DGA Botnet Detection Using Supervised Learning Methods

Hieu Mac
Bach Khoa Cybersecurity Centre
HUST
Vietnam
hieumd@soict.hust.edu.vn

Duc Tran
Bach Khoa Cybersecurity Centre
HUST
Vietnam
ductq@soict.hust.edu.vn

Van Tong
Bach Khoa Cybersecurity Centre
HUST
Vietnam
tongvanvan94@gmail.com

Linh Giang Nguyen
Bach Khoa Cybersecurity Centre
HUST
Vietnam
giangnl@soict.hust.edu.vn

Hai Anh Tran
Bach Khoa Cybersecurity Centre
HUST
Vietnam
anhth@soict.hust.edu.vn

ABSTRACT

Modern botnets are based on Domain Generation Algorithms (DGAs) to build a resilient communication between bots and Command and Control (C&C) server. The basic aim is to avoid blacklisting and evade the Intrusion Protection Systems (IPS). Given the prevalence of this mechanism, numerous solutions have been developed in the literature. In particular, supervised learning has received an increased interest as it is able to operate on the raw domains and is amenable to real-time applications. Hidden Markov Model, C4.5 decision tree, Extreme Learning Machine, Long Short-Term Memory networks have become the state of the art in DGA botnet detection. There also exist several advanced supervised learning methods, namely Support Vector Machine (SVM), Recurrent SVM, CNN+LSTM and Bidirectional LSTM, which have not been suitably appropriated in such domain. This paper presents a first attempt to thoroughly investigate all the above methods, evaluate them on the real-world collected DGA dataset involving 38 classes with 168,900 samples, and should provide a valuable reference point for future research in this field.

CCS CONCEPTS

• **Security and privacy → Intrusion/anomaly detection and malware mitigation → Malware and its mitigation**

KEYWORDS

DGA Botnet, Supervised Learning, Long Short-Term Memory networks, Recurrent SVM, Bidirectional LSTM

SoICT '17, December 7–8, 2017, Nha Trang City, Viet Nam

ACM ISBN 978-1-4503-5328-1/17/12...$15.00
https://doi.org/10.1145/3155133.3155166

ACM Reference format:

H. Mac, D. Tran, V. Tong, G. Nguyen, A. Tran. 2017. DGA Botnet Detection Using Supervised Learning Methods. In *SoICT '17: Eighth International Symposium on Information and Communication Technology, December 7–8, 2017, Nha Trang City, Viet Nam.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3155133.3155166

1 INTRODUCTION

Botnets have become a technological backbone to support cyber-criminals such as launching distributed denial of service attacks, stealing personal data and sending spam mail [2]. Most bots today are based on Domain Generation Algorithms (DGAs) to create a rendezvous point with their Command and Control (C&C) server [4]. A typical DGA consists of various seeds, which utilize static integer, current date and time to generate a list of candidate domains. The domain list is changed over the time, making it difficult for the law enforcement agencies to detect and shut down botnets. Traditional solutions include blacklisting and reverse engineering [3]. Blacklisting is however not sufficient to give protection against domain fluxing [1] since it becomes extremely challenging to determine and blacklist all the malicious domains. Reverse engineering, on the other hand, is time-consuming and requires a malware sample, which is not always possible in practical applications [1].

Machine learning has recently attracted considerable attention in security community. It also provides a mean to combat DGA and find the related malware structure. The machine learning methods can be either unsupervised or supervised. Unsupervised learning groups domains into clusters in order to take advantage of the statistical attributes for each group [8]. Krishnan et al. [16] observed that such approach is time-consuming; it needs several hours to create the domain clusters that are able to produce good generalization capabilities. In some extreme cases, the statistical attributes cannot be extracted due to the limited availability of bots, especially bots that are associated with the same DGA in the enterprise networks [5].

Supervised learning does not rely on statistical attributes to uncover DGAs. It operates directly on the raw domains and their linguistic attributes. Bilge et al. [6] developed EXPOSURE, where

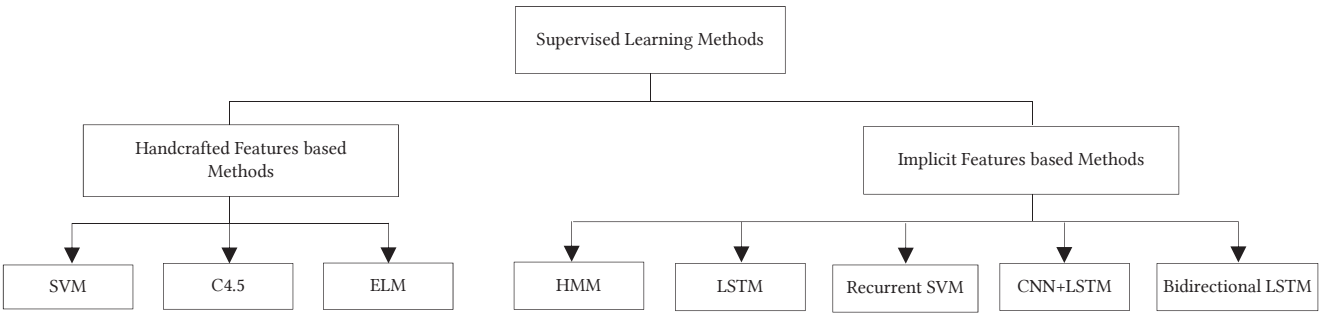

Figure 1: Taxonomy for the supervised learning methods in DGA botnet detection.

the C4.5 decision tree (C4.5) is constructed using the attributes extracted from DNS traffic. Shi et al. [7] utilized Extreme Learning Machine (ELM) to discriminate benign from malicious domains. Antonakakis et al. [2] trained one distinct Hidden Markov Model (HMM) per DGA. HMM receives in input a domain and classifies whether it is automatically generated. Woodbridge et al. [8] leveraged the so-called Long Short-Term Memory network (LSTM) that produces a 90% detection rate with a 1:10000 False Positive (FP) rate. C4.5, ELM, HMM and LSTM seem to be adequate for DGA detection in concrete frameworks; but no attempt has been made to evaluate them on a reasonably large dataset. There also exist several advanced supervised learning methods such as Support Vector Machine (SVM), Recurrent SVM [9], [10], CNN+LSTM [11], and Bidirectional LSTM [12], which have not been validated in this application domain.

No single method can be the best performer for all problems. Hence, this paper aims at providing a thorough investigation on the different supervised learning methods with the aim to determine the appropriate one for DGA detection. This should be a useful source for practitioners and a valuable reference point for future research in this field. This paper is structured as follows. Section 2 presents an overview on the supervised learning methods that can be applied in recognizing DGAs. The evaluation measures and comparative results are discussed in detail in Section 3. Section 4 is dedicated to conclusion and future works.

2 SUPERVISED LEARNING METHODS

Suppervised learning [32] attempts to discover the relationship between input and its corresponding output. In general, the relationship is represented in a structure, also known as a model that can be used to predict the outputs for some future inputs. In this section, we present a comprehensive overview on the supervised learning methods for DGA botnet detection. These methods can be grouped into two categories: handcrafted features based and implicit features based methods (Fig. 1).

2.1 Handcrafted Features Based Methods

Constructing set of properties that helps describe each DGA class is essential in any conventional supervised learning methods. This is due to fact that different models have different

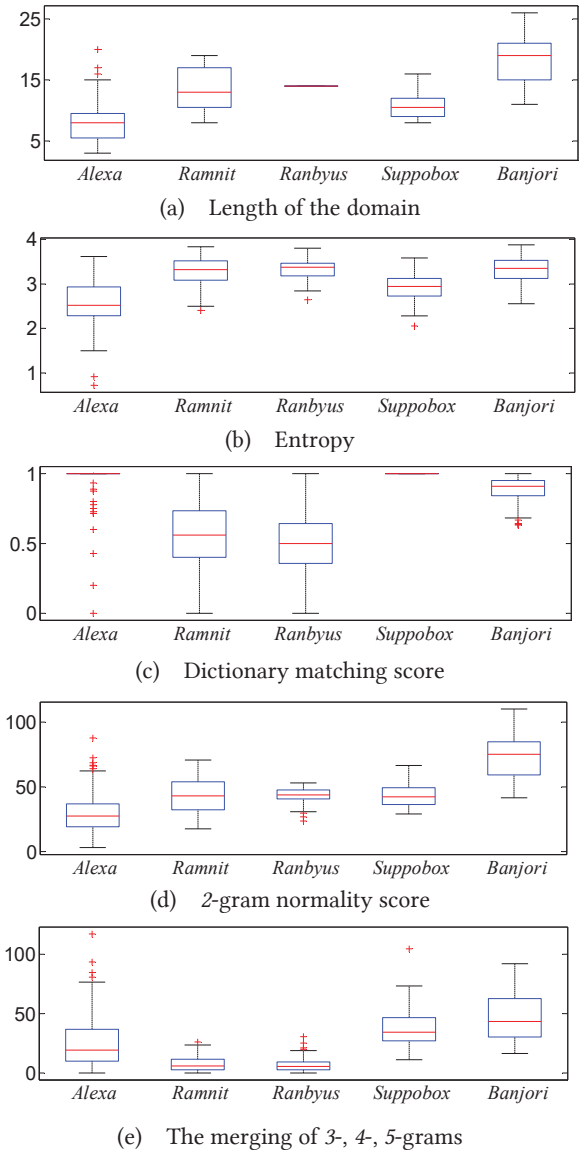

Figure 2: Boxplots showing the distinction between the linguistic attributes related to the non-DGA (Alexa) and 4 DGA classes.

restriction on the type of data that is fed into them [18]. Let d be a domain name, which contains a string of characters, numbers and maybe dash. We also assume that t_n is a n −gram and $|p_n|$ is the total number of n −grams in d. Based on the previous works [3-8], [22], we have selected 9 linguistic attributes, which can characterize a given domain name. These include domain length $|d|$, entropy, dictionary matching score, *n*-gram normality scores ($n = 1,2,3,4,5$) and the merging of *3*-, *4*-, and *5*-grams.

Entropy attribute indicates the average uncertainty of a single random variable. In DGA botnet detection, it is the information that is produced on the average for each 1-gram of d

$$E(d) = -\sum_{t_1 \epsilon p_1} \frac{count(t_1)}{|p_1|} * \log(\frac{count(t_1)}{|p_1|}) \quad (1)$$

Dictionary matching score measures the degree that a string in d can be explained by a dictionary [4]. Let us assume that d is divided into n words w_i. The matching score is subsequently given as

$$R(d) = \frac{\sum_{i=1}^{n} |w_i|}{|d|} \quad (2)$$

The *n*-gram normality scores are built on the premise that there are frequent character patterns in natural language [15]. These scores can be computed using the number of t_n in d (abbreviated as $count(t_n)$) and the frequency of t_n in the *Alexa* top 100,000 domains (abbreviated as $f(t_n)$)

$$S(d) = \frac{\sum_{t_n \epsilon p_n} count(t_n) * f(t_n)}{|d| - n + 1} \quad (3)$$

Fig.2 depicts the linguistic attributes related to DGA and non-DGA (*Alexa*) domains. The DGA malwares involve *Ramnit*, *Ranbyus*, *Suppobox* and *Banjori*. As illustrated, DGA domains are longer and exhibit higher level of linguistic randomness with respect to non-DGA domains (see Figs 2(a) and 2(b)). *Alexa* and *Suppobox* have similar dictionary matching score since *Suppobox* is also based on pronounceable domains. It is observed that the entropy and *n*-gram normality scores for *Ramnit* and *Ranbyus* are mostly identical. This is due the fact that these malwares exploit the same generator that concatenates multiplies, divisions and modulo in a single seed.

It is clear that the domain length, entropy and dictionary matching score have sufficient discriminant power to distinguish malicious domains from the legitimate ones. The *n*-gram normality scores however provide additional information to improve the overall accuracy. These attributes are concatenated to form a feature vector, which is then fed into the supervised learning methods, including C4.5 decision tree (C4.5), Support Vector Machine (SVM) and Extreme Learning Machine (ELM).

C4.5 utilizes a greedy top-down procedure, where an attribute is selected as root node and the samples are divided into subsets using the selected attribute [19]. This process is repeated until the stopping criterion is met. The fact that C4.5 is simple and technically easy to use makes it the most attractive option for DGA botnet detection in the literature [6], [30], [31].

SVM has a strong theoretical background and is a powerful tool for data classification. The aim of SVM is to project input samples into a high dimensional space and find the optimal hyperplane in this space to separate the samples [17]. SVM is inherently binary classifier, and can be extended to solve multiclass problems using one-versus-one strategy. The principle of such strategy is to evaluate all possible pairwise classifiers and label an input samples to the class with the most votes. The one-versus-one strategy is considered to be better alternative to the one-versus-rest counterpart because it is substantially faster and is more suitable for the problem, which involves a very large number of DGA classes [18].

ELM provides an effective solution for the single hidden layer feedforward networks (SLFNs), which does not require the hidden layer to be tuned [7]. ELM is extremely fast in training. Unlike other gradient-based learning algorithms, it works for all bounded non-constant piecewise continuous activation functions. ELM has achieved excellent accuracy in numerous applications, such as image segmentation and human action recognition. We refer the readers to [20] for the details related to ELM.

2.2 Implicit Features Based Methods

As mentioned above, C4.5, SVM and ELM rely on the handcrafted features to build the classification model. Using handcrafted features has a potential drawback as they can be easily circumvented by the malware author. Moreover, building a new set of features is time-consuming and may not possible in practice. Antonakakis et al. [2] overcame this problem by introducing the feature-less **Hidden Markov Model** (HMM).

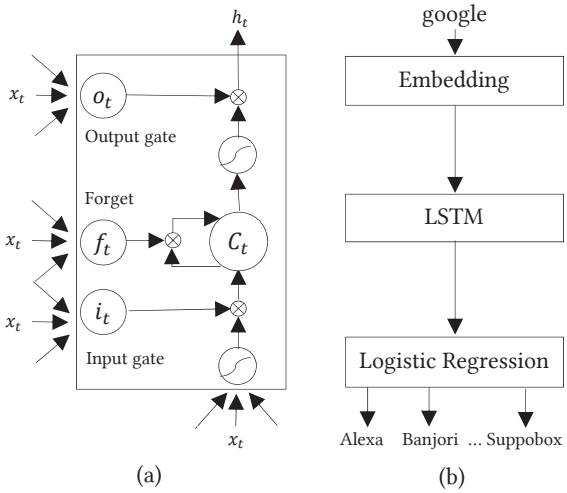

Figure 3: (a) LSTM memory block with only one cell; (b) The LSTM Algorithm.

HMM is able to operate directly on the raw domains. It receives in input a domain and identifies whether the domain is automatically generated. However, as shown in [8] and later in this paper, HMM provides worse detection rate than expected on the dataset, which contains a large number of DGA classes.

Long Short-Term Memory network (LSTM) [13], [14] holds more promise for recognizing DGA malwares since it is capable of modeling temporal sequences and their long-term dependencies [8]. Traditional HMM is limited to discrete state space, while LSTM has Turing capabilities, making it more suitable for all sequence learning tasks. LSTM basic unit is the memory block containing one or more memory cells and three multiplicative gating units (see Fig. 3a). LSTM aims at mapping an input sequence x_t to an output sequence y_t by using the following equations interactively from t =1 to T

$$f_t = \sigma_g(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \quad (4)$$

$$i_t = \sigma_g(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \quad (5)$$

$$o_t = \sigma_g(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \quad (6)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_h(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \quad (7)$$

$$h_t = o_t \odot \sigma_h(c_t) \quad (8)$$

$$y_t = \emptyset(W_{yh}h_t + b_y) \quad (9)$$

where σ_g, σ_h are sigmoid and hyperbolic tangent activation functions. W_{lx} and W_{lx} are the weight matrices with $l \in \{f, i, o, c\}$ denoting the forget gate, input gate, output gate and the cell. $\emptyset$ is the Softmax function that is applied to maximize the log-likelihood and label the input sequence. In this regard, x_t is assigned to class k, which has the highest y_t value.

Woodbridge et al. [8] leveraged LSTM to classify DGA in real-time. LSTM is fast in testing and provides an accuracy of 90% with a 1:10000 FP rate. It is also compact, involving an embedding layer, a LSTM network layer and a fully connected layer. The LSTM network layer consists of 128 blocks and one memory cell per block, while the fully connected layer can be either logistic regression or multinomial logistic regression. Motivated by the pioneering work of LSTM, several LSTM variants have been developed in the literature. **Recurrent SVM** is constructed using the idea of replacing the Softmax with SVM. Softmax tries to minimize the cross-entropy, while the aim of SVM is to find the maximum margin between samples from different classes [9]. In **CNN+LSTM**, the input domain is fed into a single Convolutional Neural Network (CNN) with max-pooling across the sequence for each convolutional feature to find the morphological patterns [11]. The output of CNN is then treated as the input of LSTM to reduce the temporal variations. As a consequence, each CNN and LSTM block captures information about the input representation at different scale [21]. For this reason, CNN+LSTM is expected to be better alternative to the original LSTM.

Bidirectional LSTM is an extension of the traditional LSTM, consists of a forward and backward LSTM. It is observed to achieve higher generalization performance on sequence classification problems [12]. In Bidirectional LSTM, the forward hidden sequence h_t, the backward hidden sequence k_t and output sequence y_t are computed as follows:

$$h_t = \mathcal{F}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (10)$$

$$k_t = \mathcal{F}(W_{xk}x_t + W_{kk}h_{t-1} + b_k) \quad (11)$$

$$y_t = \sigma_g(W_{yh}h_t + W_{yk}k_t + b_y) \quad (12)$$

where $\mathcal{F}$ is an update function, which is implemented by combining Eqs. (4) and (8). The Bidirectional LSTM allows the output units y_t to learn a representation from both the past and future information without having fixed-size window around t [23]. In this paper, it is based on two LSTM layers (forward and backward) with 128 memory blocks in each direction.

Table 1: Summary of the collected dataset

Domain Type	#Sample	Domain Type	#Sample
Geodo	58	*Fobber*	60
Beebone	42	*Alexa*	88347
Murofet	816	*Dyre*	800
Pykspa	1422	*cryptowall*	94
Padcrypt	58	*Corebot*	28
Ramnit	9158	*P*	200
Volatile	50	*Bedep*	172
Ranbyus	1232	*Matsnu*	48
Qakbot	4000	*PT Goz*	6600
Simda	1365	*Necurs*	2398
Ramdo	200	*Pushdo*	168
Suppobox	101	*Cryptolocker*	600
Locky	186	*Dircrypt*	57
Tempedreve	25	*Shifu*	234
Qadars	40	*Bamital*	60
Symmi	64	*Kraken*	508
Banjori	42166	*Nymaim*	600
Tinba	6385	*Shiotob*	1253
Hesperbot	192	*W32.Virut*	60

3 EXPERIMENTS

This section is dedicated to assess the various supervised learning methods, i.e., HMM, C4.5, ELM, SVM, LSTM, Recurrent SVM, CNN+LSTM and Bidirectional LSTM. In particular, we evaluate these methods in both binary (DGA vs. non-DGA) and multiclass (which DGA?) problems. The Wilcoxon signed ranks test is also performed to compare each pair of methods based on their F1-scores. All the codes were written using Keras and scikit-learning libraries [26], [27], and were executed on a PC running Ubuntu 16.04 x64 with Intel Core i5 and 8 GBs of RAM.

3.1 Dataset Specification

The experiments are carried out on a real-world dataset that contains 1 non-DGA (Alexa) and 37 DGA classes. It is collected from two sources: The Alexa top 1 million domains [28] and the OSINT DGA feed from Bambenek Consulting [29]. In total, there are 88,357 legitimate domains and 81,490 DGA domains. The dataset also includes some notable DGA families such as *Cryptolocker*, *Locky*, *Kraken*, *Gameover Zeus*. *Matsnu Cryptowall*, *Suppobox* and *Volatile* are based on domains, which were generated using English dictionary word list. Table 1 illustrates

the number of samples for the non-DGA and DGA classes. Five-fold cross validation (CV) is performed instead of the traditional ten-fold CV as several classes are very small in number in the dataset.

3.2 Evaluation Measures

In this section, we present the measures that are used to evaluate the various methods in our experiments. These include Precision, Recall and F1-score. Assume that $B(TP, TN, FP, FN)$ is a binary evaluation measure, which is computed based on the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). We have that

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

$$F1 - score = \frac{2}{1/Recall + 1/Precision} \tag{15}$$

The average result over all classes is determined using both micro- and macro-averaging. Suppose that λ is a class label, the

Table 2: Performance comparison of HMM, C4.5, ELM and SVM on the multiclass task

	HMM			**C4.5**			**ELM**			**SVM**		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Geodo	0.0127	0.4167	0.0246	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Beebone	0.0308	0.7500	0.0591	0.6250	1.0000	0.7692	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Murofet	0.8235	0.2577	0.3925	0.3810	0.4706	0.4211	0.9301	0.3803	0.5297	0.9785	0.5583	0.7109
Pykspa	0.3090	0.1937	0.2381	0.0000	0.0000	0.0000	0.7972	0.3615	0.4912	0.9625	0.2711	0.4231
Padcrypt	0.2069	1.0000	0.3429	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.1667	0.2857
Ramnit	0.1081	0.0551	0.0730	0.0000	0.0000	0.0000	0.5019	0.7098	0.5878	0.4604	0.7811	0.5794
Volatile	0.0136	0.6000	0.0267	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Ranbyus	0.0424	0.2236	0.0713	0.0000	0.0000	0.0000	0.4492	0.8699	0.5922	0.0000	0.0000	0.0000
Qakbot	0.1240	0.0587	0.0797	0.9773	0.9835	0.9804	0.6459	0.3049	0.4141	0.7262	0.315	0.4394
Simda	0.0137	0.1465	0.0250	0.7685	0.9640	0.8552	0.7094	0.0952	0.1652	0.4138	0.044	0.0795
Ramdo	0.0388	0.7250	0.0737	0.0000	0.0000	0.0000	0.2500	0.1416	0.1655	0.0000	0.0000	0.0000
Suppobox	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Locky	0.0000	0.0000	0.0000	0.3492	0.2767	0.3088	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Tempedreve	0.0015	0.8000	0.0031	0.9507	0.9766	0.9635	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Qadars	0.0309	0.7500	0.0594	1.0000	1.0000	1.0000	0.3333	0.0416	0.0740	0.0000	0.0000	0.0000
Symmi	0.0065	0.1538	0.0125	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Banjori	0.9143	0.1051	0.1885	0.6667	0.2857	0.4000	0.9977	0.9999	0.9988	0.9959	0.9995	0.9977
Tinba	0.0000	0.0000	0.0000	0.6000	0.4167	0.4918	0.7805	0.9723	0.8659	0.7545	0.9483	0.8404
Hesperbot	0.0037	0.0526	0.0069	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Fobber	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Alexa	1.0000	0.0002	0.0003	0.9899	0.9868	0.9883	0.9478	0.9867	0.9669	0.9418	0.9900	0.9653
Dyre	0.9697	1.0000	0.9846	0.1646	0.0567	0.0844	0.9717	1.0000	0.9856	1.0000	1.0000	1.0000
Cryptowall	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Corebot	0.0017	0.4000	0.0035	0.3116	0.2191	0.2573	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
P	0.2727	0.2250	0.2466	0.0645	0.0140	0.0230	0.4666	0.2083	0.2762	1.0000	0.2000	0.3333
Bedep	0.0060	0.1471	0.0115	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Matsnu	0.0000	0.0000	0.0000	0.0800	0.0435	0.0563	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
PT Goz	0.9811	0.6682	0.7950	0.9091	1.0000	0.9524	0.9774	0.9868	0.9821	0.9932	0.9970	0.9951
Necurs	0.0244	0.0729	0.0366	0.0000	0.0000	0.0000	0.2139	0.0347	0.0588	0.0000	0.0000	0.0000
Pushdo	0.0036	0.2353	0.0071	0.1071	0.0268	0.0429	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Cryptolocker	0.0163	0.6917	0.0318	0.6406	0.5538	0.5940	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Dircrypt	0.0017	0.0909	0.0034	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Shifu	0.0250	1.0000	0.0489	0.2222	0.2000	0.2105	0.1032	0.0567	0.0700	0.0000	0.0000	0.0000
Bamital	0.6316	1.0000	0.7742	0.4839	0.5797	0.5275	0.8555	0.8055	0.8226	1.0000	1.0000	1.0000
Kraken	0.0041	0.0196	0.0068	0.4545	0.4545	0.4545	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Nymaim	0.0085	0.2250	0.0165	0.3062	0.3900	0.3431	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Shiotob	0.2404	0.2749	0.2565	0.4767	0.3761	0.4205	0.8048	0.6480	0.7175	0.9074	0.5857	0.7119
W32.Virut	0.0035	1.0000	0.0070	0.4403	0.2439	0.3139	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Micro-averaging	0.8085	0.0782	0.1426	0.8652	0.8854	0.8751	0.8853	0.9045	0.8947	0.8754	0.9020	0.8885
Macro-averaging	0.1808	0.3510	0.2386	0.3150	0.3031	0.3089	0.3088	0.2527	0.2779	0.3193	0.2331	0.2695

micro- and macro-averaging can be calculated as follow.

$$B_{macro} = \frac{1}{q}\sum_{\lambda=1}^{q} B(TP_\lambda, TN_\lambda, FP_\lambda, FN_\lambda) \quad (16)$$

$$B_{micro} = B(\sum_{\lambda=1}^{q} TP_\lambda, \sum_{\lambda=1}^{q} TN_\lambda, \sum_{\lambda=1}^{q} FP_\lambda, \sum_{\lambda=1}^{q} FN_\lambda) \quad (17)$$

3.3 Results

In two-class problem, all the malicious domains are grouped into the single DGA class. The supervised learning methods aim to determine whether an input domain name is automatically generated. Fig. 4 illustrates the various ROC curves and their Area under the Curve (AUC). As it can be seen, the highest detection rate is achieved by the Recurrent SVM (AUC=0.9969). This is followed by the Bidirectional LSTM (AUC=0.9964), CNN+LSTM (AUC=0.9959). All the LSTM variants are proved to be better than the original LSTM (AUC=0.9955). The accuracy of C4.5 decision tree is slightly lower than that of LSTM, while HMM demonstrates to be the worst performer (AUC=0.8965).

In multiclass problem, the macro-averaging and micro-

Table 3: Performance comparison of LSTM, Recurrent SVM, CNN+LSTM and Bidirectional LSTM on the multiclass task

	LSTM			**Recurrent SVM**			**CNN+LSTM**			**Bidirectional LSTM**		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Geodo	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Beebone	0.4000	0.2250	0.2872	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Murofet	0.7197	0.5509	0.6185	0.7059	0.5926	0.6443	0.9014	0.3422	0.4961	0.7459	0.5170	0.6107
Pykspa	0.8294	0.6782	0.7457	0.8744	0.6963	0.7753	0.7035	0.5977	0.6463	0.8222	0.6801	0.7445
Padcrypt	0.9242	0.5833	0.7077	1.0000	0.8000	0.8889	0.9091	0.7692	0.8333	0.7500	0.4000	0.5217
Ramnit	0.5786	0.8226	0.6793	0.5632	0.8143	0.6659	0.5657	0.7992	0.6625	0.5953	0.7869	0.6778
Volatile	0.9600	0.4000	0.5543	0.9167	0.7857	0.8462	1.0000	0.7333	0.8462	1.0000	0.8182	0.9000
Ranbyus	0.4239	0.5040	0.4593	0.3897	0.4905	0.4343	0.4313	0.6181	0.5081	0.4077	0.6883	0.5120
Qakbot	0.7005	0.5565	0.6196	0.7237	0.5160	0.6024	0.7178	0.4749	0.5716	0.7446	0.5049	0.6017
Simda	0.9067	0.8125	0.8525	0.9574	0.8982	0.9268	0.7636	0.7976	0.7802	0.7591	0.9121	0.8286
Ramdo	0.9658	0.9750	0.9702	0.9722	1.0000	0.9859	0.8864	1.0000	0.9398	1.0000	0.8780	0.9351
Suppobox	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2500	0.0556	0.0909
Locky	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Tempedreve	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Qadars	0.0000	0.0000	0.0000	1.0000	0.2727	0.4286	0.7143	0.6250	0.6667	0.0000	0.0000	0.0000
Symmi	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5000	0.0714	0.1250	0.3333	0.0667	0.1111
Banjori	0.9992	1.0000	0.9996	0.9999	1.0000	0.9999	0.9996	1.0000	0.9998	0.9996	1.0000	0.9998
Tinba	0.8884	0.9815	0.9327	0.8823	0.9707	0.9244	0.8946	0.9377	0.9157	0.8843	0.9778	0.9287
Hesperbot	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Fobber	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Alexa	0.9727	0.9929	0.9827	0.9787	0.9924	0.9855	0.9681	0.9890	0.9785	0.9753	0.9893	0.9822
Dyre	0.9755	0.9925	0.9839	0.9742	0.9934	0.9837	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Cryptowall	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0625	0.1176
Corebot	0.0000	0.0000	0.0000	1.0000	0.2500	0.4000	1.0000	0.1667	0.2857	0.8000	0.6667	0.7273
P	0.7521	0.3050	0.3858	0.5312	0.4595	0.4928	0.7778	0.3256	0.4590	0.7143	0.5128	0.5970
Bedep	0.8608	0.2588	0.3965	0.8182	0.2571	0.3913	0.5556	0.1667	0.2564	0.7647	0.3421	0.4727
Matsnu	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
PT Gz	0.9958	0.9994	0.9976	0.9992	0.9992	0.9992	0.9976	0.9992	0.9984	0.9985	0.9985	0.9985
Necurs	0.4673	0.0583	0.1036	0.3651	0.0922	0.1472	0.4778	0.0911	0.1530	0.4928	0.1475	0.2270
Pushdo	0.8806	0.1706	0.2744	0.5238	0.3438	0.4151	0.3158	0.3636	0.3380	0.6154	0.2051	0.3077
Cryptolocker	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2143	0.0270	0.0480	0.1538	0.0172	0.0310
Dircrypt	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Shifu	0.4064	0.3064	0.3416	0.2603	0.3800	0.3089	0.2405	0.3800	0.2946	0.3929	0.4490	0.4190
Bamital	0.7833	0.5500	0.6366	0.8000	0.6667	0.7273	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Kraken	0.1666	0.0039	0.0076	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3077	0.0385	0.0684
Nymaim	0.2875	0.0040	0.0692	0.2432	0.0667	0.1047	0.1667	0.0084	0.0160	0.1500	0.0462	0.0706
Shiotob	0.9114	0.8845	0.8976	0.9587	0.8788	0.9170	0.9234	0.8821	0.9023	0.9472	0.8826	0.9137
W32.Virut	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Micro-averaging	0.9193	0.9315	0.9253	0.9206	0.9314	0.9260	0.9159	0.9253	0.9206	0.9232	0.9309	0.9270
Macro-averaging	0.4672	0.3583	0.4055	0.5115	0.4268	0.4653	0.5164	0.4254	0.4665	0.5422	0.4379	0.4845

averaging recall, precision and F1-score are shown in Tables 2 and 3. Macro-averaging treats all classes equally, while micro-averaging favors the class, which have more samples. Macro-averaging should be a better measure; however micro-averaging is also presented for interested readers. It can be proved that macro-averaging recall is equal to the accuracy, reported in the literature. In Table 2, HMM achieves lower detection rate than expected. The rationale for this is that HMM requires a huge amount of data to train the model, while several DGA classes, such as *Tempedreve* and *Corebot* have very little representation in the training data. We note that in [2] HMM was only evaluated using *Conficker*, *Morofet*, *Bobax*, and *Sinowal* malwares. It becomes obvious that C4.5 is superior to both the SVM and ELM. The dominance of Recurrent SVM and Bidirectional LSTM was established by a large margin, leaving LSTM and CNN+LSMT in a group with very small difference between them. LSTM cannot recognize 15 DGA families. This number is reduced to 8 when Bidirectional LSTM is applied to detect malicious domains. Apart from HMM, the implicit features based methods were observed to be better than the hand-crafted features based ones.

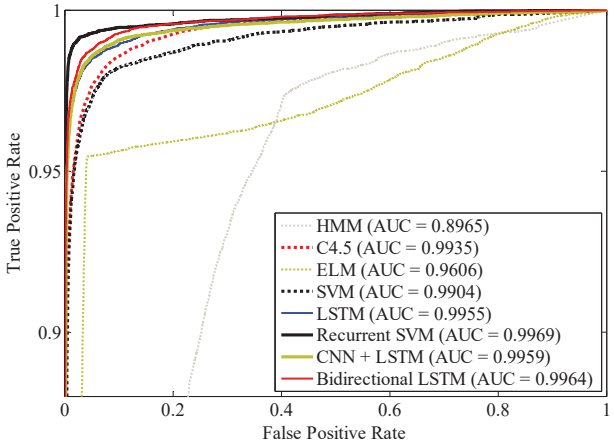

Figure 4: The ROC curves related to the supervised learning methods.

In order to compare each pair of algorithms, we conduct the Wilcoxon signed ranks test [24] using KEEL Data-Mining Software Tool [25]. Wilcoxon singed ranks test compares for the positive and negative difference. Table 4 shows the detailed ranks; each number above the diagonal is the sum of ranks (denoted as R^-) for the data classes on which the algorithm on the row is better than the algorithm in the corresponding column and each number below the diagonal is the sum of ranks (denoted as R^+) for the data classes on which the algorithm on the column is better than the algorithm in the corresponding row [25]. For the confidence level $\alpha = 0.95$ and $N = 38$, the two algorithms are considered as being significantly different if the smaller of R^- and R^+ is less than 235. As it can be seen in Table 5, Recurrent SVM and Bidirectional LSTM are the best performers. A significant difference is also observed between Bidirectional LSTM and other supervised learning methods.

Table 6 illustrates the evaluation time, which is critical for practical uses. There is almost no computation cost in LSTM (9 ms). In Bidirectional LSTM, additional processes are needed because updating input and output layers cannot be achieved at once [12]. Bidirectional LSTM requires 27 ms to process a domain. The evaluation time related to ELM is given in [7]. C4.5, ELM and SVM are most computationally expensive since these methods are based on the hand-crafted attributes. It is clear that HMM, C4.5, ELM and SVM are not suitable for real-time DGA detection applications.

There are still 8 DGA malwares that cannot be detected by all the supervised learning methods. *Geodo*, *Tempedreve*, *Hesperbot*, *Fobber*, *Dircrypt*, *Qadars* and *Locky* are misclassified as *Ramnit* since these malwares share a common generator, which has uniform distribution over the characters. *Matsnu* is based on pronounceable domains. Hence, it cannot be isolated from *Alexa*.

Table 4: Ranks computed by the Wilcoxon test

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
HMM (1)	-	210.5	227	311	128	100	63	46.5
C4.5 (2)	492.5	-	406.5	423	277.5	219.5	237.5	223
ELM (3)	476	334.5	-	413.5	126	96	63.5	53
SVM (4)	392	318	289.5	-	125.5	96	64	47.5
LSTM (5)	575	425.5	577	615.5	-	199	295.5	179
Recurrent SVM (6)	603	483.5	607	607	542	-	471	281
CNN+LSTM (7)	640	503.5	639.5	639	445.5	232	-	192
Bidirectional LSTM (8)	656.5	518	688	655.5	524	422	549	-

Table 5: Summary of the Wilcoxon test. ● the method in the row improves the method of the column, while ○ the method in the column improves the method of the row. Upper diagonal of level significance α = 0.9; Lower diagonal level of significance α = 0.95

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
HMM (1)	–	○	○		○	○	○	○
C4.5 (2)	●	–				○	○	○
ELM (3)			–		○	○	○	○
SVM (4)				–	○	○	○	○
LSTM (5)	●		●	●	–	○		○
Recurrent SVM (6)	●	●	●	●	●	–	●	
CNN+LSTM (7)	●		●	●			–	○
Bidirectional LSTM (8)	●	●	●	●	●		●	–

Table 6: The evaluation time (in ms) of HMM, C4.5, ELM, SVM, LSTM, Recurrent SVM, CNN+LSTM, Bi-LSTM

Method	Evaluation time	Method	Evaluation time
HMM	48	LSTM	9
C4.5	90	Recurrent SVM	15
ELM	420	CNN+LSTM	12
SVM	100	Bidirectional LSTM	27

4 CONCLUSIONS

DGA botnets have become a technology backbone to support cyber-criminals. The supervised learning provides a mean to recognize and shut down this type of botnet. We have thoroughly investigated various supervised learning methods, including Hidden Markov Model, C4.5 decision tree, Support Vector Machines, Extreme Learning Machine, Long Short-Term Memory network, Recurrent SVM, CNN+LSTM and Bidirectional LSTM. Experiments demonstrated that Bidirectional LSTM and Recurrent SVM achieve the highest detection rate on both the binary and multiclass classification problems. These methods share some important features with the LSTM, and hence, making them amenable to real-time detection applications.

ACKNOWLEDGMENTS

This research is supported by the Vietnam Ministry of Education and Training research project "Development of DDoS attack prevention and Botnet detection system" B2016-BKA-06.

REFERENCES

[1] S. Yadav, A.K.K Reddy, A.L.N. Reddy, S. Ranjan, *Detecting algorithmically generated domain-flux attacks with DNS traffic analysis*, IEEE/ACM Transactions on Networking 20.5 (2012): 1663-1677.

[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: the 21st USENIX Security Symposium (USENIX Security 12) (2012).

[3] Y. Zhou, Q.S. Li, Q. Miao, K. Yin, *DGA-Based Botnet Detection Using DNS Traffic*, Journal of Internet Services and Information Security, 3.3/4 (2013): 116-123.

[4] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, *Phoenix: DGA-based botnet tracking and intelligence*, International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA) (2014), LNCS 8550, 192-211

[5] H. Zhang, M. Gharaibeh, S. Thanasoulas, and C. Papadopoulos, *Botdigger: Detecting dga bots in a single network*, Proceedings of the IEEE International Workshop on Traffic Monitoring and Analaysis. 2016.

[6] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis, *Ndss*. 2011.

[7] Y. Shi, C. Gong and L. Juntao, Malicious Domain Name Detection Based on Extreme Machine Learning, *Neural Processing Letters* (2017): 1-11.

[8] J. Woodbridge, H.S. Anderson, A. Ahuja, and D. Grant, *Predicting Domain Generation Algorithms with Long Short-Term Memory Networks*. arXiv preprint arXiv:1611.00791 (2016).

[9] Y. Tang, Deep learning using linear support vector machines, *arXiv preprint arXiv:1306.0239* (2013).

[10] S.X. Zhang, R. Zhao, C. Liu, J. Li, and Y. Gong *Recurrent support vector machines for speech recognition*, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016.

[11] Kim, Yoon, et al. Character-Aware Neural Language Models. AAAI. 2016.

[12] A. Graves, and J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, Neural Networks 18.5 (2005): 602-610.

[13] S. Hochreiter, and J. Schmidhuber, Long short-term memory, Neural computation 9(8) (1997): 1735-1780.

[14] F.A. Gers, J. Schmidhuber, and F. Cummins, Learning to forget: Continual prediction with LSTM, Neural computation 12(10) (2000): 2451-2471.

[15] Jay Jacobs, *Building a DGA Classifier: Feature Engineering*. Available online at: http://datadrivensecurity.info/blog/posts/2014/Oct/dga-part2/. October 2014

[16] S. Krishnan, T. Taylor, F. Monrose, and J. McHugh, *Crossing the threshold: Detecting network malfeasance via sequential hypothesis testing*, 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (2013) 1–12

[17] N. Cristianini, and J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge university press, 2000.

[18] J. Milgram, M. Cheriet, and R. Sabourin, *"One against one" or "one against all": Which one is better for handwriting recognition with SVMs?*, Tenth international workshop on frontiers in handwriting recognition. La Baule, 2006.

[19] J.R. Quinlan, C4. 5: programs for machine learning, Elsevier, 2014

[20] G.B Huang, Q.-Y. Zhu, and C.-K. Siew, *Extreme learning machine: theory and applications*, Neurocomputing 70.1 (2006): 489-501.

[21] W. Yin, K. Kann, M. Yu, and H. Schütze, *Comparative Study of CNN and RNN for Natural Language Processing*, arXiv preprint arXiv:1702. 01923 (2017).

[22] V. Tong, and G. Nguyen, A method for detecting DGA botnet based on semantic and cluster analysis, *Proceedings of the Seventh Symposium on Information and Communication Technology*. ACM, 2016.

[23] P. Su, X. Ding, Y. Zhang, Y. Li, and N. Zhao, *Predicting Blood Pressure with Deep Bidirectional LSTM Network*, arXiv preprint arXiv:1705.04524 (2017).

[24] J. Demšar, *Statistical comparisons of classifiers over multiple data sets*, Journal of Machine learning research 7 (2006): 1-30.

[25] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, *Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework*, Journal of Multiple-Valued Logic Soft Computing 17 (2–3) (2011) 255–287.

[26] Chollet, François. *Keras (2015)*. URL http://keras. io (2017).

[27] F, Pedregosa, et al., *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research 12 (2011): 2825-2830.

[28] Does Alexa have a list of its top-ranked websites? Available online at: https://support.alexa.com/hc/en-us/articles/200449834-Does-Alexa-have-a-list-of-its-topranked-websites-. (2017).

[29] Bambenek Consulting - Master feeds. Available online at: http://osint.bambenekconsulting.com/feeds/ (2016).

[30] M. Masud, T. Al-khateeb, L. Khan, B. Thuraisingham, and K. Hamlen, *Flow-based identification of botnet traffic by mining multiple log files*, in Distributed Framework and Applications, 2008. DFmA 2008. First International Conference on, oct. 2008, pp. 200 –206.

[31] M. Antonakakis, et al., Building a Dynamic Reputation System for DNS, USENIX security symposium. 2010.

[32] Kotsiantis, Sotiris B., I. Zaharakis, and P. Pintelas. *Supervised machine learning: A review of classification techniques*. (2007): 3-24.