CNN-based DGA Detection with High Coverage

Shaofang Zhou
College of Computer Science and Technology
Zhejiang University
Hangzhou, China
zhoushaofang@gmail.com

Lanfen Lin
College of Computer Science and Technology
Zhejiang University
Hangzhou, China
llf@zju.edu.cn

Junkun Yuan
Research Center of Network Big Data in Health Care
Zhejiang Lab
Hangzhou, China
scottyuan163@163.com

Feng Wang
Department of Computer and Information Technology
Zhejiang Police College
Hangzhou, China
wangfeng@zju.edu.cn

Zhaoting Ling
College of Computer Science and Technology
Zhejiang University
Hangzhou, China
lingzhaoting@outlook.com

Jia Cui
China Information Technology Security Evaluation Center
Beijing, China
cuij@itsec.gov.cn

***Abstract*—Attackers often use domain generation algorithms (DGAs) to create various kinds of pseudorandom domains dynamically and select a part of them to connect with command and control servers, therefore it is important to automatically detect the algorithmically generated domains (AGDs). AGDs can be broken down into two categories: character-based domains and wordlist-based domains. Recently, methods based on machine learning and deep learning have been widely explored. However, much of the previous work perform well in detecting one kind of DGA families but poorly in classifying another kind. A general detection system which is applicable to both kinds of domains still remains a challenge. To address this problem, we propose a novel real-time detection method with high accuracy as well as high coverage. We first convey a domain name into a sequence of word-level or character-level components, then design a deep neural network based on temporal convolutional network to extract the implicit pattern and classify the domain into two or more categories. Our experimental results demonstrate that our model outperforms state-of-the-art approaches in both binary classification and multi-class classification, and shows a good performance in detecting different kinds of DGAs. Besides, the high training efficiency of our model makes it adjust to new malicious domains quickly.**

***Index Terms*—domain generation algorithm, malicious domain names, deep learning, convolutional neural network**

I. Introduction

Malware needs to control the infected devices to connect with command and control (C&C) servers. Traditional method used by the attackers is hardcoding a list of static domains or IP addresses in the malware code, which makes it easy for the defenders to cut all the links once a malware was found and reverse engineered. Thus, newer families of malware begin to use domain generation algorithms (DGAs) to circumvent network defenses. Attackers can generate a very large number of pseudorandom domains dynamically via various kinds of DGAs using different generation schemes, including: (1) arithmetic-based: calculating a sequence of alphanumeric codes, (2) hash-based: using the hexadecimal representation of a hash, (3) wordlist-based: concatenating several words, (4) permutation-based: producing different permutations of an initial domain name [1]. Blacklists and sinkhole become inefficient due to the low coverage. It is important to automatically detect the algorithmically generated domains (AGDs).

Recently, many studies have been addressed the problem of DGA detection. There are two kinds of methods: retrospective detection and real-time detection. Retrospective detection methods take batches of domains as input and cluster similar domains according to lexical features and/or other contextual information before classification. Such methods are usually slow and cannot detect the ADGs before the communication have been established, thus they cannot be used for real-time detection and prevention.

On the other hand, real-time detection methods can classify malicious domains with benign domains solely on the basis of a single domain name. Early real-time approaches are mainly based on traditional machine learning or statistical learning, having the limitation of the dependence on the hand-craft features. To accomplish feature-less real-time detection, Woodbridge et al. [2] proposed a method utilizing Long Short-Term Memory (LSTM) network, which is the first work applying deep learning to this problem. Subsequently, methods based on different deep learning models including recurrent neural networks (RNNs) and convolutional neural networks (CNNs) have been widely explored and show significantly great performance in detection, except for the domains generated by wordlist-based DGAs [3]–[6]. We infer the reason might be that the wordlist-based DGAs generate malicious domains which have similar character distribution with benign domains, while others, we can collectively call them character-based DGAs, mostly produce random-looking domains.

Currently, several researchers turn their attention to the problem of detecting wordlist-based AGDs, coming up with some solutions such as: using random forest with hand-craft features [7], building word graph in domains [8], designing a score to measure the similarity between domains and English words [9], and utilizing context-sensitive word embedding [10]. They focus on wordlist-based DGAs detection and have a good performance, but increase the computational complexity or decrease the accuracy when detecting character-based DGAs. Therefore, how to detect AGDs in real-time with high coverage still remains a challenge.

In real world, the detection system needs to identify different kinds of AGDs hiding in the benign domains, so a general classifier has higher application value. On the other hand, there is an adversarial relationship between attackers and defenders. Attackers keep on devising new algorithms to evade detection. A feasible way for defenders is to retrain the model using the latest data. Thus it is important to increase the training efficiency so that the model can adapt to new DGA families as quickly as possible.

In this paper, we proposed a novel method of real-time detection for both character-based AGDs and wordlist-based AGDs. The main idea behind our approach is that we utilize a convolutional neural network (CNN) architecture to extract the implicit pattern of component strings in a domain. The component string might be single letter, number, other punctuation or meaningful word. The major advantages of our approach are as following:

- It provides a universal classifier which can distinguish various kinds of AGDs with benign domains. Especially, it gets great result in detection for wordlist-based DGAs.
- It achieves better performance than the state-of-the-art methods in both binary classification task and multi-class classification task.
- It has high training efficiency, thus it takes less time to retrain. As a result, our classifier can adjust to new malicious domains quickly.

The remaining part of the paper is structured as follows: In Section II, we introduce related work. In Section III, we describe the details of the methodology used for this study. In Section IV and Section V, we show the experimental studies for evaluating the proposed model. In Section VI, we summarize our work and discuss some future work.

II. Related Work

Over past few years, a great deal of previous research has focused on automatically detection of algorithmically-generated domains. Yadav et al. [11], [12] proposed the methodology through the statistical analysis of the unigram and bigram distribution of grouped domains. The detection system Pleiades designed by Antonakakis et al. [13] found DGA by clustering similar Non-Existent Domains (NXDomains) according to statistical characteristics and query hosts, then used a supervised learning approach to identify the DGA family. Similarly, Zhou et al. [14] captured NXDomain traffic and grouped domains via calculating the similarity of live time span and visit pattern. The problem of these retrospective methods is that they all need batches of domains to make a classification.

Later, some researchers attempted to classify based on single domain. Krishnan et al. [15] presented an algorithm based on sequential hypothesis testing, which extracted the NXDomain traffic patterns of individual hosts. Study undertaken by Raghuram et al. [16] indicated that algorithmically generated domain names and human-created domain names typically had different distributions, thus a probability model of character sequences was built for classification. Mowbray et al. [17] discovered DGAs from Domain Name Service (DNS) query data based on the distribution of second-level string lengths. The Phoenix proposed by Schiavoni et al. [18] distinguish DGAs and non-DGAs using a combination of string and IP-based features. The limitation of these approaches is the requirement of contextual information or hand-crafted features.

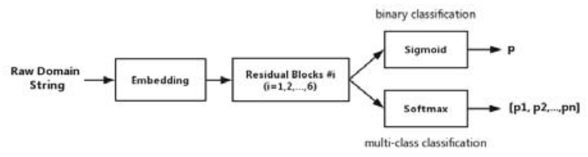

Fig. 1. The overall network architecture of our method.

Recently, some deep learning methods had been applied to DGA detection. The first work was conducted by Woodbridge et al. [2] which employed a LSTM network and proved that the neural network could carry out feature-less and real-time detection with high accuracy. Subsequently, Lison et al. [3] evaluated the performance of different designs of RNN on a larger dataset. Yu et al. [4] proposed a novel technology to collect data from real traffic, and used CNN and RNN to make a classification. Saxe et al. [5] explored the capacity of CNN in cybersecurity problems and designed a parallel CNN architecture. Besides, there were a lot of studies such as [6], [19]–[22] comparing the performance of different DGA classifiers, which further demonstrated that deep learning approaches significantly outperform traditional methods.

However, methods mentioned above still underperformed at detecting wordlist-based DGAs. To overcome this problem, Yang et al. [7] concluded 16 features through analysis and used a random forest as classifier. Pereira et al. [8] proposed a graph-based approach called *WordGraph* to learn the dictionary used by DGA. Curtin et al. [9] devised *smashword* score to measure the similarity between a DGA family and English words, then used a novel RNN architecture combined with domain registration side information for detection. Koh et al. [10] utilized Embeddings from Language Models (ELMo), a context-sensitive word embedding, to learn the semantic signatures of the wordlist-based DGA families.

III. Method

We propose a novel method for the DGA detection problem with special emphasis on high coverage and high efficiency. In order to adapt to different kinds of AGDs including character-based and wordlist-based domains, we first convey a domain name into a sequence of word-level or character-level components, which might be single letter, number, other punctuation or meaningful word. Empirical evaluation demonstrated that CNN architecture could be applied to sequence modeling task and outperformed baseline RNN architecture [23]. Inspired by this work, we adjust temporal convolutional network (TCN) to DGA detection problem due to its good performance and high efficiency. We design a CNN-based deep neural network architecture to extract the implicit pattern of component strings in a domain and classify the domain into two or more categories.

The overall network architecture is illustrated in Fig. 1. Our model consists of a embedding layer, six stacked residual blocks and a full-connected layer. In binary classification, the activation function in last layer is sigmoid, while in multi-class classification, the activation function is softmax.

A. Embedding

The input domains are fully qualified domain names (FQDNs). Each FQDN consists of a list of strings, technically called labels, separated by a dot. Labels going from right to left are called the top-level domain (TLD), the second-level domain, the third-level domain and so on, respectively. For example, the domain *wikipedia.org* is composed of the TLD *com* and the second-level domain *wikipedia*. The valid TLDs are prescribed, but other labels are open for reservation by end-users.

The embedding module aim at dividing the domain name into a sequence of components, then covert each component into an corresponding embedding vector. Firstly, we segment the domain name into different labels. Secondly, we further divide the separated domain labels into word-level components. Thirdly, we check each component to see whether it can be further separated into character-level components. Finally, the component strings are passed through a embedding layer.

It is easy to separate labels in domains since they are delimited by dots. However, how to parse out words from domain labels is a more difficult problem corresponding to word segmentation. One solution is to find the optimum segmentation by means of calculating word probabilities of each substrings based on word frequency dictionary. There have been a wealth of useful tools based on such method and amongst them we choose *WordSegment*. Besides, on purpose of getting better segmentation, we assume that if a substring is not in the dictionary, it is not a common meaningful word and need to be divided into single characters.

As an example, the second-level domain *microsoftonline* of *microsoftonline.com* can be divided into *microsoft* and *online*, which are both meaningful words. So the list of components is *[microsoft, online]*. Give another example, when the input is *weaqovir*, the optimum result through word segmentation is *weaqovir*, which is not included in the frequency dictionary, so it need be further split up and the list of components is *[w, e, a, q, o, v, i, r]*.

After segmentation, the input domain is converted into a sequence of component strings $[s_1, s_2, \ldots, s_n]$. Then we map

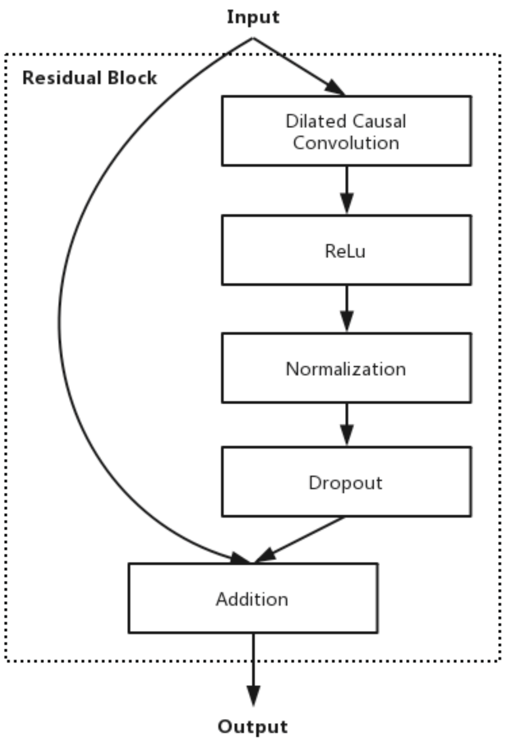

Fig. 2. The structure of residual block.

strings to integers through hash algorithm and get the vector $[i_1, i_2, \ldots, i_n]$. The last step is to embedded the vector into $n \times m$ floating point matrix, each row of which is a embedding m-dimension vector. In our network, we set m to 128.

B. Residual Blocks

Once the domains are embedded into matrix, we use six similar stacked residual blocks to extract and aggregate features. The structure of residual block is showed in Fig. 2.

The first and most important layer of residual block is the dilated causal convolutional layer, which is more suitable for handling sequence data than plain convolutional layer. As we known, the discrete operator $*$ between signal f and kernel k is defined as:

$$(f * k)(t) = \sum_{\tau=-\infty}^{+\infty} k(\tau) * f(t-\tau) \qquad (1)$$

While the dilated causal convolution operator, denoted by $*_l$, is defined as:

$$(f *_l k)(t) = \sum_{\tau=0}^{+\infty} k(\tau) * f(t - d \cdot \tau) \qquad (2)$$

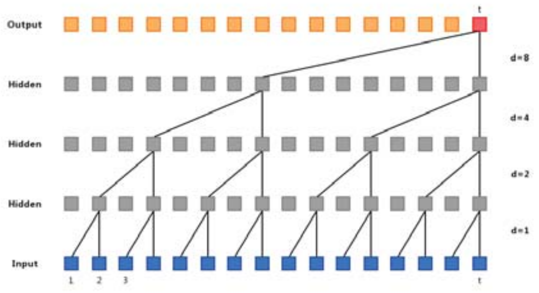

Fig. 3. An example of four dilated causal convolutional layers, where kernel size is 2 and dilation factor d is initialized to 1 then increases exponentially. The output at t is calculated by 16 elements of the input before t.

The major differences includes: (1) The dilation factor d is introduced, as a result, the elements of signal involved in the calculation take a step by d. It helps efficient enlargement of the receptive field. (2) The output at time step t calculates only with the elements at or before t in the input. Thus, it avoids leakage of future information when generating data in the past.

We follow the common way when using dilated convolution, i.e., to make the dilation factor increase exponentially at each convolutional layer. Fig. 3 shows an example of four layers. After analyzing the maximum length of domains, we eventually use six residual blocks so that the receptive field is large enough. For the residual block $i (i = 1, 2, \dots, 6)$, the dilation factor d is assigned to 2^{i-1}, i.e., $d = 1$ in the first block, $d = 2$ in the second block, $d = 4$ in the third block, and so on.

Following the dilated causal convolutional layer, there are a rectified linear unit (ReLU) layer, a normalization layer and a dropout layer.

Besides, we apply residual connection, which means that the result of the dropout layer is added to the input of the residual block. It can further increase model capacity and make the training easier.

C. Full-connected Layer

Finally, a full-connected layer takes the implicit extracted feature vector as input and make a classification.

In this paper, we focus on two task of classification: (1) binary classification, i.e., determine whether the domain is generated by DGAs or not, (2) multi-class classification, i.e., determine which DGA family the domain belongs to.

For binary classification, we use sigmoid activation function and output a real value $p \in [0, 1]$, which indicates the possibility that the domain is malicious. While for multi-class classification, we use softmax activation function and output a vector $[p_1, p_2, \dots, p_n]$, where n is the number of classes and $p_i (i = 0, 1, \dots, n)$ indicates the possibility that the domain belong to class i.

TABLE I
A PART OF DGA DATA

DGA Kind	DGA Family	Number Of Domains
Arithmetic-based	Banjori	400,000
Hash-based	Bamital	271,128
Permutation-based	Volatile Cedar	498
Wordlist-based	Matsnu	81,297

IV. EXPERIMENTAL SETUP

In this section, we introduce our experimental setup, including the datasets, the metrics and the baseline models.

A. Datasets

The data for experiments contains benign domains and malicious domains. We collect them from two source: (1) For benign domains, we download Alexa top 1 million domains[1]. (2) For malicious domains, we download the full data until the end of 2017 from DGArchive.[2]

To decrease the impact of class imbalance, we ignore the DGA families whose domains are fewer than 100. In addition, if domains of a DGA family are more than 400,000, we random select 400,000 domains. Eventually, the experimental dataset contains 73 DGA families and one special benign family, total 10,817,465 domains. Four kinds of DGAs are all included in the data. It is a very large-scale and diverse dataset and has similar distribution with data in the real word. TABLE I shows the details of some representative DGA families.

B. Metrics

The following metrics are used to evaluate the performance of models:

- Precision. It is defined as in:

$$Precision = \frac{\sum TruePositive}{\sum TruePositive + \sum FalsePositive} \quad (3)$$

- Recall. It is defined as in:

$$Recall = \frac{\sum TruePositive}{\sum TruePositive + \sum FalseNegative} \quad (4)$$

- F_1 score. It is defined as in:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (5)$$

- Area under the curve (AUC). It is the area under receiver operating characteristic (ROC) curves. To plot ROC curves, we first calculate the true positive rate (TPR) which is defined as in (6) and false positive rate (FPR) which is defined as in (7), then plot TPR and FPR at various thresholds.

$$TPR = \frac{\sum TruePositive}{\sum TruePositive + \sum FalseNegative} \quad (6)$$

$$FPR = \frac{\sum FalsePositive}{\sum FalsePositive + \sum TrueNegative} \quad (7)$$

[1]http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
[2]https://dgarchive.caad.fkie.fraunhofer.de/

For multi-class classification, the results of different classes must be averaged. We consider two average types:

- Macro average. It calculates the unweighted mean of each class. The class imbalance will not be taken into account.
- Weighted average. It calculates metrics for each class, then average the scores of classes weighted by the number of related class, which considers class imbalance.

C. Baseline Models

Empirical comparison between different character-level DGA detection models shows that there is little difference between them in terms of accuracy [6]. Therefore, we choose two representative model of them as baseline models: LSTM model proposed by Woodbridge et al. [2] and CNN model proposed by Saxe et al. [5].

In addition, among existing works which focus on detecting wordlist-based AGDs, we choose the model utilizing ELMo proposed by Koh et al. [10] as another baseline model, because it is the state-of-the-art approach based on deep learning and does not rely on other contextual information.

V. Results

We implement our model and baseline models using Keras framework, then evaluate the performance of these four models for binary classification task and multi-class classification task, respectively. All experiments are performed using 10-fold cross validation. The results are presented in this section.

A. Binary Classifiaction

TABLE II presents Precision, Recall, F_1 score as well as the training time of four models on binary classification. As we can see, our model gets best score in terms of Precision. Although the Recall of our model is not the highest, the F_1 score of our model, which measures the trade-off of Precision and Recall, is higher than those of other baselines. Besides, the ROC curves of four models and their AUC are shown in Fig.4. Our model shows best performance with the maximum AUC of 0.9962. Moreover, at the same FPR, the TPR of our model are always larger than that of other models.

It is notable that our model not only gets high accuracy on character-based domains, but also has outstanding ability to classify wordlist-based domains. For example, our model can detect Matsnu, one of representative wordlist-based DGAs, with an Recall of 0.9590. While for LSTM model, CNN model and ELMo model, the Recall results are 0.7962, 0.7610 and 0.7552, respectively.

In terms of the training time, two models based on CNN architecture, i.e., our model and CNN model, spend significantly less time than LSTM model and ELMo model. Although the training time of our model is more than CNN model, but the gap between these two models is small.

To comprehensively concluded results mentioned above, our model slightly outperforms other baseline models for binary classification task, which suggests that our model has the ability to make a more accurate decision about whether a given domain is malicious or benign.

TABLE II
RESULT OF BINARY CLASSIFICATION

Model	Precision	Recall	F_1 Score	Training Time (s)
LSTM [2]	0.9925	0.9866	0.9895	17,890
CNN [5]	0.9896	0.9880	0.9888	2636
ELMo [10]	0.9834	0.9896	0.9865	75,621
Our Model	0.9961	0.9875	0.9918	4226

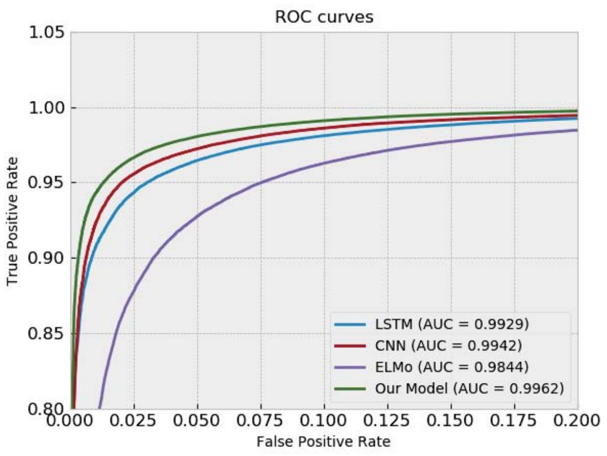

Fig. 4. ROC curves for the binary classification by our model and other baseline models.

B. Multi-class Classifiaction

The empirical results of multi-class classification by four models are set out in TABLE III. Both macro average scores and weighted average scores are provided. Our model significantly outperforms other baseline models on all metrics, and it is a little slower than CNN model but much quicker than LSTM model and ELMo model.

The results suggest that our model is more likely to accurately identify which DGA family the malicious domains belong to. In practice, the right DGA family labels can provide useful information for further analysis.

Specially, ELMo model has a relatively poor performance in this experiment. The reason might be that ELMo model have an excellent performance merely on the dataset including a few DGA families.

VI. Conclusion

This paper presents a novel real-time DGA detection method with special emphasis on high coverage and high efficiency. We first convey a domain name into a sequence of word-level or character-level components, which might be single letter, number, other punctuation or meaningful word, then design a deep neural network based on temporal convolutional network to extract the implicit pattern and classify the domain into two or more categories. Experiments on dataset including various kinds of DGA families showed that our model outperforms other state-of-the-art methods in terms of detection accuracy, detection coverage as well as training efficiency.

TABLE III
RESULT OF MULTI-CLASS CLASSIFICATION

Model	**Macro Average**			**Weighted Average**			**Training Time** (s)
	Precision	*Recall*	F_1 *Score*	*Precision*	*Recall*	F_1 *Score*	
LSTM [2]	0.5705	0.5843	0.5615	0.7596	0.7851	0.7632	24,950
CNN [5]	0.5493	0.5478	0.5307	0.7593	0.7675	0.7467	1326
ELMo [10]	01760	0.1559	0.1437	0.3085	0.3353	0.2766	56,040
Our Model	0.6115	0.6390	0.6123	0.8017	0.8217	0.8082	4553

Further research should be undertaken to explore how to improve the performance of detecting the domains generated by new DGA family which is not included in the training data. In addition, how to minimize the impact of class imbalance in multi-class classification remains to be answered.

ACKNOWLEDGMENT

This work was supported by the Application Program of New Model of Intelligent Manufacturing under the Grant No. 2016ZNZZ01.

REFERENCES

[1] D. Plohmann, K. Yakdan, M. A. Klatt, J. Bader, and E. Gerhards-Padilla, A Comprehensive Measurement Study of Domain Generating Malware, in USENIX Security Symposium, 2016.
[2] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, Predicting Domain Generation Algorithms with Long Short-Term Memory Networks, arXiv:1611.00791 [cs.CR], 2016.
[3] P. Lison and V. Mavroeidis, Automatic Detection of Malware-Generated Domains with Recurrent Neural Models, arXiv:1709.07102 [cs.CR], 2017.
[4] B. Yu, D. L. Gray, J. Pan, M. De Cock, and A. C. A. Nascimento, Inline DGA Detection with Deep Networks, in 2017 IEEE International Conference on Data Mining Workshops (ICDMW), 2017, pp. 683692.
[5] J. Saxe and K. Berlin, eXpose: A Character-Level Convolutional Malicious URLs , File Paths and Registry Keys, arXiv:1702.08568 [cs.CR], 2017.
[6] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, Character Level based Detection of DGA Domain Names, in 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 18.
[7] L. Yang et al., A Novel Detection Method for Word-Based DGA, in Cloud Computing and Security, 2018, pp. 472483.
[8] M. Pereira, S. Coleman, B. Yu, M. DeCock, and A. Nascimento, Dictionary extraction and detection of algorithmically generated domain names in passive DNS traffic, in Lecture Notes in Computer Science, 2018, vol. 11050 LNCS, pp. 295314.
[9] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, Detecting DGA domains with recurrent neural networks and side information, arXiv:1810.02023 [cs.CR], 2018.
[10] J. J. Koh and B. Rhodes, Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings, in 2018 IEEE International Conference on Big Data, 2018, pp. 29662971.
[11] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, Detecting algorithmically generated malicious domain names, in the 10th ACM SIGCOMM conference on Internet measurement, 2010, pp. 4861.
[12] S. Yadav, A. K. K. Reddy, A. L. Narasimha Reddy, and S. Ranjan, Detecting algorithmically generated domain-flux attacks with DNS traffic analysis, IEEE/ACM Trans. Netw., vol. 20, no. 5, pp. 16631677, 2012.
[13] M. Antonakakis et al., From Throw-away Traffic to Bots: Detecting the Rise of DGA-based Malware, in the 21st USENIX Conference on Security Symposium, 2012, p. 24.
[14] Y. Zhou, Q.-S. Li, Q. Miao, and K. Yim, DGA-Based Botnet Detection Using DNS Traffic, Journal of Internet Services and Information Security, vol. 3, no. 11, pp. 116123, 2013.
[15] S. Krishnan, T. Taylor, F. Monrose, and J. McHugh, Crossing the threshold: Detecting network malfeasance via sequential hypothesis testing, in 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2013, pp. 112.
[16] J. Raghuram, D. J. Miller, and G. Kesidis, Unsupervised, low latency anomaly detection of algorithmically generated domain names by generative probabilistic modeling, Journal of Advanced Research, vol. 5, no. 4, pp. 423433, Jul. 2014.
[17] M. Mowbray and J. Hagen, Finding Domain-Generation Algorithms by Looking at Length Distribution, in 2014 IEEE International Symposium on Software Reliability Engineering Workshops, 2014, pp. 395400.
[18] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, Phoenix: DGA-based botnet tracking and intelligence, in Lecture Notes in Computer Science, 2014, vol. 8550 LNCS, pp. 192211.
[19] H. Mac, D. Tran, V. Tong, L. G. Nguyen, and H. A. Tran, DGA Botnet Detection Using Supervised Learning Methods, in Proceedings of the Eighth International Symposium on Information and Communication Technology, 2017, pp. 211218.
[20] R. Sivaguru, C. Choudhary, B. Yu, V. Tymchenko, A. Nascimento, and M. De Cock, An Evaluation of DGA Classifiers, in 2018 IEEE International Conference on Big Data, 2018, pp. 50585067.
[21] R. Vinayakumar, K. P. Soman, P. Poornachandran, and S. Sachin Kumar, Evaluating deep learning approaches to characterize and classify the DGAs at scale, Journal of Intelligent & Fuzzy Systems, vol. 34, no. 3, pp. 12651276, Mar. 2018.
[22] C. Choudhary, R. Sivaguru, M. Pereira, A. C. Nascimento, and M. De Cock, Algorithmically Generated Domain Detection and Malware Family Classification, Secur. Comput. Commun., pp. 640655, 2019.
[23] S. Bai, J. Z. Kolter, and V. Koltun, An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, arXiv:1803.01271 [cs.LG], 2018.