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 featureless 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)