Domain Generating Algorithm based Malicious Domains Detection

Md. Ahsan Ayub, Steven Smith, Ambareen Siraj, and Paul Tinker
Department of Computer Science, Tennessee Tech University, Cookeville, USA
Emails: mayub42@tntech.edu, smsmith23@students.tntech.edu, asiraj@tntech.edu, and pjtinker42@students.tntech.edu

***Abstract*—Botnets often use Domain Generating Algorithms (DGAs) to facilitate covert server communication in carrying out different types of cyber-attacks. Attackers employ these algorithms to generate millions of sites for victim machines to connect to, thus evading defense using blacklists. DGAs enables attacks to be facilitated without the fear of command and control (C&C) servers being identified and permanently blocked. Utilizing the domain fluxing technique, attackers making use of DGAs can constantly change the domains used by their C&C servers from one to another in a very short time, whenever they are blocked. Therefore, automated detection of DGA generated domains can serve as an essential countermeasure to prevent malicious botnet communication promptly. In our research, we devise a comprehensive solution to detect malicious DGA generated domains used in malware communication. Two distinct feature extraction methods, the Bigram model and the Word2Vec model, are applied for text processing in combination with machine learning and deep learning techniques on a large and very diverse dataset for DGAs that exist currently, containing 84 different traditional and dictionary-based DGA families. Our results demonstrate exceptional success in both binary classification (classifying a given domain as benign or malicious), and multiclass classification (identifying the specific DGA variation or family that produced the domain).**

***Index Terms*—Bigram, Domain Generating Algorithm, Machine Learning, Malicious Domain Name, Malware, Word2Vec**

I. Introduction

Domain Generating Algorithms (DGAs) is a major technique for spoofed website domain name creation, which are used in many modern cyber-attacks, including botnet command and control (C&C) server communication and phishing [12]. Additionally, malicious Algorithmically Generated Domains (AGDs) produced by DGAs are often used to flood network logs with non-existent domains (NXDomain), which creates issues with manual network log analysis [1]. The DGA algorithms make use of domain fluxing, a technique used to query generated domain names rapidly but stealthily until a response is received from a C&C server. It is also used to change domain names and/or IP addresses used for the server in short intervals, which provide short windows in time for the detection of malicious activities. A DGA can generate hundreds to millions of potentially malicious domain names pseudo-randomly using a seed chosen by the malware author. This allows the attacker to have access to a massive number of domains, among which a small subset is then used for communication in any given time frame. If one domain is blacklisted, the attacker will simply move the server to one of the other malicious domains from the same subset. This has significantly reduced the effectiveness of defense mechanisms, such as blacklisting, reverse engineering, sinkholing, and preemptive registration of domains [4].

DGAs come in different forms and implementations. They enable botnets to conduct Distributed Denial of Service (DDOS) attacks, data exfiltration, and compromise of systems without detection. With botnets such as *Conficker*, *Kraken*, and *Torpig*, infected machines can query a large number of domain names, which appear to be random for C&C connections, *e.g.*, *http://uurvuqudmnnk1o.com* generated by the *Bedep* malware family. This type of DGAs is known as traditional. Additionally, attackers also use the Dictionary-Based DGAs technique that utilizes word dictionaries to create seemingly legit domain names, making detection methods that look for randomized domain names render ineffective. This type of DGAs can appear legitimate to network administrators and users as they often combine words to appear similar to real domains, *e.g.*, *http://journeystation.net* generated by the *Suppobox* malware family. This makes distinguishing between benign domains and AGDs even more challenging. New methods to detect botnet communication utilizing DGAs are of high importance to the security community and hence, the goal of this research. In this research, we list the following major contributions:

- A detailed experimental analysis of several machine and deep learning techniques is performed to demonstrate effective DGA binary and multiclass detection for both DGA types through the use of two powerful text processing feature extraction methods: Bigram and Word2Vec;
- VirusTotal API engine [10] is utilized for feature creation along with incorporating feature engineering techniques to extract features from the sole domain string; and
- A large-scale evaluation on samples from 84 different DGA families is conducted to show that our approach can effectively detect 69 different DGA families.

II. Datasets

We use the Majestic Million list of top-ranked domain names as our benign ground truth data [5] and DGArchive [9] as malicious domains dataset containing 84 different malware families split into 77 traditional and 7 dictionary-based DGAs.

Domain Scanning. We leverage two different techniques to analyze the dataset: first, we scan each domain using the VirusTotal API service [10], and then crosscheck whether a

domain returns an NXDomain (non-existent domain) response when queried through common DNS servers. We develop a VirusTotal API lookup engine to scan each domain. The domains flagged as malicious by VirusTotal are far more likely to be AGDs. We use DNS lookup to determine if the domains are active. Most of the domains returned NXDomain response since the domains used by the different DGA Algorithms are short-lived. The VirusTotal scans reveal that these domains are often found to be non-malicious. This approach shows that employing additional detection techniques on top of existing defense methods helps to combat both types of traditional and dictionary-based DGAs. We present the results of Virustotal scans for each malware family's domain in Table I.

A. VirusTotal Scan Report

As mentioned previously, we utilize the VirusTotal public API v2.0 to scan every domain in our dataset. This API service aggregates information from different antivirus products, website characterization tools, and website scanning engines.

After receiving the scan report for a given domain URL, we label the record with three distinct attributes: *no records* - when the domain is not known to VirusTotal; *safe* - when VirusTotal's scanning engines suggest a clean or non-malicious site; and *malicious* - if any scanning engine detects the domain as unsafe. Table I provides an analysis of these three attributes for each malware family. Additionally, we present the ratio of the number of malicious domains identified versus the total number of domains in the family for each family. Since the families only use a small subset of domain names from the generated domains, scan accuracy is very important. The scan report yield no results for 46.4% of the malware families (39 out of 84) because the domains were down and not previously scanned. It provides false-negative results for 25% of the malware families (21 out of 84) by identifying them as "safe". Accurate results are obtained for only 28.6% of the malware families (24 out of 84), where the scans flagged them as malicious. These results are consistent with our intuition that there would be a much higher percentage of domains with benign or unknown results due to the short-lived nature of the DGA domains. This allows many of these domains to escape detection by the engines before they switch to a different domain. Additionally, those viewed as benign most likely did not have a malicious payload. These were simply used for receiving stolen data or malicious communication rather than as a tool of compromise machines. In order to best illustrate these findings, the third column in Table I (*Scan Evaluation*) is presented to correlate each family's scan result with the previously mentioned attributes.

B. Observation of Characteristics

The following observations are also worthwhile for discussion. Seven of the malware families are dictionary-based DGAs: *Banjori*, *Downloader*, *Gozi*, *Matsnu*, *Necurs*, *Suppobox*, and *Volatile Ceder* while the remaining 77 make use of traditional DGAs. Out of the 7 dictionary-based DGAs, 4 were particularly concerning since VirusTotal did not flag them as malicious. Only one dictionary-based DGA (*Suppobox*) out of the 7 has been labeled with the malicious attribute. Additionally, none of the dictionary-based DGAs use numeric characters in their domain strings. The use of numeric characters is quite common for traditional DGAs, providing a clear differentiator between the two types.

Different families have different numbers of unique TLDs (Top Level Domains) such as .com, .edu, etc. Two malware families - *Necurs* and *Xxhex* generate domains using 44 distinct TLDs. This is very unusual since other families utilize only 1 to 14 unique TLDs. When checking the NXDomain status for each DGA family, we found that the number of existing domains varied from one DGA family to another. There are 10 DGAs that were found to have 20% or more active domains out of the 84 families. Among this subset, two families were found to have 100% live or active domains, and two other families contained over 50% live domains.

Human Engineered Features. Along with extracting the VirusTotal scan and NXDomain results for each domain for the use as features in our analysis, we have derived the following features to develop the machine learning classification models:

- The rounded-up percentage of the ratio of vowels to consonant, *i.e.*, the value for *www.google.com* is 50;
- The rounded-up percentage of the ratio of symbols to letters, *i.e.*, the value for *www.google.com* is 17;
- The length of the domain string, *i.e.*, the value would be 14 for *www.google.com*;
- The rounded-up percentage of the ratio of numeric to letters, *i.e.*, the value for *www.google.com* is 0; and
- The Top Level Domain (TLD) of the URL, *i.e.*, the TLD for *www.google.com* is *com*.

III. Experiment Methodology

We begin our experiment by analyzing the dataset and extracting human engineered features as per section 2(B). We then perform two different feature extraction methods applied to the domain strings: the Bigram model [3] and the Word2Vec model [6]. Derived vector space of the textual data later fed into several Machine Learning techniques and a Long Short-Term Memory (LSTM). The LSTM uses the gate structure to add/remove information to a cell state by employing a Sigmoid Neural Net layer (σ) followed by a pointwise multiplication. It works by ensuring no irrelevant information is passed to the network. Then, it processes the prior information with the current information to selectively update the cell state, and at last, an output gate is placed in the network to return a transformed version of the cell state [2].

A. Implementation Details

This section describes the implementation of the prototype method for the detection of DGA-based malicious domains.

Feature Engineering. As mentioned, we use two distinct approaches to represent the domain corpus. Using Python, we integrate the Bigram model with *Count Vectorizer* library of

TABLE I

OVERVIEW OF OUR STUDIED DGA DATASETS' ASSESSMENT. THE PERCENTAGE CALCULATIONS ARE COMPUTED FROM RESPECTIVE MALWARE FAMILY'S OBSERVATIONS/RECORDS. THE THIRD COLUMN DENOTES THE PERFORMANCE OF VIRUSTOTAL'S SCAN: ● IS REPORTING THE URLS MOSTLY AS MALICIOUS; ▲ IS REPORTING THE URLS MOSTLY AS SAFE (WHICH IS ALARMING); AND ○ IS FOR LABELING NO SIGNIFICANCE IN SCAN REPORT.

Malware Family	Scan Evaluation	Type of DGA	VirusTotal Scan Report			No. of TLDs (distinct)	Live Domains	Domain String w/o Numeric
			No Records	Safe	Malicious			
Bamital	○	Traditional	99.49%	0.23%	0.28%	3	28%	0%
Banjori	▲	Dictionary	0%	31%	69%	1	0%	100%
Bedep	●	Traditional	0%	2.43%	/ 97.57%	1	0%	51%
Beebone	●	Traditional	0%	1.33%	98.67%	5	1%	0%
Blackhole	●	Traditional	0%	0.36%	99.64%	1	0%	100%
Bobax	●	Traditional	6.92%	0%	93.08%	2	37%	100%
CCleaner	●	Traditional	0%	0%	100%	1	100%	0%
Chinad	○	Traditional	100%	0%	0%	7	0%	0%
Chir	▲	Traditional	0%	94.9%	5.1%	2	0%	0%
Conficker	○	Traditional	98.59%	0.51%	0.9%	5	1%	100%
Corebot	○	Traditional	83.97%	6.29%	9.74%	7	26%	0%
Cryptolocker	○	Traditional	100%	0%	0%	7	0%	100%
Darkshell	○	Traditional	70.83%	12.5%	16.67%	1	15%	0%
Diamondfox	○	Traditional	96.38%	1.93%	1.69%	4	0%	1%
Dircrypt	●	Traditional	0%	17%	83%	1	3%	100%
Dnsbenchmark	○	Traditional	100%	0%	0%	1	0%	0%
Dnschanger	○	Traditional	100%	0%	0%	1	0%	100%
Downloader	▲	Dictionary	0%	92%	8%	1	100%	100%
Dyre	○	Traditional	100%	0%	0%	8	13%	0%
Ebury	○	Traditional	99.85%	0%	0.15%	3	0%	0%
Ekforward	○	Traditional	90.14%	9.86%	0%	1	0%	0%
Emotet	○	Traditional	100%	0%	0%	1	0%	100%
Feodo	●	Traditional	0%	12.22%	87.78%	1	0%	100%
Fobber	●	Traditional	58.62%	1.1%	40.28%	1	0%	100%
Gameover	○	Traditional	99.98%	0%	0.02%	4	0%	0%
Gameover P2P	○	Traditional	100%	0%	0%	6	0%	100%
Gozi	○	Dictionary	99.85%	0.1%	0.05%	1	0%	100%
Goznym	▲	Traditional	12.67%	67.22%	20.11%	1	4%	100%
Gspy	▲	Traditional	7.32%	92.68%	0%	1	0%	0%
Hesperbot	●	Traditional	0%	0.67%	99.33%	1	0%	100%
Infy	○	Traditional	100%	0%	0%	3	0%	0%
Locky	○	Traditional	99.43%	0.37%	0.2%	14	0%	100%
Madmax	●	Traditional	23.97%	4.17%	71.86%	4	50%	17%
Matsnu	▲	Dictionary	65.18%	26.52%	8.3%	1	16%	100%
Mirai	●	Traditional	40.6%	11.8%	47.6%	3	17%	100%
Modpack	●	Traditional	0%	13%	87%	1	51%	6%
Murofet	○	Traditional	100%	0%	0%	5	0%	100%
Murofetweekly	○	Traditional	100%	0%	0%	6	0%	0%
Necurs	○	Dictionary	99.89%	0.02%	0.09%	44	2%	100%
Nymaim	▲	Traditional	55.5%	19%	25.5%	6	10%	100%
Oderoor	●	Traditional	12.17%	7.83%	80%	4	2%	100%
Omexo	▲	Traditional	5%	95%	0%	1	0%	0%
Padcrypt	●	Traditional	56.56%	6.64%	36.8%	11	0%	100%
Pandabanker	○	Traditional	85.14%	0.33%	14.53%	2	4%	0%
Proslikefan	▲	Traditional	100%	0%	0%	3	0%	100%
Pushdo	○	Traditional	99.92%	0%	0.08%	2	0%	100%
Pushdotid	○	Traditional	77.34%	18.44%	4.22%	5	0%	100%
Pykspa	○	Traditional	79.68%	13.41%	6.91%	4	0%	100%
Pykspa 2	●	Traditional	51.25%	1.92%	46.83%	6	5%	100%
Pykspa 2S	●	Traditional	0.01%	0.08%	99.91%	6	0%	100%
Qadars	○	Traditional	100%	0%	0%	3	0%	2%
Qakbot	○	Traditional	100%	0%	0%	5	0%	100%
Ramdo	●	Traditional	0%	2.09%	97.91%	1	5%	100%
Ramnit	●	Traditional	0%	21.67%	78.33%	1	1%	100%
Ranbys	○	Traditional	100%	0%	0%	8	0%	100%
Randomloader	○	Traditional	100%	0%	0%	4	25%	100%
Redyms	▲	Traditional	0%	82%	18%	1	0%	100%
Rovnix	○	Traditional	99.17%	0.5%	0.33%	5	0%	1%
Shifu	●	Traditional	0%	0.95%	99.05%	2	0%	100%
Simda	▲	Traditional	27.5%	67.4%	5.1%	2	0%	100%
Sisron	○	Traditional	100%	0%	0%	4	0%	100%
Sphinx	○	Traditional	98.35%	0%	1.65%	1	0%	100%
Suppobox	●	Dictionary	11.73%	6.83%	81.44%	2	7%	100%
Sutra	▲	Traditional	90%	2.15%	7.85%	1	0%	100%
Symmi	●	Traditional	40.34%	2.08%	57.58%	1	0%	100%
Szribi	○	Traditional	99.67%	0.8%	0.25%	1	0%	100%
Tempedreve	▲	Traditional	0%	37.5%	62.5%	4	1%	100%
Tempedrevetdd	▲	Traditional	5.3%	82.2%	12.5%	4	1%	100%
Tinba	▲	Traditional	5.71%	32.08%	62.21%	12	44%	100%
Tofsee	▲	Traditional	65.35%	27.95%	6.7%	2	0%	100%
Torpig	○	Traditional	99.84%	0%	0.14%	3	0%	84%
Tsifiri	●	Traditional	0%	0%	100%	1	18%	0%
Ud2	▲	Traditional	63.25%	36.75%	0%	1	0%	0%
Ud3	▲	Traditional	61.66%	36.67%	1.67%	5	2%	0%
Ud4	▲	Traditional	17%	53%	30%	5	20%	100%
Urizone	●	Traditional	0%	0.2%	99.8%	2	1%	8%
Vawtrak	●	Traditional	0%	18.25%	81.75%	3	4%	100%
Vidro	○	Traditional	99.59%	1.41%	0%	3	0%	100%
Vidrotid	○	Traditional	77.33%	1%	21.67%	3	0%	100%
Virut	○	Traditional	92.63%	5%	2.37%	1	2%	100%
Violatilecedar	▲	Dictionary	0%	26.5%	73.5%	2	0%	100%
Wd	○	Traditional	100%	0%	0%	2	0%	0%
Xshellghost	○	Traditional	100%	0%	0%	1	0%	100%
Xxhex	▲	Traditional	5.95%	90.2%	3.85%	44	7%	0%

Scikit-Learn to convert a corpus to a matrix of token counts and the Word2Vec model with the Gensim library [7].

Classification Models. While the vector representation of the Bigram model is fed into several machine learning classifiers, such as Logistic Regression, Decision Tree, and Artificial Neural Network (ANN), the Word2Vec model is assigned to an LSTM network. We build and train Logistic Regression and Decision Tree Classifiers with a stratified K-Folds Cross-Validator to avoid overfitting of the models. This removes the necessity of a validation set. The validator returns folds by preserving the ratio of samples for each class in the dataset. We specify 5-fold splits to build both models.

We split the entire dataset for ANN and LSTM models into 70% training, 20% testing, and 10% validation instances. Additionally, we perform stratified splits on training and testing instances to preserve the same percentage for each target class as in the complete set provided in the dataset. The ANN network consists of one input layer, two hidden layers with an activation function of Rectified Linear Units (ReLU), and one output layer with a Sigmoid activation function for the binary classification and SoftMax for multiclass classification. The LSTM model for both the binary and multiclass classification have very similar composition, with the primary changes to be found in the activation and loss functions for each type of the classifiers with only minor hyperparameter variation. Including both the Word2Vec embedding input and the remaining features require two input layers. The Word2Vec input layer is fed directly into the LSTM layer, which is constructed using both a dropout and recurrent dropout rate of 0.9 and ReLU as the activation function. Next, the LSTM layer is concatenated with the remaining features via another dense input layer. The concatenated data is then fed into another dropout layer with a rate of 0.9. Then, the data is fed to another dense (fully-connected) layer, comprised of 64 units and employing ReLU activation, to increase model complexity. Finally, the data is fed to the output layer, which employs a Sigmoid and SoftMax activation function for binary and multiclass classification respectively.

When compiling both ANN and LSTM, we utilize Binary Cross Entropy and Sparse Categorical Cross Entropy for binary and multiclass classification respectively as our loss function to measure the cost incurred from incorrect predictions. To achieve the lowest loss in the network weights, we apply Adam for ANN and Stochastic Gradient Descent optimizer for LSTM as our adaptive learning rate algorithms. To improve the generalization of our model on unseen data, we incorporate the Early Stopping regularization technique for both ANN and LSTM. In the spirit of open science, our implementation, along with the datasets, has been made open source with the MIT License and is available online[1].

Model Evaluation. We consider different metrics for evaluation, *e.g.*, Accuracy, Precision (measures the ratio of true positive instances to all positively labeled instances), Recall (derives the ratio of true positive instances to all instances that should have been labeled positive), F_1 (computes the harmonic mean of the precision and recall scores).

[1]https://github.com/TnTech-CEROC/malicious_domains_dga_detection

IV. Results

We present our experimental results in the following two sections: binary classification and multiclass classification – predicting which family the domains belong to (benign or anyone out of the 84 DGA families).

A. Binary Classification

The overall performance of the models is presented in Table II. We achieve satisfactory results for both feature extraction methods. However, ANN with the Bigram model performs the best compared with the other two models in terms of accuracy, precision score, and F_1 score in particular.

TABLE II
PERFORMANCE OVERVIEW OF BINARY CLASSIFICATION TASK AMONG DIFFERENT BUILT MACHINE LEARNING MODELS.

Model	Accuracy	Precision	Recall	F_1
Logistic Regression	0.9816	0.9911	0.9993	0.9833
Decision Tree	0.9965	0.9965	0.9988	0.9941
ANN	**0.9979**	**0.9979**	**0.9979**	**0.998**
LSTM	0.995	0.9949	0.9958	0.994

During our inspection of monitoring the models' performance over the accuracy over epoch and loss over epoch curves for both ANN and LSTM, we observe that 10 and 6 epochs were used respectively to plot accuracy over epoch and loss over epoch curves for ANN while 20 epochs we found to be used for plotting both curves for LSTM.

B. Multiclass Classification

Next, we analyze how well the models perform relating a domain to a particular malicious family, that is, multiclass classification. This strengthens our study as we aim not only to classify the domains accurately but also to further investigate which families the models fail to detect and why. We summarize our findings for multiclass classification results in Table III that presents an overview of all the built models' performances for each of the studied malware family in terms of Precision, Recall, and F_1 scores. We highlight 15 out of 84 malware families where the models were unable to achieve good classification results having precision, recall, and F_1 scores of less than 61% for all of the built machine learning models. Among these 15 families, none of the models were able to detect malicious domains from the two DGA-families, *Dnschanger* and *Randomloader*. Additionally, the table shows the micro average, macro average, and weighted average values of our findings. Similar to binary classification, we achieve much better results for ANN compared with the other two classifiers that use the Bigram model for feature extraction. The LSTM with the Word2Vec model has achieved satisfactory performance similar to the other classifiers.

TABLE III

PERFORMANCE OVERVIEW OF MULTICLASS CLASSIFICATION TASK AMONG DIFFERENT BUILT MACHINE LEARNING MODELS.

Malware Family	Total Records	Logistic Regression			Decision Tree			Artificial Neural Network			Long Short-Term Memory		
		Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
Majestic (Benign)	125,000	0.9929	0.992	0.996	0.9972	0.9986	0.9979	0.997	0.993	0.9981	0.9968	0.9998	0.9983
Bamital	1,780	0.9726	0.9972	0.9847	1.0	1.0	1.0	1.0	1.0	1.0	0.9621	0.9972	0.9793
Banjori	1,130	1.0	1.0	1.0	0.9956	1.0	0.9978	0.9956	1.0	0.9978	0.7092	0.9602	0.8158
Bedep	**1,030**	**0.6404**	**0.5561**	**0.5963**	**0.5806**	**0.6117**	**0.5957**	**0.5056**	**0.6602**	**0.5726**	**0.4167**	**0.801**	**0.5482**
Beebone	150	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9	0.9474
Blackhole	1,130	0.6098	1.0	0.7516	0.995	0.8894	0.9393	0.9375	0.9292	0.9333	0.8218	1.0	0.9022
Bobax	260	1.0	0.9615	0.9804	0.7903	0.9423	0.8596	0.9423	0.9423	0.9423	0.8644	0.9808	0.9189
CCleaner	12	1.0	0.5	0.6667	1.0	1.0	1.0	1.0	1.0	1.0	0	0	0
Chinad	1,130	0.6481	0.6222	0.6349	0.9912	0.9912	0.9912	0.8989	0.708	0.7921	0.8856	0.9248	0.9048
Chir	100	0.8	0.4211	0.5517	0.7368	0.7	0.7179	0.72	0.9474	0.8182	1.0	0.4211	0.5926
Conficker	**3,130**	**0.4454**	**0.4211**	**0.5517**	**0.4956**	**0.5399**	**0.5168**	**0.4644**	**0.7508**	**0.5739**	**0.48**	**0.6342**	**0.5465**
Corebot	1,130	0.9852	0.8889	0.9346	0.9777	0.969	0.9733	1.0	0.8982	0.9464	0.9883	0.7478	0.8514
Cryptolocker	4,130	0.5749	0.6279	0.6002	0.6811	0.7058	0.6932	0.7688	0.7688	0.7688	0.6687	0.6501	0.6593
Darkshell	50	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5556	0.7143
Diamondfox	850	1.0	0.9515	0.9752	1.0	1.0	1.0	1.0	1.0	1.0	0.9643	0.9759	0.9701
Dircrypt	**500**	**0.1852**	**0.1**	**0.1299**	**0.2911**	**0.23**	**0.257**	**0.2719**	**0.31**	**0.2897**	**0**	**0**	**0**
Dnsbenchmark	**4**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Dnschanger	100	0.3353	0.6816	0.4495	0.5107	0.6789	0.5829	0.5497	0.3821	0.4508	0.5646	0.9593	0.7108
Downloader	1,200	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Dyre	650	0.9958	1.0	0.9979	1.0	1.0	1.0	0.9833	0.9916	0.9874	1.0	0.991	0.9958
Ebury	570	0.8978	0.9462	0.9213	1.0	1.0	1.0	1.0	0.9615	0.9804	1.0	0.7538	0.8589
Ekforward	1,400	1.0	1.0	1.0	1.0	1.0	1.0	0.9127	1.0	0.9544	1.0	0.7478	0.8557
Emotet	1,080	0.8842	0.9821	0.9306	0.9291	0.9821	0.9549	0.9329	0.9929	0.9619	0.9302	1.0	0.9639
Feodo	725	1.0	0.8056	0.8923	0.9474	1.0	0.973	0.9615	0.6944	0.8065	0.8966	0.7222	0.8
Fobber	**180**	**0.082**	**0.069**	**0.0749**	**0.2941**	**0.2414**	**0.2652**	**0.2456**	**0.5793**	**0.345**	**0.3817**	**0.3448**	**0.3623**
Gameover	8,500	0.9657	0.9953	0.9803	0.9988	0.9965	0.9976	0.9988	0.9982	0.9985	0.9918	0.9988	0.9953
Gameover P2P	1,950	0.9421	0.9689	0.9553	0.8355	0.814	0.8246	0.9074	0.9871	0.9455	0.9399	0.6873	0.794
Gozi	2,000	0.9603	0.9699	0.9651	0.9066	0.9225	0.9145	0.8982	0.9925	0.943	0.9557	0.755	0.8436
Goznym	**360**	**0.1786**	**0.0694**	**0.1**	**0.1831**	**0.3562**	**0.2419**	**0.3077**	**0.2778**	**0.292**	**0.2564**	**0.1389**	**0.1809**
Gspy	**40**	**0.3333**	**0.125**	**0.1818**	**0.3333**	**0.375**	**0.3529**	**1.0**	**0.125**	**0.2222**	**0.4286**	**0.75**	**0.5455**
Hesperbot	**150**	**0**	**0**	**0**	**0.0588**	**0.0333**	**0.0426**	**0.5**	**0.0333**	**0.0625**	**0.3125**	**0.1667**	**0.2174**
Infy	1,050	0.9859	1.0	0.9929	1.0	1.0	1.0	0.9589	1.0	0.979	0.972	0.9905	0.9811
Locky	3,550	0.8578	0.8251	0.8411	0.8565	0.7648	0.808	0.895	0.9	0.8975	0.9193	0.8183	0.8659
Madmax	300	0.9333	0.4912	0.6437	0.8462	0.7586	0.8	0.8772	0.8772	0.8772	0.8727	0.8421	0.8571
Matsnu	4,050	0.9154	0.9346	0.9249	0.9281	0.8444	0.8843	0.9462	0.9778	0.9617	0.7465	0.8654	0.8016
Mirai	500	0.9804	1.0	0.9901	0.95	0.95	0.95	1.0	0.98	0.9899	1.0	0.99	0.995
Modpack	200	1.0	0.975	0.9873	1.0	1.0	1.0	0.8837	0.95	0.9157	0.973	0.9	0.9351
Murofet	1,200	0.5121	0.4417	0.4743	0.4686	0.4667	0.4676	0.5335	0.7958	0.6388	0.4154	0.4708	0.4414
Murofetweekly	1,200	1.0	0.9958	0.9979	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.975	0.9873
Necurs	4,200	0.9309	0.6893	0.7921	0.7819	0.7726	0.7772	0.9209	0.7619	0.8339	0.9585	0.769	0.8534
Nymaim	**1,200**	**0.3053**	**0.1208**	**0.1731**	**0.3081**	**0.2573**	**0.2573**	**0.6667**	**0.1833**	**0.2876**	**0.4587**	**0.2083**	**0.2865**
Oderoor	1,200	0.7094	0.6	0.9501	0.6273	0.575	0.6	0.6976	0.5958	0.6427	0.4956	0.4708	0.4829
Omexo	20	1.0	1.0	1.0	1.0	1.0	1.0	0	0	0	0	0	0
Padcrypt	1,220	0.9959	1.0	0.998	0.8294	0.7172	0.7692	0.9959	0.9959	0.9959	0.8692	0.4631	0.6043
Pandabanker	1,200	0.952	0.9958	0.9734	0.9877	1.0	0.9938	1.0	0.9208	0.9588	0.917	0.9208	0.9189
Proslikefan	1,100	0.8031	0.9498	0.8703	0.8222	0.8409	0.8315	0.8684	0.9	0.8839	0.8	0.9455	0.8667
Pushdo	1,200	0.9312	0.9583	0.9446	0.9202	0.9125	0.9163	0.9565	0.9167	0.9362	0.8732	0.775	0.8212
Pushdotid	900	0.7833	0.8833	0.8303	0.5099	0.5722	0.5383	0.8111	0.8111	0.8111	0.5441	0.4111	0.4684
Pykspa	**1,200**	**0.227**	**0.1333**	**0.168**	**0.2151**	**0.1667**	**0.1878**	**0.3248**	**0.2125**	**0.2569**	**0.2661**	**0.1375**	**0.1813**
Pykspa 2	1,200	0.7547	0.6667	0.708	0.8531	0.75	0.7982	0.819	0.7167	0.7644	0.5424	0.1333	0.214
Pykspa 2S	1,200	0.7681	0.8417	0.8032	0.8014	0.9414	0.8659	0.7959	0.8125	0.8041	0.4888	0.725	0.5839
Qadars	2,000	0.9901	0.9975	0.9938	0.9824	0.9775	0.9799	1.0	0.99	0.995	0.879	0.9625	0.9189
Qakbot	4,000	0.5585	0.5188	0.5379	0.4838	0.4722	0.7412	0.58	0.58	0.6508	0.636	0.4587	0.533
Ramdo	1,200	1.0	0.9874	0.9937	0.9959	1.0	0.9979	0.9959	1.0	0.9979	0.9677	1.0	0.9836
Ramnit	1,200	0.4621	0.5333	0.4952	0.3668	0.3958	0.3808	0.454	0.3292	0.3816	0.2727	0.1375	0.1828
Ranbys	2,000	0.7704	0.755	0.7626	0.8916	0.925	0.908	0.8559	0.7575	0.8037	0.8443	0.8675	0.8557
Randomloader	**4**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Redyms	30	0.8571	1.0	0.9231	0.8571	1.0	0.9231	1.0	0.5	0.6667	0.8333	0.8333	0.8333
Rovnix	1,200	0.6713	0.6042	0.636	0.9832	0.975	0.9791	0.742	0.9708	0.8412	0.8821	0.9667	0.9225
Shifu	2,000	0.9297	0.9925	0.9601	0.9684	0.9975	0.9828	0.9777	0.9875	0.9826	0.9752	0.9825	0.9788
Simda	1,200	0.9363	0.9925	0.9636	0.9547	0.9475	0.9511	0.9707	0.9925	0.9815	0.8687	0.9925	0.9265
Sisron	1,200	1.0	1.0	1.0	0.995	1.0	0.9975	1.0	1.0	1.0	0.6711	1.0	0.8032
Sphinx	1,200	0.6947	0.8875	0.7794	0.7859	0.9175	0.8466	0.8063	0.895	0.8483	0.7671	0.815	0.7903
Suppobox	1,200	0.9895	0.9796	0.9845	0.9749	0.9725	0.9937	0.9937	0.9838	0.9887	0.9161	0.9875	0.9505
Sutra	2,000	0.9785	0.6825	0.8041	0.9409	0.995	0.9672	0.8909	1.0	0.9423	0.9184	0.8725	0.8949
Symmi	1,200	1.0	0.9958	0.9979	1.0	0.9875	0.9937	0.9959	1.0	0.9979	0.9958	0.9958	0.9958
Szribi	1,200	0.8872	0.9833	0.9328	0.9228	0.9958	0.9579	0.9449	1.0	0.9717	0.7742	1.0	0.8727
Tempedreve	**200**	**0.1875**	**0.075**	**0.1071**	**0.1351**	**0.125**	**0.1299**	**0.2105**	**0.1**	**0.1356**	**0**	**0**	**0**
Tempedrevetdd	1,000	0.5323	0.33	0.4074	0.7227	0.86	0.7854	0.75	0.72	0.7347	0.8049	0.825	0.8148
Tinba	2,000	0.9007	0.9323	0.9163	0.9011	0.82	0.8586	0.9619	0.8825	0.9205	0.9293	0.8875	0.9079
Tofsee	2,000	0.9859	0.875	0.9272	0.9975	1.0	0.9988	1.0	0.9975	0.9987	0.9195	1.0	0.9581
Torpig	1,200	0.9286	0.3792	0.5385	0.8968	0.9417	0.9187	0.9835	0.9958	0.9896	0.681	0.9875	0.8061
Tsifiri	50	1.0	1.0	1.0	1.0	1.0	1.0	0	0	0	0	0	0
Ud2	2,000	0.9824	0.975	0.9787	1.0	1.0	1.0	1.0	1.0	1.0	0.9926	1.0	0.9963
Ud3	60	1.0	0.6667	0.8	0.9167	0.9167	0.9167	1.0	1.0	1.0	1.0	0.5833	0.7368
Ud4	100	0.9048	0.95	0.9268	0.9167	0.55	0.6875	0.9333	0.7	0.8	1.0	0.4	0.5714
Urizone	2,000	0.8762	0.885	0.8806	0.9206	0.87	0.8946	0.8701	0.8875	0.8787	0.9008	0.885	0.8928
Vawtrak	2,000	0.8017	0.94	0.8654	0.7108	0.59	0.6448	0.866	0.905	0.8851	0.6938	0.64	0.6658
Vidro	**1,275**	**0.4591**	**0.3961**	**0.4253**	**0.4304**	**0.3882**	**0.4082**	**0.6909**	**0.4471**	**0.5429**	**0.5164**	**0.4314**	**0.4701**
Vidrotid	**400**	**0.5**	**0.2833**	**0.3617**	**0.1136**	**0.0833**	**0.0962**	**0.9**	**0.15**	**0.2571**	**0**	**0**	**0**
Virut	3,200	0.8624	0.9413	0.9001	0.9326	0.9686	0.9502	0.9196	0.9471	0.9331	0.9359	0.98	0.9574
Violatilecedar	500	1.0	1.0	1.0	1.0	1.0	1.0	0.99	1.0	0.995	0.9515	0.9899	0.9703
Wd	1,200	1.0	1.0	1.0	1.0	1.0	1.0	0.995	1.0	0.9975	1.0	1.0	1.0
Xshellghost	**35**	**0**	**0**	**0**	**0**	**0**	**0**	**1.0**	**0.1429**	**0.25**	**0**	**0**	**0**
Xxhex	2,000	1.0	1.0	1.0	0.995	1.0	0.9975	0.9975	0.9875	0.9925	0.9258	0.9975	0.9603
Micro Avg.		0.9147	0.9147	0.9147	0.9211	0.9211	0.9211	0.9358	0.9358	0.9358	0.9104	0.9104	0.9104
Macro Avg.		0.7882	0.7574	0.7629	0.7754	0.7755	0.7734	0.8128	0.7657	0.7692	0.7174	0.6924	0.6914
Weighted Avg.		0.914	0.9147	0.9116	0.921	0.9211	0.9205	0.9388	0.9358	0.9343	0.9069	0.9104	0.9045

V. Limitations and Future Work

As per the discussions in the Results section, the multiclass models achieved poor performance for prediction of malicious domains belonging to the following 15 families (all traditional DGAs) out of 84: *Bedep*, *Conficker*, *Dircrypt*, *Dnsbenchmark*, *Fobber*, *Goznym*, *Gspy*, *Herperbot Nymaim*, *Pykspa*, *Randomloader*, *Tempredreve*, *Vidro*, *Vidrotid*, and *Xxshellghosht*. In search of probable reasons behind this result, we observe that only 5 out of 15 generated an adequate number of domains for the models to effectively train, while the rest had a very limited number of domains (less than or equal to 300) present in our dataset. We also found that there was significant randomness in the malicious domain examples for the following 5 malware families: *Bedep*, *Conficker*, *Fobber*, *Nymaim*, *Pykspa*, and *Vidro*, which might have caused the inaccurate multiclass classification. More examples of these families in training may lead to finding more unique and decisive patterns. Therefore, we concluded that the lack of a representative training dataset is the primary reason behind the classifiers failing to produce better results for this particular dataset we use in our research.

VI. Related Work

DGA based domain detection has been heavily studied in the past. Yadav *et al.* [12] explored methods for detecting DGA domains in their research, focusing on traditional DGA domains. For feature engineering, they have employed K-L Divergence with Unigram distribution, Jaccard measure of Bigrams, and Edit-Distance. For the classification of domains queried on a network as malicious or benign, researchers have used LASSO or L1-Regularized Regression as a supervised machine learning technique. The models' results were compared against external sources such as McAfee Site Advisor to confirm if those detected as malicious were accurate. The use of external resources to confirm a site's malicious identity is also demonstrated in our research as we compare domains in our dataset against VirusTotal's database [10].

Deep Learning networks have been used for the successful detection of DGAs in previous research, especially in the case of Dictionary-based DGAs. Woodbridge *et al.* [11] applied LSTM networks to the problem of real-time detection of DGA domains. An advantage of their LSTM network was being able to create the model without any manual feature classification. They also classified which domains belonged to specific malware families. They achieved a 90% detection rate with good Precision. Pereira *et al.* [8] focused on Dictionary-based DGA detection. They also created dictionaries for each malware family by extracting words used in domains generated by various malware families with their WordGraph method. The WordGraph method consisted of utilizing feature engineering to create a graph of the words in the domain names, and cluster these together into different groups. The more tightly knit groups represented different DGA families.

VII. Conclusion

In this research, we develop a suite of effective detection mechanisms for both traditional and dictionary-based DGAs. We analyze 84 different malware families' dataset with the Virus-Total API service to verify the status of each malicious domain through multiple Antivirus scan engines. We utilize the DNS queries to determine whether or not a domain is a non-existent domain (NXDomain) and hence, short-lived. We extract useful features to further analyze the domain strings. The Bigram model was used with several Machine Learning techniques, such as Logistic Regression, Decision Tree, and Artificial Neural Network (ANN) while the Word2Vec model was used with Long Short-Term Memory (LSTM). We observe that both binary and multiclass classification tasks with the built models demonstrate satisfactory detection results. When compared to the other two Machine Learning classifiers, the ANN method with the Bigram model performed the best with an accuracy of 99.8% in detecting benign and malicious domains and 93.58% in classifying domains belonging to specific malware families. The performance of the LSTM is almost identical with an accuracy of 99.5% in binary classification and 91.04% in multiclass classification.

Acknowledgement

The entire work has been supported by the Cybersecurity Education, Research & Outreach Center (CEROC) and the College of Engineering (CoE) at Tennessee Tech University.

References

[1] 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. ACM, 2016.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[4] Marc Kührer, Christian Rossow, and Thorsten Holz. Paint it black: Evaluating the effectiveness of malware blacklists. In *International Workshop on Recent Advances in Intrusion Detection*, pages 1–21. Springer, 2014.

[5] Majestic Million Majestic, 2019. URL: https://majestic.com/reports/majestic-million/.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Mayana Pereira, Shaun Coleman, Bin Yu, Martine DeCock, 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.

[9] 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 ({USENIX} Security 16)*, pages 263–278, 2016.

[10] VirusTotal. Public api v2.0, 2019. URL: https://www.virustotal.com/en/documentation/public-api/.

[11] Jonathan Woodbridge, Hyrum S Anderson, Anjum Ahuja, and Daniel Grant. Predicting domain generation algorithms with long short-term memory networks. *arXiv preprint arXiv:1611.00791*, 2016.

[12] Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Reddy, and Supranamaya Ranjan. Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 48–61. ACM, 2010.