 7:32765–32782, 2019.

[18] Pierre Lison and Vasileios Mavroeidis. Automatic Detection of Malware-Generated Domains with Recurrent Neural Models. *preprint arXiv:1709.07102*, 2017.

[19] Pierre Lison and Vasileios Mavroeidis. Neural Reputation Models learned from Passive DNS Data. In *2017 IEEE International Conference on Big Data*, pages 3662–3671, 2017.

[20] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In *Proceedings of*

the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1245–1254, 2009.

[21] Jonathan Peck, Claire Nie, Raaghavi Sivaguru, Charles Grumer, Femi Olumofin, Bin Yu, Anderson Nascimento, and Martine De Cock. CharBot: A Simple and Effective Method for Evading DGA Classifiers. *IEEE Access*, 7:91759–91771, 2019.

[22] Mayana Pereira, Shaun Coleman, Bin Yu, Martine De Cock, and Anderson Nascimento. Dictionary Extraction and Detection of Algorithmically Generated Domain Names in Passive DNS Traffic. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 295–314. Springer, 2018.

[23] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. A Comprehensive Measurement Study of Domain Generating Malware. In *25th USENIX Security Symposium*, pages 263–278, 2016.

[24] Joshua Saxe and Konstantin Berlin. eXpose: A Character-level Convolutional Neural Network with Embeddings for Detecting Malicious URLs, File Paths and Registry Keys. *arXiv preprint arXiv:1702.08568*, 2017.

[25] Stefano Schiavoni, Federico Maggi, Lorenzo Cavallaro, and Stefano Zanero. Phoenix: DGA-based Botnet Tracking and Intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 192–211. Springer, 2014.

[26] Samuel Schüppen, Dominik Teubert, Patrick Herrmann, and Ulrike Meyer. FANCI: Feature-based Automated NXDomain Classification and Intelligence. In *27th USENIX Security Symposium*, pages 1165–1181, 2018.

[27] Lior Sidi, Asaf Nadler, and Asaf Shabtai. MaskDGA: A Black-box Evasion Technique Against DGA Classifiers and Adversarial Defenses. *arXiv preprint arXiv:1902.08909*, 2019.

[28] Manmeet Singh, Maninder Singh, and Sanmeet Kaur. Detecting bot-infected machines using DNS fingerprinting. *Digital Investigation*, 28:14–33, 2019.

[29] Raaghavi Sivaguru, Chhaya Choudhary, Bin Yu, Vadym Tymchenko, Anderson Nascimento, and Martine De Cock. An Evaluation of DGA Classifiers. In *2018 IEEE International Conference on Big Data*, pages 5058–5067, 2018.

[30] Jan Spooren, Davy Preuveneers, Lieven Desmet, Peter Janssen, and Wouter Joosen. Detection of Algorithmically Generated Domain Names used by Botnets: A Dual Arms

Race. In *Proceedings of the 34th ACM/SIGAPP Symposium On Applied Computing*, pages 1902–1910. Association for Computing Machinery, 2019.

[31] Duc Tran, Hieu Mac, Van Tong, Hai Anh Tran, and Linh Giang Nguyen. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing*, 275:2401–2413, 2018.

[32] Yevgeniy Vorobeychik and Murat Kantarcioglu. Adversarial Machine Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–169, 2018.

[33] Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb 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*, pages 1041–1044. ACM, 2016.

[34] Zhen Wang, Zhongtian Jia, and Bo Zhang. A Detection Scheme for DGA Domain Names Based on SVM. In *2018 International Conference on Mathematics, Modelling, Simulation and Algorithms (MMSA 2018)*. Atlantis Press, 2018.

[35] Lanier Watkins, Sean Beck, Jared Zook, Anna Buczak, Jeffery Chavis, William H Robinson, Jose A Morales, and Samuel Mishra. Using Semi-supervised Machine Learning to Address the Big Data Problem in DNS Networks. In *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, 2017.

[36] Jonathan Woodbridge, Hyrum S Anderson, Anjum Ahuja, and Daniel Grant. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. *preprint arXiv:1611.00791*, 2016.

[37] Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Reddy, and Supranamaya Ranjan. Detecting Algorithmically Generated Malicious Domain Names. In *Proc. of the 10th ACM SIGCOMM Conference on Internet Measurement*, pages 48–61, 2010.

[38] Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Narasimha Reddy, and Supranamaya Ranjan. Detecting Algorithmically Generated Domain-Flux Attacks with DNS Traffic Analysis. *IEEE/ACM Transactions on Networking*, 20(5):1663–1677, 2012.

[39] Bin Yu, Daniel L Gray, Jie Pan, Martine De Cock, and Anderson CA Nascimento. Inline DGA Detection with Deep Networks. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 683–692, 2017.

[40] Bin Yu, Jie Pan, Daniel Gray, Jiaming Hu, Chhaya Choudhary, Anderson CA Nascimento, and Martine De Cock. Weakly Supervised Deep Learning for the Detection of Domain Generation Algorithms. *IEEE Access*, 7:51542–51556, 2019.

[41] Bin Yu, Jie Pan, Jiaming Hu, Anderson Nascimento, and Martine De Cock. Character Level Based Detection of DGA Domain Names. In *Proc. WCCI*, pages 4168–4175, 2018.

[42] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657, 2015.

[43] Yury Zhauniarovich, Issa Khalil, Ting Yu, and Marc Dacier. A Survey on Malicious Domains Detection through DNS Data Analysis. *ACM Computing Surveys (CSUR)*, 51(4), 2018.