Computers, Materials & Continua CMC, vol.61, no.3, pp.1285-1304, 2019

Detecting Domain Generation Algorithms with Bi-LSTM

Liang Ding[1, *], Lunjie Li[1], Jianghong Han[1], Yuqi Fan[2, *] and Donghui Hu[1]

Abstract: Botnets often use domain generation algorithms (DGA) to connect to a command and control (C2) server, which enables the compromised hosts connect to the C2 server for accessing many domains. The detection of DGA domains is critical for blocking the C2 server, and for identifying the compromised hosts as well. However, the detection is difficult, because some DGA domain names look normal. Much of the previous work based on statistical analysis of machine learning relies on manual features and contextual information, which causes long response time and cannot be used for real-time detection. In addition, when a new family of DGA appears, the classifier has to be re-trained from the very beginning. This paper presents a deep learning approach based on bidirectional long short-term memory (Bi-LSTM) model for DGA domain detection. The classifier can extract features without the need for manual feature extraction, and the trainable model can effectively deal with new unknown DGA family members. In addition, the proposed model only needs the domain name without any additional context information. All domain names are preprocessed by bigram and the length of each processed domain name is set as a value longer than the most samples. Bidirectional LSTM model receives the encoded data and returns labels to check whether domain names are normal or not. Experiments show that our model outperforms state-of-the-art approaches and is able to detect new DGA families reliably.

Keywords: Bidirectional LSTM, network security, DGA.

1 Introduction

Botnets are hidden dangerous networks with great threat to the network security operation and the user data security. The Botnets are some kind of one to many centralized controlled networks, which is controlled by a Bot master and lots of compromised hosts. Through command and control (C2) Server, Bot master sends orders to the compromised hosts. Such networks have been created to conduct large-scale illegal activities, such as launching denial-of-service attacks, phishing attacks, cryptoviral extortion, which bring a large threaten to cloud computing and big data environment [Cheng, Xu, Tang et al. (2018)].

Actually, attackers usually use multiple domain names to connect to the C2 server when

[1] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, 230009, China.

[2] Department of Computer Science, University of Texas at Dallas, Campbell Rd, Richardson, Texas, 75080, USA.

[*] Corresponding Authors: Liang Ding. Email: liangding@hfut.edu.cn;
Yuqi Fan. Email: yuqi.fan@utdallas.edu.

operating botnet, so as to control the victim machines [Kührer, Rossow and Holz (2014)]. Some malware rely on static lists of domains and IP addresses that were hardcoded to connect compromised machines [Stonegross, Cova, Gilbert et al. (2011)]. The domains are often coded in malicious programs, giving attackers the flexibility to easily change the domains and their IP addresses [Hampton and Baig (2015)]. The biggest advantage of this connection is that it is easy to be implemented, while the disadvantage is that it is very easy to be detected by the authorities. Due to limited amounts of domains and IP addresses, defenders can blacklist them based on reverse techniques. However, attackers propose corresponding countermeasure by using domain generation algorithms to dynamically generate a large number of pseudo random domain names in a short period of time, effectively increase the difficulty of blacklisting and detection.

Domain generation algorithms can produce a series of pseudo random domain names, which contain strings and numbers using some seeds, encryption algorithms, such as differences operations. We can predict the generated domains by collecting samples and reverse engineering. Afterwards, we can preregister the domains or put them in a blacklist. However, there may be a huge number of generated domains in a short term, while the list cannot be updated in time. Therefore, a real-time detection of malicious domain names produced by the DGA is needed.

With the update of DGAs, the number of generated domain names is increasing and the defense work becomes more difficult. The accuracy of the traditional classification algorithm and the hidden markov model is low. The features selection based on the analysis and detection method of natural language features of domain names cannot deal with the large number of features. In addition, some DGAs may even build the algorithm to generate a large number of pseudo-domain names that conform to the characteristics of normal domain names.

In this paper, we design a model to detect domain names generated by DGAs based on bidirectional LSTM neural networks. Compared with the traditional detection methods, the proposed scheme has the following advantages:

1. Our scheme uses a featureless way to handle domain names, by which all information contained in the domain name is retained as much as possible. It also avoids manual feature selection and the difficulty of determining the features effectiveness.
2. Our scheme can well adapt to the detection of pseudo-domain names generated by new DGA. Compared with the defect of large data samples that need to be retrained in the traditional detection scheme, the proposed scheme only needs to continue the training on the original model.
3. Our scheme performs in a real-time and low-cost way. The model trained through the pre-training data samples can be deployed and used directly, and can classify the domain names and quickly blacklist suspicious domain name without requiring more information.

In this paper, we make the following contributions:

1. We obtain statistics on frequency distribution of domain names' composition and length, and analyze the differences between the normal and the DGA domain names. On the premise of retaining the original domain name information as much as possible, it is determined that bigram processing can make the differences more obvious.

2. In the experiment, we trained the processed data on LSTM networks and Bi-LSTM networks. Experimental results show that the Bi-LSTM model can effectively distinguish normal domain names and DGA domain name. The FPR (False Positive Rate) value tested by this model is 30.6% higher than the result without bigram processing.

2 Related work

There exist various approaches to detect DGA domain names. Wang et al. [Wang, Lin and Lin (2016)] proposed a DGA botnet detection mechanism utilizing the feature based characteristics of social networks. Antonakakis et al. [Antonakakis, Perdisci, Nadji et al. (2012)] presented a new technique to detect randomly generated domains without reversing. Their approach used a combination of clustering and classification algorithms. Kwon et al. [Kwon, Lee, Lee et al. (2016)] introduced PsyBoG, a fast and scalable approach, for detecting malicious behavior within large volumes of DNS traffic. Anderson et al. [Anderson, Woodbridge and Filar (2016)] leveraged the concept of generative adversarial networks to test the hypothesis of whether adversarial generated domains may be used to augment training sets in order to enhance the machine learning models against new DGAs.

Wang et al. [Wang, Jia and Zhang (2018)] studied the characters' features of DGA domain names and extracted five attributes for the Support Vector Machine (SVM) model. Chen et al. [Chen, Yan, Pang et al. (2018)] trained the classifier model through Support Vector Machine (SVM), which is based on supervised machine learning. Huang et al. [Huang, Wang, Zang et al. (2018)] proposed Helios, a DGA detection approach based on a neural language model, which exploits the word formation of domain names to identify those generated by DGAs.

Yadav et al. [Yadav, Reddy, Reddy et al. (2012)] described a model by testing distribution of alphanumeric characters and bigrams in all domains to detect DGA domain names. Wang et al. [Wang and Chen (2017)] proposed N-Gram features to increase the accuracy of classification models. Schiavoni et al. [Schiavoni, Maggi, Cavallaro et al. (2014)] combined linguistic and IP-based features and presented the Phoenix framework to identify DGA domain names. Mowbray et al. [Mowbray and Hagen (2014)] proposed a method by identifying client IP addresses with an unusual distribution of second-level string lengths to classify domain names. Woodbridge et al. [Woodbridge, Anderson, Ahuja et al. (2016)] described a method to predict domains generated by DGAs with Long Short-Term Memory networks.

Based on method described by Woodbridge et al. [Woodbridge, Anderson, Ahuja et al. (2016)], Lison et al. [Lison and Mavroeidis (2017)] compared the empirical performance of various design choices, using of embedding, type of recurrent units, etc. For processing sequential data such as natural language, a neural network model of RNN (Recurrent Neural Network) usually used [Mahoney (1999); Mikolov, Karafiat, Burget et al. (2010); Robinson (1994)]. Hochreiter et al. analyzed the problem of gradient explosion and disappearance brought by back propagation through time algorithm, which brought problems such as gradient oscillation and learning difficulty to the learning algorithm. Network structure of LSTM was proposed [Gers, Schmidhuber and Cummins (2000); Gers, Schraudolph and Schmidhuber (2002); Hochreiter and Schmidhuber (1996);

Hochreiter and Schmidhuber (1997)].

The basic idea of bidirectional recurrent neural network (BRNN) is to propose that each training sequence are two RNNs, forward and backward respectively [Schuster and Paliwal (1997)]. This structure provides complete past and future context information for each point in the output layer's input sequence. As a member of BRNN, Bi-LSTM has its general structure characteristics [Graves and Schmidhuber (2005)]. Liu used Bi-LSTM proposed a sentence encoding-based model for recognizing text entailment [Liu, Sun, Lin et al. (2016)].

The method presented in this paper is based on the operations presented in Woodbridge et al. [Woodbridge, Anderson, Ahuja et al. (2016)]. This paper improves the existing approach as follows:

1. By analyzing the character composition and length of the domain names, we can find the relationship between the characters in the domain name, and use the binary grammar (bi-gram) method to preprocess the domain name.
2. By using bidirectional training sample data from the Bi-LSTM network, future context relationships are introduced in addition to the past context relationships.
3. Based on a large number of DGA data samples, the results are more suitable for practical use.

3 System implementation

For effective detection of domain names, we set up a dictionary that contains the characters of domain names and the corresponding positive integer values by analyzing the characteristics of domain names. According to the form of the domain names, we convert the characters into one-dimensional vectors. When the vectors are obtained by Embedding layer, we put them in LSTM neural networks or Bi-LSTM neural networks to get labels, based on which we determine whether the test domain name is generated by DGAs. The normal set and DGA family set in the data set are divided into training set and test set respectively at a ratio of 4:1. The design and implementation of the system is divided into three parts: characteristic analysis, data processing and neural network model.

3.1 Characteristic analysis

Because the different levels of domain names on the Internet are managed by different agencies, the way that each agency manages domain names and the rules for naming them are also different. But there are some common rules for naming names: the domain name contains 26 English alphabet letters (case insensitive), 10 Arabic numerals, and a few other characters. We mainly analyze the following four aspects:

• *Source*

The experimental domain name data used in this paper are from the global top one million domain names published by Alexa website and more than 1.4 million domain names generated by 28 different domain name generation algorithms that ensure data samples are representative and authoritative.

• *Character composition*

Though the analysis about word frequency statistics of the domain name samples processed by unigram, we find that the regularity of the distribution of each character in the domain name. It can be seen from Fig. 1 that the frequency distributions of each character in the normal sample and DGA sample are significantly different.

In both samples, the frequency of numbers is lower, and the frequency of English alphabet letters is relatively higher, which indicates that English alphabet letters are the main constituent characters of domain names. In the DGA sample, the frequency of the numbers is still low, but both are higher than the normal samples. The frequency distribution of English alphabet letters in the DGAs domain names is of the average level, and the overall fluctuation is smaller than normal.

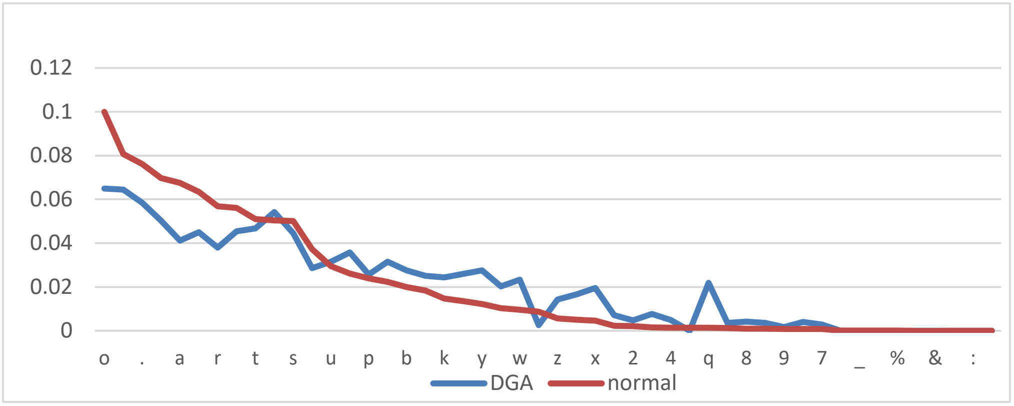

Figure 1: Probability of each symbol

Fig. 2 shows that in normal samples, the international suffix domain name ".com" appears in a large number in normal samples and the frequencies of "co", ".c", "om" and "m%" are much higher than other characters. On the contrary, the frequency of remaining characters is smoothly reduced. The frequency difference of characters in the DGA samples is large, indicating that the frequency of high-frequency characters in normal samples is not as high as that in the DGA samples.

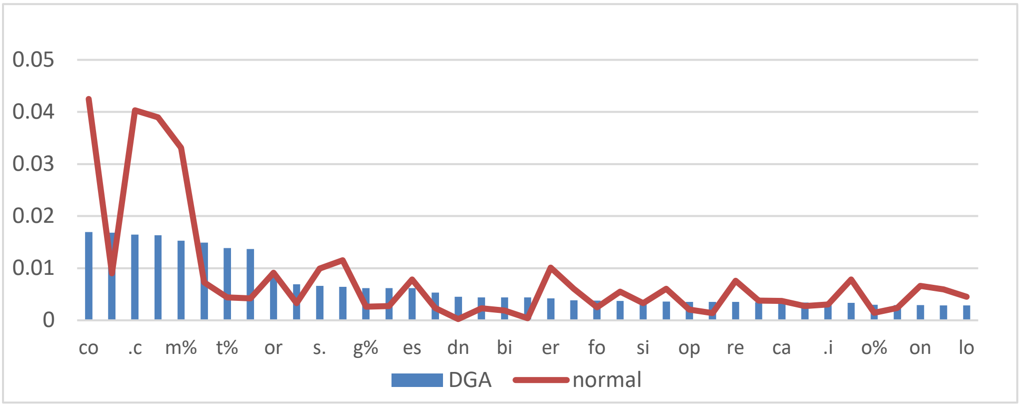

Figure 2: Probability of each component

• *Length*

Obviously, the same domain name samples processed by unigram and bigram respectively differ by only 1 in length. As can be seen from Fig. 3, most domain names are between 4 and 35 in length, and the distribution characteristics of the two data samples are different. In normal samples, the domain length distribution is close to the normal distribution. A domain name with a length of 12 has the highest frequency, and the domain name of other lengths is relatively small. In the DGA samples, due to the limitations of the generation algorithm, the domain length distribution presents a centralized situation, which is in accordance with the characteristics of the pseudo-random generation.

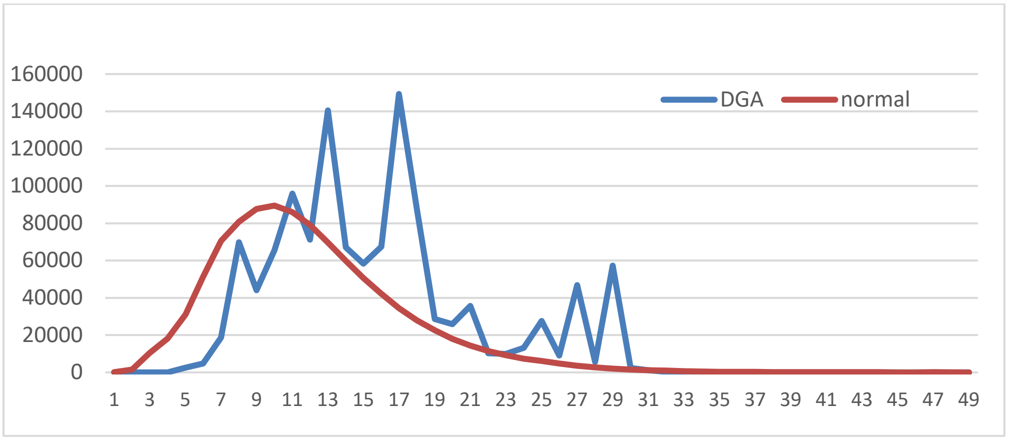

Figure 3: Statistics of domain name length

• *First character*

According to the statistical analysis of the first character of the initial domain name in Fig. 4, it can be seen from the figure that in the normal sample and DGA sample, the proportion of the first character is English letter is much higher than that of the number. In normal samples, letters 'q', 'x' and 'z' are relatively low. The frequency change of each English letter in the DGA samples is more gradual. However, due to the individual DGA family algorithm, the probability of '0' and '1' being the first character of the domain name in the digital part is greatly increased compared with other numbers.

Based on the above analysis of domain name characteristics, we are more convinced that there is a textual natural language difference between normal and DGA domain names. Besides, there are great differences between them with the processing of bigram.

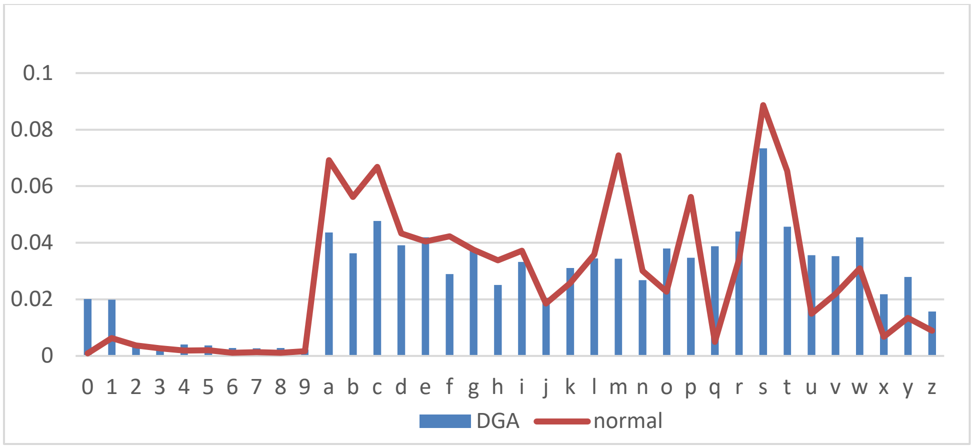

Figure 4: Probability of each initial

3.2 Data processing

We create the dictionary for all characters that appear by applying unary grammar to all existing domain names samples. The statistics show that all domain names consist of 44 characters. By artificially specifying that each character corresponds to a different positive integer, each single domain name in the sample is transformed into one-dimensional vectors.

To conduct binary grammar processing for all domain names, we need to mark the beginning and end of the domain names with '%' (no symbol % is included in the known domain names). The lengths of the sequence are different, after each domain name being processed by different ways. Unigram is N, and bigram is N+1 (N is the length of the domain name). We obtain statistics on the processed samples and create the dictionary.

The processing results of unary and binary grammar are different. Unary grammar results show that each domain name is separated by each character to form a character sequence. Binary grammar results illustrate that each domain name extracts adjacent characters (including start and end character '%') one by one to form a sequence.

The data set contains a collection of various DGA domain names, as well as a global one million normal domain names downloaded from the Alexa website. We mark each of the domain strings that contain the DGA domain name as 1, and all of the normal domain names as 0. For different types of DGA samples, we also generate 40,000 pieces of 28 different types of DGA domain names. 40,000 normal domain names are randomly selected from the normal data set, and the labels are treated the same as the DGA samples.

3.3 Neural network model

In the experiment, we construct the neural network through Keras. In this paper, four training models are set up. We use unigram, bigram and LSTM, Bi-LSTM to combine with each other and develop modules that use various grammars to generate vectors and perform machine learning training. For the training models, we set up sequential model with Embedding layer, the LSTM (Bi-LSTM) layer, Dropout layer, Dense layer (activation function is sigmoid).

• *Embedding layer*

The input to the embedding layer in the deep learning model is a vector which encodes each character of the domain name string into a sequence of positive integers via a dictionary. This dictionary is obtained by counting the characters that appear in all domain names and then encoding them with a non-zero positive integer. The dimensions of vectors can be either unfixed or fixed. As the fixed vector dimension can greatly improve the model training effect, we choose a fixed vector dimension.

As for the determination of vector dimension, the previous method is to determine the longest domain name length in the data set, and set the value as vector dimension to hold all domain name strings. If the vector dimension does not reach the maximum dimension, we pad zeroes to make it reach the maximum dimension. This process can be done using sequence preprocessor function in Keras. The statistics in Fig. 5 show that the lengths of domain names mainly concentrate in one region, while the number of large domain names is very small. We count 99.9% of the total number of domain names, all of which are in the range of 4-38 characters, while the number of domain names over 38 characters is very small. Therefore, we determine the dimension of the vector to be 38, and fixe the other vector dimensions to be 38 using the sequence preprocessor function in Keras.

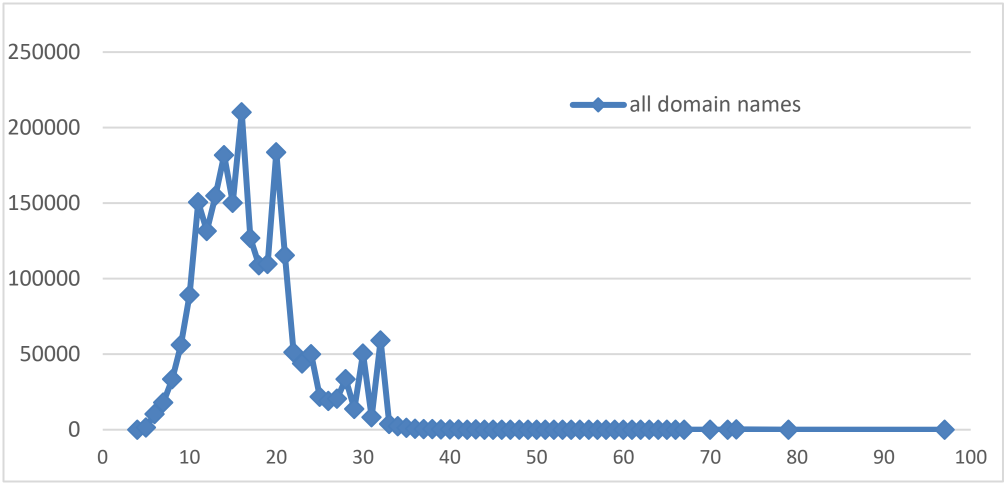

Figure 5: Statistics of domain name length

Compared with previous methods, their vector dimensions are often greatly affected by a small number of long domain names, which can easily cause too much computation. The approach we use can cover almost all domain names, while greatly reducing the training complexity.

After transforming the results obtained by the two grammars into vector matrices, we respectively input them into LSTM and Bi-LSTM models for training. The input length of training is the dimension of each vector. Among this article, it is set as 38. The specific analysis is as follows:

For simplicity, let us first introduce some notations. We define V is a vector and v_i is an element in the vector. We define $V_{a:b}$ as the row vector V from a to b, i.e.,

$$V_{a:b} = [V_a \ V_{a+1} \cdots V_b] \tag{1}$$

Define M is a matrix, and $m_{i,j}$ is an element in the matrix. We define $M_{i,a:b}$ as the row vector of matrix M consisting of elements from columns a to b of row i, i.e.,

$$M_{i,a:b}=[m_{i,a} \ m_{i,a+1} \cdots m_{i,b}] \tag{2}$$

$M_{a:b,j}$ as the vector of M consisting of elements from rows a to b of column j, i.e.,

$$M_{a:b,j}=[m_{a,j} \ m_{a+1,j} \cdots m_{b,j}]^T \tag{3}$$

$M_{a:b,c:d}$ as the sub matrix of M consisting of elements from cell a and c to cell b and d, i.e.,

$$M_{a:b,c:d}=\begin{bmatrix} m_{a,c} & \cdots & m_{a,d} \\ \vdots & \ddots & \vdots \\ m_{b,c} & \cdots & m_{b,d} \end{bmatrix} \tag{4}$$

We convert all domain names to matrix X based on the obtained dictionary containing values as follows:

$$X=\begin{bmatrix} x_{1,1} & \cdots & x_{1,T} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,T} \end{bmatrix} \tag{5}$$

In order to speed up the operation of the program, we take T to be 38 and m to be all the domain names, by the analysis of the domain name length above. We set the vector in the matrix to V_m, that is

$$V_m = [x_{m,1} \ x_{m,2} \cdots x_{m,38}] \tag{6}$$

V_m represents the m-th domain name in the data set, where the element $x_{m,1}$ is the first component in the domain name (single character in unigram and double character under bigram). At the same time, in order to adapt to the input of LSTM and Bi-LSTM network model, we set the dimension of each row of the matrix (tensor) output by the embedding layer to 128. In the embedding layer, according to the size 'l' of the dictionary we input, it performs the unique one-hot encoding of the elements in each column vector, operates on the weight matrix W stored in the embedding layer, and then outputs the matrix.

$$W=\begin{bmatrix} w_{1,1} & \cdots & w_{1,128} \\ \vdots & \ddots & \vdots \\ w_{l+1,1} & \cdots & w_{l+1,128} \end{bmatrix} \tag{7}$$

For the determination of l, we can also assume that set A

$$A = \{x_{1,1}\} \cup \{x_{1,2}\} \cup \ldots \cup \{x_{m,T}\} \tag{8}$$

$$l=|A| \tag{9}$$

The element V_m in each column vector $V_{m,i}$ is encoded by one-hot code

$$V_{m,i}= (0 \ldots 1 \ldots 0) \tag{10}$$

For the sake of demonstration, we might as well assume that we have such a column vector

$$V_m=[x_{m,1}\ \ x_{m,2}\ \ x_{m,3}]^T \tag{11}$$

Among them the one-hot coding of $x_{m,1}$, $x_{m,2}$, $x_{m,3}$

$$x_{m,1}=(1\ 0\ 0) \tag{12}$$

$$x_{m,2}=(0\ 1\ 0) \tag{13}$$

$$x_{m,3}=(0\ 0\ 1) \tag{14}$$

The weight matrix W is

$$W=\begin{bmatrix} w_{1,1} & \cdots & w_{1,128} \\ w_{2,1} & \ddots & w_{2,128} \\ w_{3,1} & \cdots & w_{3,128} \end{bmatrix} \tag{15}$$

Then, the column vector V_m can be converted to the matrix S by the weight matrix W

$$S = V_m \times W \tag{16}$$

$$=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} w_{1,1} & \cdots & w_{1,128} \\ w_{2,1} & \ddots & w_{2,128} \\ w_{3,1} & \cdots & w_{3,128} \end{bmatrix} \tag{17}$$

$$=\begin{bmatrix} w_{1,1} & \cdots & w_{1,128} \\ w_{2,1} & \ddots & w_{2,128} \\ w_{3,1} & \cdots & w_{3,128} \end{bmatrix} \tag{18}$$

In our experiment, the matrix S is

$$S=\begin{bmatrix} s_{1,1} & \cdots & s_{1,128} \\ \vdots & \ddots & \vdots \\ s_{l+1,1} & \cdots & s_{l+1,128} \end{bmatrix} \tag{19}$$

For any member $s_{i,j}$ of the matrix S

$$\forall s_{i,j} \in \{w_{1,1}\} \cup \{w_{1,2}\} \cup \ldots \cup \{w_{l+1,128}\} \tag{20}$$

Through different domain name processing methods of unigram and bigram, we can get dictionaries of different lengths. Through our dataset statistics, the size of the dictionary processed by unigram is 44, and the size of the dictionary processed by bigram is 1789. Then the one-hot code length and the weight matrix W in the Embedding layer are 45 and 1790 respectively.

Although the matrix S obtained by the weight matrix transformation in the experiment is a matrix of 38 rows and 128 columns, the difference between the dimensions of the bigram preprocessing and the unigram preprocessing is significantly different due to the large difference in the dimension during the conversion process.

• *The neural networks (LSTM and Bi-LSTM)*

Each domain name is encoded according to a dictionary and obtained by the embedding layer conversion, and the encoded domain name is input into the corresponding network model for the output calculation. The structure of the LSTM network model is shown in the Fig. 6.

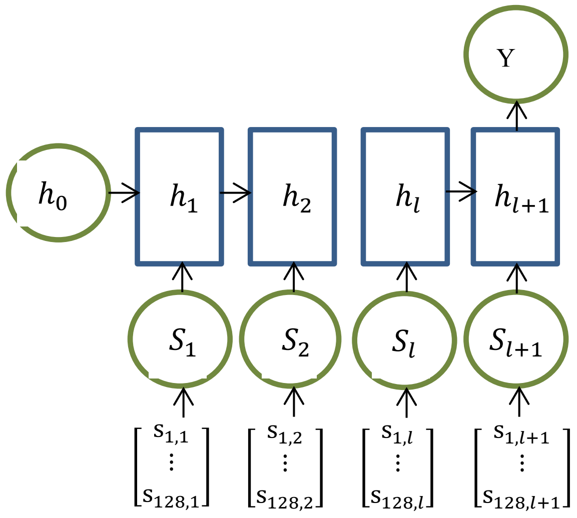
Y
h_0
h_1
h_2
h_l
h_{l+1}
S_1
S_2
S_l
S_{l+1}
$\begin{bmatrix} s_{1,1} \\ \vdots \\ s_{128,1} \end{bmatrix}$
$\begin{bmatrix} s_{1,2} \\ \vdots \\ s_{128,2} \end{bmatrix}$
$\begin{bmatrix} s_{1,l} \\ \vdots \\ s_{128,l} \end{bmatrix}$
$\begin{bmatrix} s_{1,l+1} \\ \vdots \\ s_{128,l+1} \end{bmatrix}$

Figure 6: The structure of the LSTM network

Every input vector S corresponds to a structure h_i, which receives the previous structure h_{i-1} and the current input vector S_i. The model outputs the result to the next structure h_{i+1} by certain calculation, that is

$$h_i = f\ (h_{i-1}, S_i) \tag{21}$$

The final output y is obtained by a certain calculation method from the last structure h_{l+1}, that is

$$y = g\ (h_{l+1}) \tag{22}$$

To control the output results between [0, 1], we use the sigmoid function

$$\mathrm{Sig}(x) = \frac{1}{1+e^{-x}} \tag{23}$$

Then the final output Y is

$$Y = \mathrm{Sig}\ (y) \tag{24}$$

$$= \mathrm{Sig}\ (g\ (h_{l+1})) \tag{25}$$

We make the following decision according to the output result

$$\mathrm{Result} = \begin{cases} Normal & (Y < 0.5) \\ Dga & (Y \geq 0.5) \end{cases} \tag{26}$$

The structure of the Bi-LSTM network model is shown in the Fig. 7.

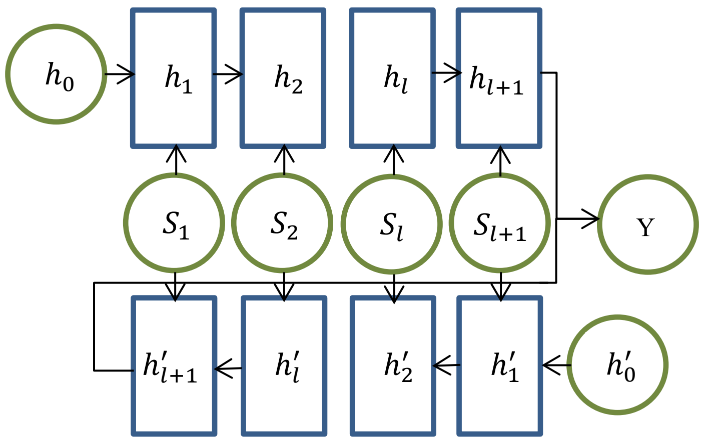

Figure 7: The structure of the Bi-LSTM network

The basic structure of the Bi-LSTM network is similar to LSTM. The final output is obtained by a certain operation of h_{l+1}, h'_{l+1} and the sigmoid function. That is

Y =Sig (g′ (h_{l+1}, h'_{l+1})) (27)

The main process of the system training part we built is as follows.

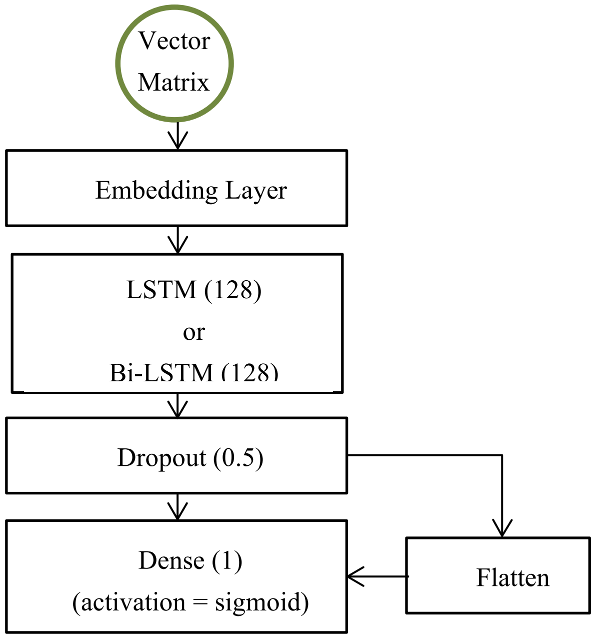

Figure 8: The main process of the system

4 Performance evaluation

In this section, we analyze the results of training and testing. The experiments focus on the detection of the DGA and normal domain names. The normal data are from the Alexa top one million domains and the DGA data are generated by 28 DGA families. Tab. 1 shows the DGA families we used and the number of each family domains. We process the one million normal data and more than 467 thousands DGA data in unigram grammar and binary grammar, and then use LSTM and Bi-LSTM networks model for training and testing.

Table 1: DGA family used

DGA Family	**Frequency**	**DGA Family**	**Frequency**	**DGA Family**	**Frequency**
banjori	16000	murofet	16313	qadars	16714
chinad	16039	necurs	16000	qakbot	17000
corebot	16000	newgoz	16000	ramdo	16000
dircrypt	16006	nymaim	16026	ranbyus	16008
dnschanger	16000	nymaim2	31670	shiotob	16000
fobber	16060	padcrypt	16005	simda	16000
gozi	16000	pizd	16000	suppobox	16000
kraken	16000	proslikefan	16544	symmi	16000
locky	16000	pykspa	16029	tempedreve	16412
				tinba	16404
				total	467230

We use several performance metrics for the evaluation of the detection network models. These metrics are True Positive Rate (TPR), Recall, False Positive Rate (FPR), Precision and ACC. TPR is the ratio between the number of correctly detected DGA domains to the total number of DGA domains. FPR is the ratio between the number of normal domains that are incorrectly classified as DGA and the total number of normal domains. Precision is the ratio between the number of correctly detected DGA domains and the total number of domains detected as DGA domains, Whereas ACC is the ratio between the number of correctly detected DGA domains plus normal domains and the total number of the test domains.

In the following experiments, we evaluate our proposed Bi-LSTM network model in three cases. Case 1 uses unigram, named as Bi-LSTM-Ug. Case 2 use bigram, called as Bi-LSTM-Bg. Case 3 is a hybrid model combining of Bi-LSTM-Bg and LSTM.

We also compare our model with the state of art methods in the respective domains.

- A featureless LSTM model defined in Woodbridge et al. [Woodbridge, Anderson, Ahuja et al. (2016)].
- A SVM classifier model using manual features defined in Chen et al. [Chen, Yan, Pang et al. (2018)]. The manual features of the SVM included the following:
- the length of domain name;
- the ratio between vowel and domain name;
- the entropy of character distribution of the domain name;
- bigram frequency distribution occurrences count.

4.1 Evaluation metrics

FN: False Negative. It is considered as a negative sample, but it is actually a positive sample.

FP: False Positive. It is considered as a positive sample, but it is actually a negative sample.

TN: True Negative. It is considered as a negative sample, but it is actually a negative sample.

TP: True Positive. It is considered as a positive sample, but it is actually a positive sample.

TPR: True positive rate. It can be calculated as follow.

$$\text{TPR} = \frac{\sum TP}{\sum TP + \sum FN} \tag{28}$$

FPR: False positive rate. It can be calculated as follow.

$$\text{FPR} = \frac{\sum FP}{\sum TN + \sum FP} \tag{29}$$

Recall: Recall ratio. It can be calculated as follow.

$$\text{Recall} = \frac{\sum TP}{\sum TP + \sum FN} \tag{30}$$

Precision: The ratio of the number of correctly retrieved samples to the total number of positive samples.

$$\text{Precision} = \frac{\sum TP}{\sum TP + \sum FP} \tag{31}$$

$\boldsymbol{F_1}$: The ratio is the harmonic mean of Precision and Recall.

$$\boldsymbol{F_1} = 2\,\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}} \tag{32}$$

ACC: The ratio of the number of correctly classified samples to the total number of samples.

$$\text{ACC} = \frac{\sum TP + \sum TN}{\sum TP + \sum TN + \sum FP + \sum FN} \tag{33}$$

4.2 Results analysis

According to the results in Tab. 2, the results on ACC, Recall and $\boldsymbol{F_1}$ are the best in the Bi-LSTM model, and the results on Precision and ROC AUC are the best in the Bi-LSTM-Bg model. The results of the Bi-LSTM-Bg model on the three performance metrics of ACC, Recall and $\boldsymbol{F_1}$ are only 0.002, 0.02 and 0.002, respectively.

Table 2: Results of the four models

Model	ACC	PRECISION	RECALL	F_1	ROC AUC
LSTM	0.973	0.976	0.975	0.976	0.9773
Bi-LSTM-Ug	**0.974**	0.972	**0.981**	**0.977**	0.9974
Bi-LSTM-Bg	**0.974**	**0.990**	0.963	0.976	**0.9984**

The results of the three sets of models in each performance metric differ little from each other. With regard to Precision, the Bi-LSTM-Bg model is 1.852% higher than the worst model and 1.434% higher than the sub-optimal model. In the case of Recall, it is the worst model, with a decline of 2.035% compared with the optimal model and 1.231% compared with the sub-inferior model. Precision is the ratio of the true number to the total number of results returned after retrieval, while Recall is the ratio of the true number to the whole data set (retrieved or not). The reasons for the results above can be further explained by Tab. 3.

Table 3: Detailed results of the four models

Model	TP	FP	TN	FN	TPR	FPR	NUM
LSTM	236573	5729	194847	6077	0.975	0.029	443226
Bi-LSTM-Ug	238656	6867	193120	4583	0.981	0.034	443226
Bi-LSTM-Bg	234643	**2274**	**197258**	9051	0.963	**0.011**	443226

According to the results in Tab. 3, under the data samples of the same capacity, the Bi-LSTM-Bg model has significant advantages in FP, TN and FPR, and its results are still improved by 60.307%, 1.237% and 62.069%, respectively, compared with the sub-optimal of each performance metric. Compared with the worst of all performance metrics, 66.885%, 2.143% and 67.647% are increased respectively. But the results on FN with the Bi-LSTM-Bg model is the worst, with a value of 2.128 times that of the optimal model. By introducing to various parameters in Tab. 3, we know FP dropped substantially, FN jumped sharply. Because there are more data samples classified by Bi-LSTM-Bg model as the DGA. It is that its normal probability is lower, so more DGA samples are correct classified. There are more normal samples were mistaken to be classified as the DGA. According to the results in Tab. 2, while the ACC (overall accuracy rate) does not significantly decline and the FPR is significantly reduced, which is more helpful for the detection of DGA family domain names, among which the error rate of normal domain names is acceptable. This conclusion can be drawn from the DGA family domain name detection results in Tab. 4.

Table 4: ACC and FPR of the four models

DGA FAMILY	ACC			FPR			NUM
	LSTM	Bi-LSTM-Ug	Bi-LSTM-Bg	LSTM	Bi-LSTM-Ug	Bi-LSTM-Bg	
banjori	**1.000**	**1.000**	**1.000**	**0.000**	**0.000**	**0.000**	16000
chinad	0.999	0.999	0.999	0.002	0.003	**0.001**	16039
corebot	**1.000**	**1.000**	**1.000**	**0.000**	**0.000**	**0.000**	16000
dircrypt	0.946	0.957	**0.982**	0.086	0.053	0.007	16006
dnschanger	0.962	0.982	**0.991**	0.066	0.032	**0.014**	16000
fobber	0.997	0.998	0.999	0.005	0.003	**0.002**	16060
gozi	0.954	0.961	**0.993**	0.086	0.062	**0.024**	16000
kraken	**1.000**	**1.000**	**1.000**	0.001	0.000	0.001	16000
locky	0.983	0.984	**0.993**	0.009	0.015	**0.003**	16000
murofet	0.996	0.999	**1.000**	**0.000**	**0.002**	**0.000**	16313
necurs	0.966	0.968	**0.989**	0.040	0.012	**0.005**	16000
newgoz	**1.000**	**1.000**	**1.000**	0.000	0.000	**0.000**	16000
nymaim	0.934	0.941	**0.968**	0.051	0.086	0.044	16026

nymaim2	0.960	0.961	0.974	0.135	0.149	**0.051**	31670
padcrypt	**1.000**	**1.000**	**1.000**	**0.000**	**0.000**	**0.000**	16005
pizd	0.986	0.991	**0.999**	0.027	0.019	**0.003**	16000
proslikefan	0.922	0.947	**0.971**	0.145	0.033	**0.022**	16544
pykspa	0.953	0.954	**0.973**	0.049	0.047	**0.014**	16029
qadars	**1.000**	**1.000**	**1.000**	**0.000**	**0.000**	**0.000**	16714
qakbot	0.977	0.971	**0.991**	0.018	**0.010**	0.011	17000
ramdo	**1.000**	**1.000**	**1.000**	**0.000**	**0.000**	**0.000**	16000
ranbyus	0.992	0.991	0.998	0.006	0.016	0.004	16008
shiotob	0.986	0.986	0.994	0.005	0.004	0.003	16000
simda	0.996	0.996	**1.000**	0.007	0.009	**0.000**	16000
suppobox	0.987	0.986	**0.994**	0.027	0.007	0.013	16000
symmi	1.000	1.000	**1.000**	**0.000**	**0.000**	**0.000**	16000
tempedreve	0.965	0.964	**0.996**	0.065	0.018	**0.009**	16412
tinba	0.983	0.986	**0.999**	0.006	0.011	**0.001**	16404

The Bi-LSTM-Bg model test results in Tab. 4 show that 85.7% of the DGA family domain names have an FPR value of 1:100, which is the best among the three models and is superior to LSTM model .The ACC value of 85.7% DGA family domain name is 0.99, and 82.1% of them are the optimal results of the same DGA family domain name. In the paper of Woodbridge et al. [Woodbridge, Anderson, Ahuja et al. (2016)], we know that the model detection results in this paper are superior to the traditional bigram model and HMM (Hidden Markov Model). According to our experimental results, under sufficient training samples of DGA, the Bi-LSTM-Bg model can effectively detect suspicious DGA domain names, and the miss is far lower than other models tested in the experiments. The overall performance of the model is better than the model that proposed by Woodbridge et al. [Woodbridge, Anderson, Ahuja et al. (2016)].

However, in the actual detection, in addition to the normal domain names and the DGA domain names with known generation algorithm, the domain names to be detected may also contain unknown DGA domain names. How well does our model detect these unknown DGA domains?

We detect the domain names of five unknown DGA families, such as sisron, github_malware, javascript_malware, unknown_malware and vawtrak. It can be seen from Fig. 9 that the FPR value of Bi-LSTM-Bg model is much lower than that of SVM model by comparing the FPR results detected by each model. It shows that the Bi-LSTM-Bg has better ability to distinguish the unknown DGA family domains. Therefore, its ability to detect the unknown DGA family is much stronger than the traditional machine learning model.

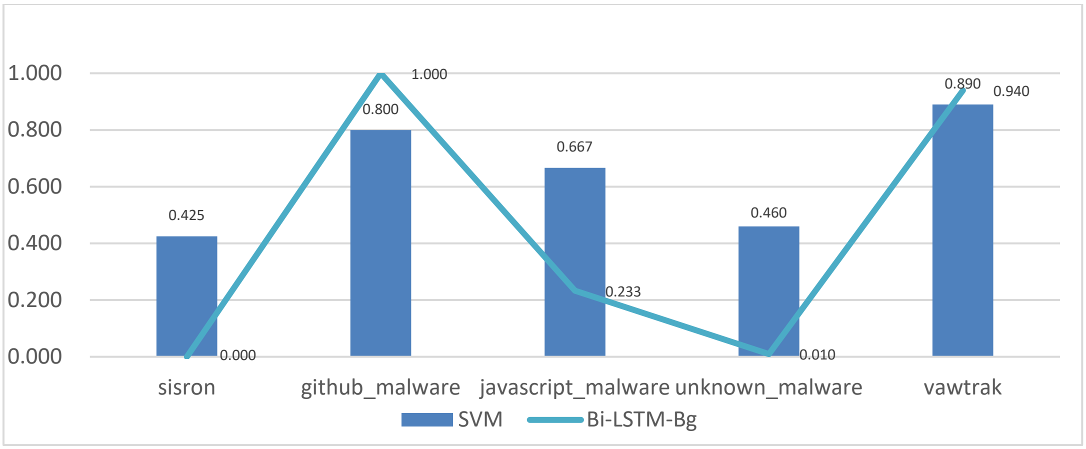

Figure 9: FPR comparison

In the experiment, other models are used to test the DGA domain name samples generated by some DGA domains without training, and the test results are shown in Tab. 5.

Table 5: FPR on the new datasets

DGA FAMILY	**FPR**			**NUM**
	LSTM	**Bi-LSTM-Ug**	**Bi-LSTM-Bg**	
sisron	**0.000**	**0.000**	**0.000**	**40**
github_malware	**0.000**	**0.000**	1.000	**120**
javascript_malware	0.050	0.033	0.233	**60**
unknown_malware	0.001	**0.000**	0.010	**100**
vawtrak	0.053	0.063	0.940	**300**
Total	0.032	0.034	0.687	**620**

As shown in Tab. 5, when we test the untrained DGA samples, we find that the misjudgment rates of the LSTM and Bi-LSTM-Ug models for the DGA are better than that of the Bi-LSTM-Bg models in the tests without training. Among them, some DGA family generate domain names with high detection rate in the four models, while there are a large number of domains with poor results in the optimal Bi-LSTM-Bg model detection, but achieve good results in the LSTM model.

There is no doubt that compared with the machine learning model, the neural network model has the unique ability in training new data. Therefore, we can combine the two optimal models, LSTM model and Bi-LSTM-Bg model, into a new model, Hybrid-LSTM. The experimental results are shown in Fig. 10. Each group of models was trained with the suspicious samples detected in this group, and the new domain data of the five families above were tested with the trained model. In the Hybrid-LSTM model, LSTM is used for detection in the early stage, and Bi-LSTM-Bg model is used for training and experimental detection and analysis of the detected suspicious samples. Num is the sample size of each DGA family. It can be seen from Fig.10 that Hybrid-LSTM model has the best effect among all the neural network models after a small amount of data training.

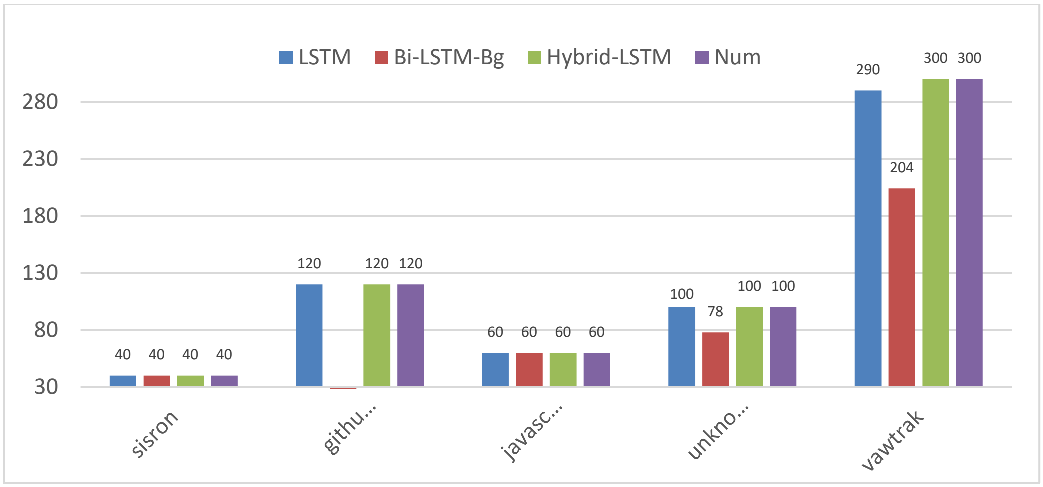

Figure 10: Detecting new DGAs comparison

4 Conclusion

This paper proposes a botnet's DGA detection model based on Bi-LSTM. To the best of our knowledge, this is the first time to deal domain name structure features with Bidirectional LSTM network. The model can extract features without the need of manual feature creation, and its trainable model can effectively deal with new unknown DGA family members. In addition, it only needs the domain name, without any additional context information. Hybrid-LSTM model is a kind of neural network model with high accuracy detection of known and unknown DGA families. Bi-LSTM-Bg model and LSTM model can be deployed on the detection system. For the domain name to be tested, if it is the domain name of the known DGA families, the Bi-LSTM-Bg model can ensure that the domain name of the most known DGA families is recognized. If it is the domain name of the unknown DGA families, the domain name can mostly be detected by the Hybrid-LSTM model. Since the botnets controlling process often lasts for hours or even days, when the suspect domain name is identified as the DGA domain name, training in the above models can ensure the effective identification of the DGA domain name afterwards.

Acknowledgement: This work was supported by the National Natural Science Foundation of China (NSFC) under the grant (No. U1836102).

References

Anderson, H. S.; Woodbridge, J.; Filar, B. (2016): DeepDGA: adversarially-tuned domain generation and detection. arXiv:1610.01969.

Antonakakis, M.; Perdisci, R.; Nadji, Y.; Vasiloglou, N.; Abunimeh, S. et al. (2012): From throw-away traffic to bots: detecting the rise of DGA-based malware. *Usenix Security Symposium*, pp. 491-506.

Chen, Y.; Yan, S.; Pang, T.; Chen, R. (2018): Detection of dga domains based on support vector machine. *Third International Conference on Security of Smart Cities, Industrial Control System and Communications*, pp. 1-4.

Cheng, R.; Xu, R.; Tang, X.; Sheng, V. S.; Cai, C. (2018): An abnormal network flow feature sequence prediction approach for ddos attacks detection in big data environment. *Computers, Materials & Continua*, vol. 55, no. 1, pp. 95-119.

Gers, F. A.; Schmidhuber, J.; Cummins, F. (2000): Learning to forget : continual prediction with lSTM. *Neural Computation*, vol. 12, no. 10, pp. 2451-2471.

Gers, F. A.; Schraudolph, N. N.; Schmidhuber, J. (2002): Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, vol. 3, no. 1, pp. 115-143.

Graves, A.; Schmidhuber, J. (2005): Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, vol. 18, no. 5, pp. 602-610.

Hampton, N.; Baig, Z. A. (2015): Ransomware: emergence of the cyber-extortion menace. *Proceedings of the13th Australian Information Security Management*.

Hochreiter, S.; Schmidhuber, J. (1996): LSTM can solve hard long time lag problems. *Advances in Neural Information Processing Systems*.

Hochreiter, S.; Schmidhuber, J. (1997): Long short-term memory. *Neural Computation*, vol. 9, no. 8, pp. 1735-1780.

Huang, J.; Wang, P.; Zang, T.; Qiang, Q.; Wang, Y. et al. (2018): Detecting domain generation algorithms with convolutional neural language models. *17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering.*

Kührer, M.; Rossow, C.; Holz, T. (2014): Paint it black: evaluating the effectiveness of malware blacklists. *International Workshop on Recent Advances in Intrusion Detection*.

Kwon, J.; Lee, J.; Lee, H.; Perrig, A. (2016): PsyBoG: a scalable botnet detection method for large-scale DNS traffic. *Computer Networks*, vol. 97, no. 1, pp. 48-73.

Lison, P.; Mavroeidis, V. (2017): Automatic detection of malware-generated domains with recurrent neural models. arXiv:1709.07102.

Liu, Y.; Sun, C.; Lin, L.; Wang, X. (2016): Learning natural language inference using bidirectional lstm model and inner-attention. arXiv:1605.09090.

Mahoney, M. (1999): Text compression as a test for artificial intelligence. *Sixteenth National Conference on Artificial Intelligence & the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*.

Mikolov, T.; Karafiat, M.; Burget, L.; Cernocký, J.; Khudanpur, S. (2010): Recurrent neural network based language model. *Conference of the International Speech Communication Association*, pp. 1045-1048.

Mowbray, M.; Hagen, J. (2014): Finding domain-generation algorithms by looking at length distribution. *IEEE International Symposium on Software Reliability Engineering Workshops*.

Robinson, A. J. (1994): An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 298-305.

Schiavoni, S.; Maggi, F.; Cavallaro, L.; Zanero, S. (2014): Phoenix: dga-based botnet tracking and intelligence. *International Conference on Detection of Intrusions & Malware*.

Schuster, M.; Paliwal, K. K. (1997): Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681.

Stonegross, B.; Cova, M.; Gilbert, B.; Kemmerer, R.; Kruegel, C. et al. (2011): Analysis of a botnet takeover. *IEEE Security & Privacy*, vol. 9, no. 1, pp. 64-72.

Wang, T.; Chen, L. C. (2017): Detecting algorithmically generated domains using data visualization and n-grams methods. *Proceedings of Student-Faculty Research Day, CSIS, Pace University*, pp. 1-4.

Wang, Z.; Jia, Z.; Zhang, B. (2018): A detection scheme for DGA domain names based on SVM. *International Conference on Mathematics, Modelling, Simulation and Algorithms*.

Wang, T. S.; Lin, C. S.; Lin, H. T. (2016): DGA botnet detection utilizing social network analysis. *International Symposium on Computer*.

Woodbridge, J.; Anderson, H. S.; Ahuja, A.; Grant, D. (2016): Predicting domain generation algorithms with long short-term memory networks. arXiv:1611.00791.

Yadav, S.; Reddy, A. K.; Reddy, A. N.; Ranjan, S. (2012): Detecting algorithmically generated domain-flux attacks with dns traffic analysis. *IEEE ACM Transactions on Networking*, vol. 20, no. 5, pp. 1663-1677.