A Detection Algorithm of Malicious Domain Based on Deep Learning and Multi-Head Attention Mechanism

Siqi Huang[1] , Bo Yan[1] , Dongmei Zhang[2]
[1]*School of Cyberspace Security, Beijing University of Posts and Telecommunication, Beijing.*
[2]*School of Computer Science, Beijing University of Posts and Telecommunication, Beijing,*

Keywords: Domain name detection, Multi-head attention mechanism, PCFG model.

Abstract: DGA (domain generation algorithms) domain names are a class of domain names generated by specific algorithms which are used to communicate with malicious C&C servers. DGA based on the PCFG model has been proposed Lately. Under the test of existing DGA detection techniques, its anti-detection effect is very obvious. One of the reasons is that these domains are generated by legal domain names and have the same statistical characteristics of legitimate domain names. This paper proposes a detection model which combines deep learning and Multi-head attention mechanism. It employs these two techniques to extract the features of the domain names. Experiment results show that the model has a good effect on detecting domain names based on PCFG model.

1 INTRODUCTION

Botnets have been problematic for over a decade and botmasters have developed techniques to evade detection. One of the widely used techniques is domain fluxing. Domain names are generated with DGA (Domain Generated Algorithms) and they are used as the 'rendezvous' points between botnets and hidden C&C (Command and Control) servers. In this way, botnets and botmasters establish communication. According to the relevant experiment, 23 of the 43 malware samples used the DGA to generate domain names (Plohmann, 2016). It is very important for security personnel to detect malicious domain name which are generated by DGA.

2 RELATED WORK

There are two main directions to detect algorithmically generated domain names (AGDs). The first one focuses on the distribution of domain name's structure and character features, while the other one focuses on group behaviours of botnet DNS requests generated by DGAs.

Yadav focused on the character distribution differences between AGDs and benign domain names. He used three metrics: Kullback-Leibler (KL) distance, Edit distance (ED) and Jaccard Index (JI) to detect AGDs (Yadav, 2010). Weiwei classified AGDs by analysing the morphemes in the domain name sequence (Weiwei, 2013). Mowbray analysed the character composition of the fixed domain name, and then used the machine learning model to detect AGDs (Mowbray, 2014).

In terms of traffic characteristics, Antonakakis used non-existent domain name traffic data to monitor randomly generated domain names (Antonakakis, 2012). Erquiaga used Markov model which is used to distinguish between normal traffic and DGA-based domain name traffic (Erquiaga, 2016). DNSRadar (X.ma, 2014) infers unknown malicious domains from the distribution of domain cache-footprints in the network. Wang et al. proposed BotMeter to assess the population distribution of DGA-based botnets by analyzing DNS query patterns (Wang, 2016). Wang et al. proposed a method DBod for assessing botnet traffic distribution through DNS query mode (Wang, 2017).

The current DGA detection methods can effectively detect existing AGDs, but they are not able to detect all the new emerging DGA. Two new DGAs are proposed by Yu Fu, one is based on Hidden Markov Model (HMM) and the other one is based on Probabilistic Context-Free Model (PCFG) (Yu Fu, 2017). The experiments in their paper showed that

Huang, S., Yan, B. and Zhang, D.
A Detection Algorithm of Malicious Domain Based on Deep Learning and Multi-Head Attention Mechanism.
DOI: 10.5220/0008098200840091
In *Proceedings of the International Conference on Advances in Computer Technology, Information Science and Communications (CTISC 2019)*, pages 84-91
ISBN: 978-989-758-357-5

under the detection of commercial DGA detection systems BotDigger (H. Zhang, 2016) and Pleiades, these two DGAs had better anti-detection effects than the existing DGAs.

This paper aims to detect one of the DGA which is based on the PCFG model. Using the neural network combined with the Multi-Head Attention mechanism, the model for detecting PCFG-based domain names is proposed.

3 ANALYSIS OF PCFG-BASED DOMAINS'S FEATURES

Yu Fu proposed a method for domain name generation using PCFG model. A context-free grammar (CFG) is a set of recursive production rules used to generate or recognize string patterns. It is represented as a tuple G = (N, Σ, R, S) where N is a set of nonterminal symbols, Σ represents a set of terminal symbols, S is a set of special starting symbols and R represents a set of production rules. A Probabilistic Context-Free Grammar (PCFG), uses a probability vector θ to assign a probability to each production rule in R. The grammar can be visualized with a parse tree, which is a finite tree regulated by grammar rules.

Based on the above concepts, Yu Fu's paper used PCFG model to generate domain names. The rules and the parsing tree used in the paper are shown in Figure 1.

Terminals	a, c
Non-terminals	S, B
Productions	$S \rightarrow BB$ $B \rightarrow aBc\,(p = 0.25)$ $B \rightarrow \varnothing\,(p = 0.75)$

Figure 1: Grammar and parse tree of PCFG model.

The terminators 'a' and 'c' are taken from the two disjoint sets A and C respectively. Yu Fu's paper also gives several alternative sources of A and C in Figure 2.

Source List	Source	Contain letters	Contain numbers	Pronoun-ceable	Example Output
pcfg_dict	English syllables	yes	no	yes	guisedswerv, daveFlorid
pcfg_dict_num	English syllables +number	yes	yes	yes	89chute, chooser3119
pcfg_ipv4	ipv4 syllables	yes	no	maybe	zenaidae, toowoom
pcfg_ipv4_num	ipv4 syllables +number	yes	yes	maybe	hrab-324, 1245ickb

Figure 2: Source of domain name composition of PCFG model.

(1) pcfg_dict: A syllable list generated from an English dictionary using hyphenation.

(2) pcfg_dict_num: The syllable list in pcfg_dict plus a number set;

(3) pcfg_ipv4: A syllable list parsed from IPv4 domain names;

(4) pcfg_ipv4_num: The collection comes from pcfg_ipv4 plus a non-alphabetic list;

The experiment results in Yu Fu's paper showed that if "a" and "c" are selected from the 'pcfg_ipv4_num' set, the generated domain names have the best anti-detection effect. Therefore, in this article we will focus on the detection of domain names which are composed of 'pcfg_ipv4_num'. After study, PCFG-based domain names have the following features:

1.Generated by legal domain name, the distribution of characters is very close to legal domain names.

Some previous detecting methods use relative entropy for analysis. The relative entropy of one single character is calculated as follows:

$$\overline{H} = \frac{H}{H_{max}} = \frac{-\sum_{i=1}^{n} p(x_i)\, log_2\, p(x_i)}{-log_2 \frac{1}{n}} \quad (1)$$

$p(x_i)$ represents the probability of character x_i occurrence, and n is the total number of the character types. Table 1 shows a comparison of the relative entropy of character distribution between PCFG-based domain names and other kinds of domain names:

Table 1: Relative entropy of different DGAs.

	1-gram	*2-gram*	example
legit	0.861	0.813	google; youtube
PCFG-based	0.857	0.864	hrab321; 1245ickb
corebot	0.999	0.952	at367lsnux1n1vg kl3xe0gf3md
kraken	0.995	0.938	xfdvisu; gopquidnxu
locky	0.999	0.947	evwjnhxh; xwkkckka
ramnit	0.999	0.926	Fhafkjiud; kxoggoma
banjori	0.93	0.916	ptmstring; umpfstring

As shown in Table 1, the PCFG-based domain names and legal domain names are very close in character distribution. This is because the PCFG-

based domain names are derived from legal domain names. The domains composed of splitted legal domain names have the same character distribution feature with the legal domain name without doubt.

2.Once the PCFG model is confirmed, what domain names will be generated only relate to the selection of source sets and the probability of every rule. It also means that one PCFG model can generate multiple types of domain names.

The domain name generated by the 'pcfg_ipv4_num' is composed of part of the legal domain name and numbers and the PCFG model for generating these domain names defines three rules. We find that the generated domain name can be described as "letter + number" finally. We can also try to generate other types of domain names by constructing other PCFG models (See Appendix):

Table 2: Domains generated by different PCFG models.

Model	Source collection	examples
Model1	pcfg_ipv4_num	hrabei7123; face01
Model2	pcfg_ipv4_num	china568; glove782
Model3	pcfg_ipv4_num	grass56; milk45
Model4	pcfg_ipv4_num	herb98beer; tariy890 76ricky78

As shown in Table 2, how the domain names are generated depends on the rules defined by the model and the probability distribution. If the rules are complex enough and the probability distribution is uniform, one PCFG model can generate multiple types of domain names. From the other side, we can see that the DGA based on the PCFG model is very scalable, which challenges us in finding the domain name by identifying features.

4 DETECTION MODEL

4.1 Multihead-Deep Detection Model

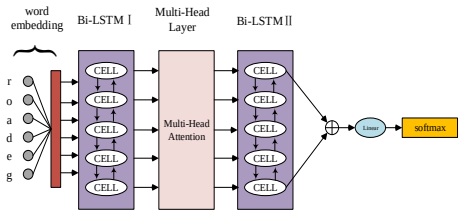

Figure 3: MultiHead-Deep model structure.

With the above analysis, it is extremely difficult to detect PCFG-based names by manually extracting features. The neural network provides us convenience that the output of the network layer can be considered as another expression of data, which can be considered as the feature extracted by neural network (Woodbridge J, 2016). Therefore, we adopted neural network in the construction of detection model.

In order to capture features better, the model uses the Multi-Head Attention mechanism (Ashish Vaswani, 2017). Multi-Head Attention is a leading technique in the field of natural language processing. Its usage scenarios include reading comprehension, abstractive summarization, textual entailment, etc. Multi-Head Attention uses the Self-Attention mechanism. Sometimes called intra-attention, it has the advantage of calculating the attention of current position and all the other positions, which can be used to compute a representation of the sequence.

Based on the above ideas, the structure of the entire model is shown in Figure 3. The Embedding layer is followed by the neural network layer, then is followed by the Multi-head Attention layer. After extracting the features from the first two layers, the output is sent to the subsequent BRNN layer, which combines the state of the last two units in the forward and backward directions respectively. Finally, this input will be passed to the closely followed SoftMax layer to get the final probability distribution after a linear transformation.

4.2 Multihead-Deep Detection Method

4.2.1 Pre-Processing

Before the detection, the domain sequence needs to be encoded in the Embedding layer. The input of neural network layer should be the word vector matrix of domain names, so we need to pre-process the domain name first in the Embedding layer. Pre-processing is divided into two steps:

(1) Dictionary Construction

Firstly, an index dictionary for each character is generated, in the form of (character: subscript). A mapping of characters to ids is constructed.

(2) Vector Coding

Each character is encoded as a vector which has the number of hidden units of the Multi-Head Attention Layer dimension. Then, a word vector matrix with dimensions [Sequence length, Number of hidden dims of Multi-Head Attention Layer] will be obtained, which will be trained along with the network. The word vector corresponding to each character can be found by the index of the previous step.

4.2.2 Network Processing

Neural Network Layer. After the processing of Embedding layer, we input the vector sequence into the neural network layer for preliminary extraction of features. At this layer, we construct FNN, CNN, and RNN as the main part of the neural network layer (Zhixing Tan, 2018).

(1) FFN（feedforward neural network）

Feedforward neural network, also known as multilayer perceptron (MLP). It is the simplest neural network in which neurons are arranged hierarchically. Each neuron is only connected to the neurons of the previous layer. It receives the output of the previous layer and passes it to the next layer. There is no feedback between the layers.

If we adopt the FFN, the formula for this layer is:

$$\boldsymbol{FFN(X) = ReLU(XW_1)W_2} \quad (2)$$

(2) CNN（Convolutional neural network）

Generally, the basic structure of CNN includes two layers, one is feature extraction layer. The input of each neuron is connected to the part of the previous layer, then the features of that part are extracted. The second layer is the feature mapping layer, each computing layer of the network is composed of multiple feature mappings, and each feature mapping is a plane. All neurons in the plane have equal weights. Each convolutional layer in the CNN is followed by a computational layer for local averaging and quadratic extraction. This unique two-feature extraction structure reduces feature resolution.

If we adopt CNN, our formula will be:

$$S(\boldsymbol{i,j}) = (\boldsymbol{X} * \boldsymbol{W})(\boldsymbol{i,j}) + \boldsymbol{b} \quad (3)$$

(3) RNN（Recurrent neural network）

Bi-LSTM is adopted in this experiment. It is composed of a forward LSTM and a backward LSTM.

The LSTM unit controls, discards or adds information through a "gate" to enable forgetting or memorizing. A "gate" is a structure that selectively passes information. It consists of a sigmoid function and an element-level multiplication operation. An LSTM unit has three such gates: a forget gate, an input gate, and an output gate. Such a unit design allows the neural network to store access states over long sequences, thereby mitigating gradient disappearance issues.

Bi-LSTM is used to capturing bidirectional semantic dependence and we adopt the following formula:

$$\overrightarrow{\boldsymbol{h_t}} = \boldsymbol{LSTM}(\boldsymbol{E_t}, \overrightarrow{\boldsymbol{h_{t-1}}}) \quad (4)$$

$$\overleftarrow{\boldsymbol{h_t}} = \boldsymbol{LSTM}(\boldsymbol{E_t}, \overleftarrow{\boldsymbol{h_{t-1}}}) \quad (5)$$

$$\boldsymbol{y_t} = \overrightarrow{\boldsymbol{h_t}} + \overleftarrow{\boldsymbol{h_t}} \quad (6)$$

E_t represents the encoded vector.

Multi-Head Attention Layer. The Multi-Head Attention layer is responsible for receiving the output of the previous neural network layer. This layer obtains the feature information of each position through the calculation of Multi-Head Attention.

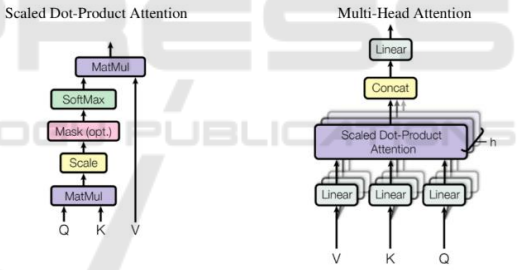

Figure 4: Structure of Multi-Head Attention.

The basic structure of Multi-Head Attention is shown in Figure 4. The Scaled Dot-Product Attention at the center is a variant of the general Attention, given the matrix $Q \in R^{n*d}$, $K \in R^{n*d}$, $V \in R^{n*d}$, Scaled Dot-Product Attention can be used to calculate the Attention score by the following formula:

$$\boldsymbol{Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d}})V} \quad (7)$$

"d" represents the number of neural network hidden units.

In the MultiHead-Deep model, Multi-Head Attention first needs to linearly transform the input vectors of Q, K, and V, then calculate them. "Multi-Head" means the calculation in the Scaled Dot-Product Attention section needs to be performed

multiple times. The "head" number means the number of calculations, but the linear projection of Q, K, and V is different for each head's calculation. Take the i-th head as an example:

$$Q' = Q * W_i^Q \quad (8)$$

$$K' = K * W_i^K \quad (9)$$

$$V' = V * W_i^V \quad (10)$$

Since this layer receives the output of the neural network, therefore:

$$Q = K = V = y_t \quad (11)$$

Finally, the result of this head is:

$$M_i = softmax(\frac{Q'K'^T}{\sqrt{d}})V' \quad (12)$$

After h operations, we splice these M_i:

$$M = Concat(M_1, M_2, \dots, M_h) \quad (13)$$

MultiHead-Deep does not directly transform the stitched result in this layer, but it sends the output of this layer to the next layer for subsequent processing.

BRNN Layer. The processing of this layer is simple. After receiving the output of the Multi-Head Attention layer, the model will combine the state of the last two units in the forward and backward directions, As follows:

$$S_t = \overrightarrow{\in}_t + \overleftarrow{\in}_t \quad (14)$$

Soft Max Layer. The output of the BRNN Layer will be linear transformed, then the result will be processed by the SoftMax layer to obtain the final probability distribution. We define the loss function as the cross-entropy loss function:

$$E = -\sum_{j=1}^{T} y_i \log P_j \quad (15)$$

5 EXPERIMENT

5.1 Experiment Design

This experiment intends to use the data set to find a better network model through comparison experiments and test the accuracy of the MultiHead-Deep model for PCFG-based malicious domain names. For this purpose, we have prepared two sets of comparative experiments:

(1) Using different neural network models (RNN, CNN and FFN) to detect the detection effect of MultiHead-Deep against PCFG-based malicious domain names under different models;

(2) Using a conventional DGA detection algorithm to detect PCFG-based malicious domain names, and comparing the results with the MultiHead-Deep model;

5.2 Experiment Procedure

5.2.1 Data Set Preparation

In this paper, the data set is prepared in the way given in Yu Fu's paper. Considering the actual situation, the domain names generated by the PCFG model are second-level domain names.

The experiment crawl top 100,000 legal domain names from Alexa, and the domain names are used to regenerate PCFG-based domains according to the algorithm given in the Yu Fu's paper.

60% of the domain name collection (a mixture of legal domain names and PCFG-based domain names) will be used as a training set, 20% as a test set and the left 20% as a verification set.

5.2.2 Parameter Settings

The parameters that need to be determined when the model is initialized.

Table 3 Initialization parameters of MultiHead-Deep model.

Parameter index	value
Number of Multi-Head Attention hidden Units	288
Heads of Multi-Head Attention	8
Learning rate	0.001
Gradient	5.0
Dropout rate	0.5
Epoch number	10

5.3 Experiment Result

Experiment one: Compare MultiHead-Deep's detection of PCFG-based domain names using different neural network models.

Table 4: Results of MultiHead-Deep's detection using different network models detecting Model 1.

Model1	*Recall*/%	*Precision*/%	*F1*/%
MultiHead-CNN	96.13	89.78	92.85
MultiHead-FFN	86.48	84.25	85.35
MultiHead-LSTM	92.48	91.25	91.86

Table 5: Results of MultiHead-Deep's detection using different network models detecting Model 2.

Model2	*Recall*/%	*Precision*/%	*F1*/%
MultiHead-CNN	96.29	91.27	93.71
MultiHead-FFN	87.26	87.74	88.99
MultiHead-LSTM	91.73	93.86	92.70

Table 6: Results of MultiHead-Deep's detection using different network models detecting Model 3.

Model3	*Recall*/%	*Precision*/%	*F1*/%
MultiHead-CNN	95.78	89.67	92.62
MultiHead-FFN	85.37	84.91	85.13
MultiHead-LSTM	90.41	90.93	90.67

Table 7: Results of MultiHead-Deep's detection using different network models detecting Model 4.

Model4	*Recall*%	*Precision*/%	*F1*/%
MultiHead-CNN	95.14	91.13	93.09
MultiHead-FFN	87.29	87.01	87.14
MultiHead-LSTM	91.79	91.99	91.62

The results show that when detecting the same PCFG model, we can get the conclusion that MultiHead-CNN > MultiHead-LSTM > MultiHead-FFN. It can be seen that in the experimental environment, CNN is a network model that is more suitable for detecting PCFG-based malicious domain names. Although the overall performance of MultiHead-CNN is more excellent, the difference between MultiHead-CNN and MultiHead-LSTM is very small. The disparity is no more than 2.1%.

Experiment two: Compare detection methods for different PCFG-based domain names.

Table 8: Results of different methods detecting Model 1.

Model1	*Recall*/%	*Precision*/%	*F1*/%
KL	22.70	22.20	22.35
ED	43.50	42.70	43.09
JI	51.90	51.15	51.52
MultiHead-CNN	96.13	89.78	92.85

Table 9: Results of different methods detecting Model 2.

Model2	*Recall*/%	*Precision*/%	*F1*/%
KL	22.70	22.20	22.35
ED	43.50	42.70	43.09
JI	51.90	51.15	51.52
MultiHead-CNN	96.29	91.27	93.71

Table 10: Results of different methods detecting Model 3.

Model3	*Recall*/%	*Precision*/%	*F1*/%
KL	22.70	22.20	22.35
ED	43.50	42.70	43.09
JI	51.90	51.15	51.52
MultiHead-CNN	95.78	89.67	92.62

Table 11: Results of different methods detecting Model 4.

Model4	*Recall*/%	*Precision*/%	*F1*/%
KL	22.70	22.20	22.35
ED	43.50	42.70	43.09
JI	51.90	51.15	51.52
MultiHead-CNN	95.14	91.13	93.09

In this experiment, MultiHead-Deep uses CNN as a network model for detection. When these four methods detect the same PCFG model, we can get the conclusion that MultiHead-CNN > Jaccard Index > Edit Distance > KL Divergence. It can be seen that the detection effect of the MultiHead-CNN model on the PCFG-based domain name is indeed better than the traditional method. From the perspective of the model and in the case of "detected" rate, we have model 2 > model 4 > model 1 > model 3. It can be seen that the detection effect of MultiHead-CNN is also affected by different PCFG models.

6 EXPERIMENT ANALYSIS

6.1 Feature Extraction Analysis

Traditional methods focus mainly on the random distribution of characters. Such as Zeus, Kraken and other DGA, the distribution characteristics of characters and the legal domain name are very different. Therefore, we can use KL divergence, Editing Distance and Jaccard Index to get better results. However, the PCFG-based domain names are assembled with the character part and the digital part from the legal domain names, supplemented by some hyphens. In terms of construction, it is exactly the same as the legal domain name. Even if the 'source collection' is large enough, it is possible to generate a domain name that is identical to the legal domain name. In the process of manually generating PCFG-based domain names, we found that with the increasing complexity of the PCFG model, domain names can be iteratively nested, and they show a certain partial regularity. However, using character features to distinguish between legitimate domain names and PCFG-based domain names is still very difficult, because the same feature is difficult to

completely "measure" different types of domain names generated by the same model.

The results of the above experiments show that the detection effect of CNN and Bi-LSTM network is more prominent, while the effect of FFN is relatively poor. Comparing the Bi-LSTM network with the FFN, the Bi-LSTM can synthesize the forward and backward information, and it solves the long-distance dependence problem in the ordinary RNN network and can memorize historical information. The FFN cannot learn the surrounding "context" because the data fed to the FFN is not related to the previous data, and it cannot remember the previous context information. Therefore, the features captured by the LSTM network are more detailed.

The effect of CNN is even better. After the domain name is vectorized, a domain name can be regarded as a vector matrix, which is very similar to the image processing of CNN. Every time in convolution, CNN processes the data in whole line. It's like an n-gram model. If you process every two rows, it will be a 2-gram model. At the same time, since multiple convolution kernels can be set in the model to capture different features, it has stronger feature extraction capability than FFN.

The results prove that the effect of using CNN on PCFG-based domain name detection is slightly better than that of LSTM. In the short-sentence task, CNN has an overall ability to summarize the overall structure of the sentence because of its convolution function; but in the long-sentence task, CNN can only process the information in its window. The information of adjacent windows can only be supported by the latter convolutional layer, which depends heavily on the parameters of the convolution window and the length of the movement. The domain name of this mission is not particularly long, so using the CNN model is a suitable solution.

6.2 Multi-Head Attention Layer Analysis

The MultiHead-Deep model not only uses the neural network as a way of feature extraction, but also adopts Multi-Head Attention layer, which is also the key to improving the detection effect. Multi-Head Attention employs self-attention mechanism. Its advantage is that it can capture the global connection in one step and completely solve the long-distance dependence problem. In addition, Multi-Head computing can be considered as learning in a number of different subspaces, integrating information in different subspaces to capture the features of each location as much as possible.

Figure 5 shows the accuracy curve of the MultiHead-CNN model. Under different PCFG models, the accuracy convergence value of MultiHead-CNN is different. This is because that the detection complexity of domain names generated by different models is different.

Figure 5 also shows that the MultiHead-CNN model converges faster, and MultiHead-CNN converges before the fifth iteration in different models. It is proved that the model extraction feature is excellent from the other side.

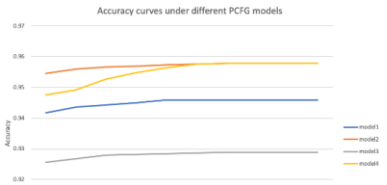

Figure 5: Accuracy trends of different PCFG model.

7 CONCLUSIONS

This paper proposes a model for detecting PCFG-based domain names using neural networks and Multi-Head Attention mechanism. Experiments show that the MultiHead-Deep model is better than traditional detection methods in detecting such DGA domain names. This model takes advantage of the neural network that does not require manual capture of features and the intrinsic link of domain names. It uses the Multi-Head Attention mechanism to capture the overall features more deeply. Different PCFG models have different detection rates, which also indicates that the PCFG mode can be extended according to the rules established by the user. In the experiment, MultiHead-Deep has shown decent results on different PCFG models, which proves the effectiveness of the model in PCFG-based domain name detection

REFERENCES

Plohmann D, Yakdan K, Klatt M, A comprehensive measurement study of domain generating malware[C]// *25th USENIX Security Symposium. Austin: Usenix*, 2016:263-278.

S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, *"Detecting algorithmically generated malicious domain names,"* in Proc. 10th ACM SIGCOMM Conf. Internet Meas., 2010, pp. 48–61.

Z. Wei-Wei and G. J. L. Qian, *"Detecting machine generated domain names based on morpheme features,"* in Proc. Int. Workshop Cloud Comput. Inf. Secur., 2013, pp. 408–411.

M. Mowbray and J. Hagen, *"Finding domain-generation algorithms by looking at length distribution,"* in Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW), Nov. 2014, pp. 395–400.

M. Antonakakis et al., *"From throw-away traffic to bots: Detecting the rise of DGA-based malware,"* in Proc. USENIX Secur. Symp., 2012, pp. 491–506.

Maria Jose Erquiaga, *"Detecting DGA Malware Traffic Through Behavioral Models "* IEEE, 2016-6

X. Ma, J. Zhang, J. Tao, J. Li, J. Tian, and X. Guan, "DNSRadar: Outsourcing malicious domain detection based on distributed cache- footprints," IEEE Trans. Inf. Forensics Security, vol. 9, no. 11, pp. 1906–1921, Nov. 2014.

T. Wang, X. Hu, J. Jang, S. Ji, M. Stoecklin, and T. Taylor, *"BotMeter: Charting DGA-botnet landscapes in large networks,"* in Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst. (ICDCS), Jun. 2016, pp. 334–343.

T.-S. Wang, H.-T. Lin, W.-T. Cheng, and C.-Y. Chen, "DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis," Comput. Secur., vol. 64, pp. 1–15, Jan. 2017.

Yu Fu, Lu Yu, Oluwakemi Hambolu, Ilker Ozcelik, Benafsh Husain, *"Stealthy Domain Generation Algorithms"* IEEE Transactions on Information Forensics and Security (Volume: 12 , Issue: 6, June 2017)

H. Zhang, M. Gharaibeh, S. Thanasoulas, and C. Papadopou- los, "BotDigger: Detecting DGA bots in a single network," in Proc. IEEE Int. Workshop Traffic Monitor. Anal., Louvain La Neuve, Belgium, Apr. 2016, pp. 16–21. [Online]. Available: http://www.cs.colostate.edu/hanzhang/papers/BotDigger-TMA16.pdf

Ashish Vaswani, Noam Shazeer, Niki Parmar, *"Attention is all your need"*, Computation and Language (cs.CL); Machine Learning (cs.LG), arXiv:1706.03762

Woodbridge J, Anderson H S, Ahuja A, et al. *Predic ting domain generation algorithms with long short-term memory networks*[EB/OL]. arXiv, 2016-11-2[1018-3 -10]. https://arxiv.org/abs/1611.00791.

Zhixing Tan, Mingxuan Wang, Jun Xie "Deep Semantic Role Labeling with Self-Attention" AAAI-2018, arXiv:1712.01586 [cs.CL]

APPENDIX:

Different PCFG Models:

Model 1：

Terminals	a, c
Non-terminals	S, B
Productions	S -> BB B -> a B c (p = 0.25) B -> Ø (p = 0.75)

Model 2：

Terminals	a, c
Non-terminals	S, B
Productions	S -> BB B-> a B c (p = 0.5) B -> Ø (p = 0.5)

Model 3：

Terminals	a, c
Non-terminals	S, B
Productions	S -> BB B-> a c (p = 0.25) B -> Ø (p = 0.75)

Model 4：

Terminals	a, c
Non-terminals	S, A, B, C
Productions	S -> A B C A -> a B -> c C -> a (p = 0.5) C -> Ø (p = 0.5)