Proceedings of the 6th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/64/20241334

Deep learning DGA malicious domain name detection based on multi-stage feature fusion

Mingtian Xie[1,2,*], Ruifeng He[1,3], Aixing He[1,4]

[1]Southwest University, No. 25 Teaching Building, Beibei District, Chongqing City

[2]xmttz23@163.com
[3]private@email.swu.edu.cn
[4]mambaforever248@163.com
*corresponding author

Abstract. In recent years, cybersecurity issues have emerged one after another, with botnets extensively utilizing Domain Generation Algorithms (DGA) to evade detection. To address the issue of insufficient detection accuracy in existing DGA malicious domain detection models, this paper proposes a deep learning detection model based on multi-stage feature fusion. By extracting local feature information and positional information of domain name sequences through the fusion of Multilayer Convolutional Neural Network (MCNN) and Transformer, and capturing the long-distance contextual semantic features of domain name sequences through Bi-directional Long Short-Term Memory Network (BiLSTM), these features are finally fused for malicious domain classification. Experimental results show that the model maintains an average Accuracy of 93.26% and an average F1-Score of 93.32% for 33 DGA families, demonstrating better comprehensive detection performance compared to other deep learning detection algorithms.

Keywords: Botnet, DGA Malicious Domain, Convolutional Neural Network, Transformer, BiLSTM.

1. Introduction

In recent developments, botnets have become increasingly complex and large, gradually becoming one of the biggest security threats on the internet. A botnet consists of a large number of controlled hosts, or bots, and one or more Command and Control (C&C) servers, with bots and C&C servers communicating with each other to pass commands and data for launching attacks, which can be used for initiating DDOS attacks, malicious software propagation, spamming, and sensitive information collection among other malicious activities. To make C&C servers more concealed and increase the robustness of botnets, attackers use DNS domain resolution to locate C&C servers. [1] Botnets use Domain Generation Algorithms (DGA) to generate a large number of domains, and both the C&C servers and victim machines run the same DGA algorithm to generate the same list of alternative domains. When an attack is launched, a small portion of these domains are registered to disguise the real C&C server domains. Hosts initiate connection requests according to the periodic domain list, achieving continuous control over the controlled hosts. Therefore, the detection of DGA malicious domains has become an important

DOI: 10.54254/2755-2721/64/20241334

research topic in defending against botnets, playing a significant role in identifying and preventing their spread.

Existing detection models mostly distinguish DGA malicious domains without studying the differences between different DGA families, leading to limited detection scope. To address this issue, this paper proposes a deep learning DGA malicious domain detection model based on multi-stage feature fusion, aimed at improving the accuracy of detection across multiple DGA families.

The structure of this paper is as follows: Section 1, related work on DGA malicious domain detection; Section 2, analysis and design of the malicious domain detection model; Section 3, experimental results and analysis; Section 4, conclusion.

2. Related Work

Currently, the detection of DGA botnets mainly falls into two categories: methods based on network traffic information and methods based on domain name character information.

2.1. Detection Based on Network Traffic Information

The methods based on network traffic information involve learning various behavioral features of malicious traffic from botnets to detect existing DGA malicious domains. The mainstream detection methods model and analyze the behavioral characteristics of botnet DNS traffic to identify DGA malicious domains within the traffic. Leyla Bilge and others [2] proposed a passive DNS analysis technique called EXPOSURE, which extracts 15 features from DNS data and then uses a decision tree classifier for detection. Antonakakis and others [3] introduced the Pleiades system, which detects DGA malicious domains by analyzing the characteristics of a large number of NXDOMAIN responses in DGA botnets. Vinayakumar and others [4] developed a scalable framework based on IPS scale for real-time network analysis, providing situational awareness, classifying and correlating malicious events using deep learning algorithms, analyzing DGA botnets, and finding the corresponding C&C servers. Rikima and others [5] presented a system for detecting DGA-based malicious software communication from encrypted domain name protocol DoH (DNS over HTTPS) traffic, classifying network traffic using Gradient Boosting Decision Trees (GBDT) and tree ensemble models based on communication behavior.

These methods have a certain latency and cannot meet the needs of real-time detection. In different network environments, the results of network traffic collection significantly influence the analysis and detection of malicious domains. Such methods have weak detection capabilities, limited detection scope, low efficiency, and a low detection rate for new DGA family variants.

2.2. Detection Based on Domain Name Character Information

Methods based on domain name character information are divided into two types: one involves manually designing features based on domain name characters, such as the proportion of meaningful characters [6], edit distance and Jaccard coefficient [7], n-grams features [7], morphemes [8], etc., and then using unsupervised and supervised learning techniques for detection. The other method directly employs deep learning approaches, allowing the algorithm model to automatically learn the characteristics of malicious domains for detection. Woodbridge and others [9] utilized Long Short-Term Memory Networks (LSTM) to achieve real-time prediction of DGAs without the need for context information or manually created features. Liang and others [10] proposed three feature extraction methods adapted to domain name length, with their HAGDetector employing CNN and LSTM models and utilizing cross-dataset transfer learning techniques. Yang Cheng and others [11] introduced a malicious domain detection method based on n-gram and Transformer, which adds start and end markers to domain data, segments them into word group elements using the n-gram algorithm, and then transforms them into vectors for input into the Transformer model. Sun and others [12] constructed a deep learning model based on BiLSTM, Attention, and CNN, employing feature extraction and dimension reduction techniques, as well as a method based on centrality construction to identify DGA domains.

These methods can achieve real-time detection, but machine learning involves the cumbersome task of manually designing features, time-consuming feature extraction, and features that are easily evaded.

DOI: 10.54254/2755-2721/64/20241334

In contrast, deep learning automatically extracts features for learning and detection, offering faster detection speed and higher real-time performance. However, deep learning detection is limited to single types, performing poorly on new DGA family variants. Therefore, the current mainstream direction in domain detection is to employ deep learning methods that integrate multiple features and mechanisms for detection.

3. DGA Malicious Domain Detection Model

The flowchart of the deep learning DGA malicious domain detection model based on multi-stage feature fusion proposed in this paper is shown in Figure 1.

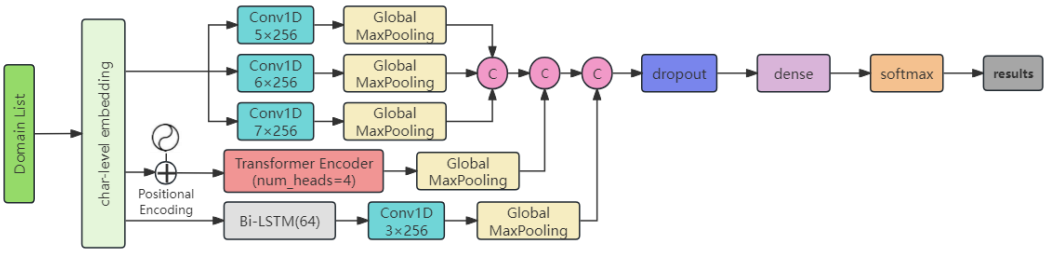

Figure 1. Flowchart of the Malicious Domain Detection Model

The detection model process starts with the preprocessing of the domain list followed by the vectorization of domain name characters. Then comes feature extraction, which is primarily divided into three parts: the Multilayer Convolutional Neural Network (MCNN) module, the Transformer module, and the Bi-directional Long Short-Term Memory Network (BiLSTM) module. The output vectors of the MCNN and Transformer modules are concatenated to learn local feature information of the domain name sequence and understand positional information. These concatenated vectors are then joined with the output vector from the BiLSTM module to obtain the final fused feature representation. A Dropout layer is then used to regularize the feature representation, reducing the risk of overfitting. Finally, the model outputs probabilities for DGA malicious domain detection through a fully connected layer and a softmax activation function.

3.1. MCNN Module

In text classification tasks, Convolutional Neural Networks (CNN) can effectively recognize and extract local features in domain name sequences by utilizing convolutions of multiple sizes. In the method designed in this paper, character vectors first pass through a composite convolution block, where convolutional kernels of sizes 5, 6, and 7 are used for feature extraction. The specific convolution operation can be represented as:

$$C_i = f(W \cdot X_{i:i+k-1} + b) \tag{1}$$

Where X represents the character vector, W is the convolutional kernel, k is the size of the convolutional kernel, b is the bias term, f is the nonlinear activation function (CNNs generally use ReLU as the activation function), and C_i is the convolution result at position i.

Convolutional kernels of different sizes can perceive combinations of characters of different lengths, thus helping the model to capture diverse patterns and structures within the domain name sequence. Subsequently, for each different kernel size, a global max-pooling operation is performed to extract the most significant features, resulting in a fixed-length feature representation, which can be specifically represented as:

$$P_k = max\{C_1, C_2, \ldots, C_{l-k+1}\} \tag{2}$$

Where P_k is the result of the global max-pooling for the convolutional kernel of size k. Finally, the features extracted by convolutional kernels of different sizes are concatenated together to form a

DOI: 10.54254/2755-2721/64/20241334

comprehensive output vector that captures the key local feature information of the domain name sequence.

3.2. Transformer Module

The Transformer structure excels in handling positional relationships and long-distance dependencies within sequence data. In the method designed for this paper, character vectors are first passed through a positional encoding layer to add positional information to the domain name sequence, which is then added to the input vector for the encoder. Utilizing a Transformer module and considering the practical computational resources available for the experiment, this paper employs a single-layer multi-head self-attention mechanism (with num_heads=4) alongside a feedforward neural network for feature extraction and representation learning of characters. Subsequently, a global max pooling operation is performed to obtain a fixed-length feature representation. The output vector obtained is used to strengthen the understanding of global dependencies and positional information within local features, thereby enhancing the model's generalization performance for multiple malicious domain families.

The self-attention mechanism, a core component of the Transformer, is pivotal in computing the attention weights between each input position and other positions. These weights are used to compute a weighted sum, thereby obtaining a context representation for each character position. The attention mechanism is formulated as follows:

$$\text{Attention}\ (\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\ (\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}})\boldsymbol{V} \tag{3}$$

Where Q, K, V represent the query, key, and value matrices, respectively.

These matrices are obtained by applying different linear transformations to the input sequence. The query matrix Q is multiplied by the key matrix K, and then passed through a softmax function to obtain a probability distribution corresponding to the length of the input sequence. This distribution represents the importance of each element to the query matrix Q. Finally, this probability distribution is multiplied by the value matrix V to obtain the self-attention vectors, representing the weighted average value of each element. The formula for multi-head attention is as follows:

$$\text{MultiHead}\ (\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{concat}\ (\ \text{head}\ _1,\ \text{head}\ _2, \cdots, head_n)\boldsymbol{W}^o \tag{4}$$

Where $\text{head}_i = \ \text{Attention}\ (\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V)$ represents the output of the i-th head, nis the total number of heads, and $\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V$ are parameter matrices for linearly mapping Q, K, V respectively, learned by the model.

3.3. BiLSTM Module

In this paper's method, a Bi-directional Long Short-Term Memory Network (BiLSTM) is used to model the domain name sequence, followed by a convolution block and a global max pooling operation to obtain a fixed-length feature representation. By processing the sequence in both forward and backward directions, BiLSTM is adept at handling long-distance sequence dependencies. The output vector effectively captures the long-term contextual semantic features of the domain name sequence. The structure of the BiLSTM algorithm model is shown in Figure 2.

DOI: 10.54254/2755-2721/64/20241334

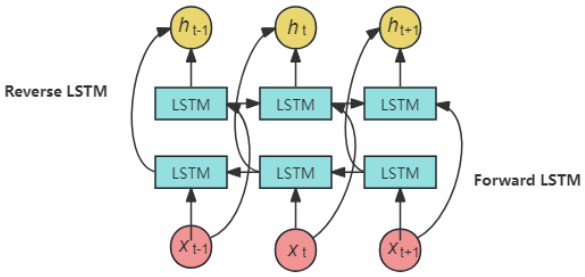

Figure 2. BiLSTM Algorithm Model

4. Experimental Verification and Result Analysis

This study's experiments were conducted on an Ubuntu 20.04 system, with a Xeon(R) Silver 4214R CPU, 90GB of RAM, and an NVIDIA RTX 3080 Ti (12GB) GPU. The model implementation was based on the TensorFlow 2.5.0 deep learning framework, with Python version 3.8, and the development environment being Pycharm.

4.1. Dataset

The experimental dataset was sourced from public malicious domain datasets including 360NetLab DGA [13], OSINT [14], and DGArchive [15]. To mitigate the impact of sample data imbalance on model training and detection effectiveness, the collected malicious domains were aggregated and deduplicated. Subsequently, 33 malicious families with ample samples were selected, with approximately 10,000 samples randomly chosen from each family to form the malicious domain dataset, totaling 326,000 malicious domains. The entire dataset was divided into a training set and a test set at a 4:1 ratio.

4.2. Evaluation Metrics

The detection model of this study used *Accuracy*, *Precision*, *Recall*, and *F1-score* as evaluation metrics, calculated as shown in equation (5):

$$\begin{cases} \text{Accuracy} = \dfrac{TP + TN}{TP + FP + TN + FN} \\ \text{Precision} = \dfrac{TP}{TP + FP} \\ \text{Recall} = \dfrac{TP}{TP + FN} \\ F1 - score = \dfrac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \end{cases} \tag{5}$$

Where TP represents the number of malicious domains correctly detected, FN represents the number of malicious domains incorrectly reported as benign, TN represents the number of legitimate domains correctly detected, and FP represents the number of legitimate domains incorrectly reported as malicious. Higher values of *Accuracy* and *F1-score* indicate superior comprehensive performance of the model.

4.3. Analysis of Experimental Results

The detection results of the proposed model for the sufficiently large dataset of 33 DGA malicious domain families are shown in Table 1.

DOI: 10.54254/2755-2721/64/20241334

Table 1. Multi-Family Malicious Domain Detection Results

family	Precision(%)	Recall(%)	F1-Score(%)	family	Precision(%)	Recall(%)	F1-Score(%)
banjori	100.00	99.85	99.92	dyre	100.00	100.00	100.00
necurs	97.44	83.80	90.11	monerodownloader	100.00	100.00	100.00
flubot	82.95	97.05	89.45	pykspa	88.90	82.90	85.80
qsnatch	99.65	98.80	99.22	simda	99.70	100.00	99.85
bazarloader	100.00	100.00	100.00	locky	85.37	65.95	74.41
tinba	86.91	97.60	92.12	urlzone	96.71	93.95	95.31
gameover	99.70	99.55	99.62	symmi	100.00	100.00	100.00
virut	93.33	99.40	96.27	tinyfluff	99.95	100.00	99.98
rovnix	100.00	99.90	99.95	nymaim	73.22	80.50	76.68
ramnit	67.17	75.40	71.30	tufik	88.17	88.30	88.23
cryptolocker	75.41	69.15	72.39	suppobox	99.20	99.70	99.45
conficker	73.52	71.90	72.70	qadars	99.90	99.80	99.85
ramdo	99.95	100.00	99.98	emotet	99.70	100.00	99.85
matsnu	99.85	99.85	99.85	mydoom	100.00	100.00	100.00
pushdo	99.20	99.65	99.43	kraken	93.11	87.20	90.06
ranbyus	86.33	90.00	88.13	pushdotid	99.17	100.00	99.59
metastealer	100.00	100.00	100.00	均值	93.47	93.34	93.32

From Table 1, it can be seen that the algorithm achieved an F1-Score of over 99% for 19 DGA families and over 90% for 23 DGA families. Overall, the model proposed in this paper is effective for multi-class detection of DGA malicious domains.

4.4. Comparative Experimental Analysis

In this section, four models were selected for comparison: LSTM [9], HDNN [16], CNN-BiLSTM [17], and FEDCC [12]. Using the same experimental setup and dataset, comparative experiments were conducted with the CT_B model proposed in this study. The results are shown in Table 3.

Table 3. Comparative Performance of Models for Multi-Family Malicious Domain Detection

Model	Acc(%)	Precision(%)	Recall(%)	F1-Score(%)
LSTM[9]	91.81	92.45	91.90	91.95
HDNN[16]	92.10	92.39	92.19	92.16
CNN_BiLSTM[17]	92.22	92.61	92.31	92.31
FEDCC[12]	92.29	92.55	92.39	92.35
CT_B	**93.26**	**93.47**	**93.34**	**93.32**

From Table 3, it is evident that, compared to the benchmark models, the model proposed in this paper shows the best overall performance, with both Accuracy and F1-Score exceeding 93%. Compared to the other four methods, the increases in Accuracy were 1.45%, 1.16%, 1.04%, and 0.97%, respectively, and the increases in F1-Score were 1.37%, 1.16%, 1.01%, and 0.97%, respectively. This demonstrates that the model proposed in this paper, by capturing multi-level features from local to global and from surface to depth, enhances the model's generalization capability for new samples.

5. Conclusion and Closing Remarks

In response to the current challenges in the precision and identification of new variant families faced by DGA malicious domain detection models, this paper proposes a deep learning DGA malicious domain

DOI: 10.54254/2755-2721/64/20241334

detection model based on multi-stage feature fusion. After vectorizing the domain name characters, the model combines Multilayer Convolutional Neural Network (MCNN) and Transformer modules to capture the local characteristics and positional information of the domain names. It then integrates with Bi-LSTM to refine the long-distance contextual semantic features of the domain name sequences, finally utilizing these fused features for malicious domain classification. Through comparison with other deep learning detection algorithms, the superiority of the algorithm model proposed in this paper for malicious domain detection performance has been validated.

Future work primarily includes two aspects: on one hand, considering the introduction of anomaly detection technology in real network environments to detect new family variants of DGA through DNS traffic behavior, thereby improving the model's sensitivity to new DGA families; on the other hand, allowing the model to undergo continuous monitoring and evaluation in actual environments, with regular updates and adjustments to the model.

References

[1] Wang, Y. Y., Wu, C. J., Liu, Q. H., et al. (2019). A survey of research and application of malicious domain name detection. Computer Applications and Software, 36(9), 310-316.

[2] Bilge L, Kirda E, Kruegel C, et al. Exposure: Finding malicious domains using passive DNS analysis[C]//Ndss. 2011: 1-17.

[3] Antonakakis M, Perdisci R, Nadji Y, et al. From {Throw-Away} Traffic to Bots: Detecting the Rise of {DGA-Based} Malware[C]//21st USENIX Security Symposium (USENIX Security 12). 2012: 491-506.

[4] Vinayakumar R, Poornachandran P, Soman K P. Scalable framework for cyber threat situational awareness based on domain name systems data analysis[J]. Big data in engineering applications, 2018: 113-142.

[5] Mitsuhashi R, ** Y, Iida K, et al. Detection of DGA-based Malware Communications from DoH Traffic Using Machine Learning Analysis[C]//2023 IEEE 20th Consumer Communications & Networking Conference (CCNC). IEEE, 2023: 224-229.

[6] Schiavoni S, Maggi F, Cavallaro L, et al. Phoenix: DGA-based botnet tracking and intelligence[C]//International Conference on detection of intrusions and malware, and vulnerability assessment. Cham: Springer International Publishing, 2014: 192-211.

[7] Cucchiarelli A, Morbidoni C, Spalazzi L, et al. Algorithmically generated malicious domain names detection based on n-grams features[J]. Expert Systems with Applications, 2021, 170: 114551.

[8] Zhang, W. W., Gong, J., Liu, Q., et al. (2016). A lightweight domain name detection algorithm based on morpheme features. Journal of Software, 27(9), 2348-2364.

[9] Predicting Domain Generation Algorithms with Long Short-Term Memory Networks.2016

[10] Liang J, Chen S, Wei Z, et al. HAGDetector: Heterogeneous DGA domain name detection model[J]. Computers & Security, 2022, 120: 102803.

[11] Yang, C., Lu, T. L., Yan, S. Y., et al. (2022). DGA malicious domain name detection based on N-gram and Transformer. Journal of the Chinese People's Public Security University (Science and Technology), 28(03), 100-108.

[12] Sun X, Liu Z. Domain generation algorithms detection with feature extraction and Domain Center construction[J]. Plos one, 2023, 18(1): e0279866.

[13] Qihoo 360 Technology Co, Ltd.360 DGA feeds[EB/OL].https://data.netlab.360.com/dga/,2022

[14] “Bambenek consulting-master feeds.” http://osint.bambenekconsulting.com/feeds/.accessed: 2023-02-08

[15] Daniel Plohmann. Fraunhofer FKIE. DGArchive. https://dgarchive.caad.fkie.fraunhofer.de, accessed 2023-04-14.

[16] Yang L, Liu G, Dai Y, et al. Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework[J]. Ieee Access, 2020, 8: 82876-82889.

DOI: 10.54254/2755-2721/64/20241334

[17] Namgung J, Son S, Moon Y S. Efficient deep learning models for dga domain detection[J]. Security and Communication Networks, 2021, 2021: 1-15.