DGA Domain Name Classification Method Based on Long Short-Term Memory with Attention Mechanism

Yanchen Qiao [1] [iD]**, Bin Zhang** [1,*]**, Weizhe Zhang** [1,2]**, Arun Kumar Sangaiah** [3] [iD] **and Hualong Wu** [1]

[1]   Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China; qiaoych@pcl.ac.cn (Y.Q.); weizhe.zhang@pcl.ac.cn (W.Z.); hualong.wu@pcl.ac.cn (H.W.)
[2]   School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150000, China
[3]   School of Computer Science and Engineering, VIT University, Vellore 632014, India; sarunkumar@vit.ac.in
[*]   Correspondence: bin.zhang@pcl.ac.cn

check for
updates

Abstract: Currently, many cyberattacks use the Domain Generation Algorithm (DGA) to generate random domain names, so as to maintain communication with the Communication and Control (C&C) server. Discovering DGA domain names in advance could help to detect attacks and response in time. However, in recent years, the General Data Protection Regulation (GDPR) has been promulgated and implemented, and the method of DGA classification based on the context information, such as the WHOIS (the information about the registered users or assignees of the domain name), is no longer applicable. At the same time, acquiring the DGA algorithm by reversing malware samples encounters the problem of no malware samples for various reasons, such as fileless malware. We propose a DGA domain name classification method based on Long Short-Term Memory (LSTM) with attention mechanism. This method is oriented to the character sequence of the domain name, and it uses the LSTM combined with attention mechanism to construct the DGA domain name classifier to achieve the rapid classification of domain names. The experimental results show that the method has a good classification result.

Keywords: security; DGA classification; attention mechanism; LSTM

1. Introduction

In recent years, cyberattacks have experienced explosive growth, which seriously threatens the security of property data of internet users. The domain name is an important kind of infrastructure for cyberattacks. It could be used to maintain a connection with the client to implement data return and command delivery. To avoid the blacklist mechanism and prolong the attack time, the attacker usually uses the DGA to generate new domain names. Domain Generation Algorithm (DGA) is an algorithm used by cyberattackers to generate domain names periodically. The domain name generated by DGA is usually called the DGA domain name. DGA domain names are not completely random. Different attack organizations have different domain name generation algorithms for different attack activities, so that the client can establish communication with the C&C server by using the domain name generated by DGA. As the DGA domain name is generated periodically, the black and white list does not necessarily exist, which affects the response time of the security organization to the attack activity. Determining the attack of a malicious domain name can effectively determine the purpose of the attack, the tools and malware used, etc., so as to ensure a rapid and effective response, greatly reducing the damage caused by cyberattacks.

To detect attacks by detecting DGA domain names, security organizations typically perform reverse analysis of malware samples. They identify DGA-related code and algorithms from malware samples and quickly generate DGA domain names that will be used by the corresponding cyberattack. In turn,

the ability to detect the corresponding cyberattack is formed. In 2014, FireEye analyzed the DGA algorithm of Srizbi malware. According to the DGA algorithm, hundreds of DGA domain names of Srizbi were squatted, which effectively curbed the spread of Srizbi. The analysis of the DGA algorithm from malware samples is a very difficult task and requires experienced reverse analysts who are particularly scarce. At the same time, the attacker uses various means such as packing, encryption, and confusion to prevent being reversed or increase the difficulty of the reverse, so as to increase the time that the DGA is analyzed, thereby prolonging the survival time of the attack. In addition, new types of attacks, as well as fileless malware, are often unable to obtain malware samples, resulting in the inability to reverse the DGA algorithm used by the attack. Therefore, it is quite difficult to obtain the DGA algorithm from the malware samples in reverse, and there are many unachievable factors in reality.

Security researchers can easily obtain WHOIS [1–6] information of domain names before the GDPR is enacted. Then, according to the domain name's context information, it can be determined if it is a DGA domain name, whether it is generated by the same DGA algorithm, and whether the attacker behind is the same. The WHOIS information of the domain name includes the registrant, email address, mobile phone, address, and the like. To avoid being tracked, an attacker typically uses fake information to register. However, to reduce costs, it is common to register multiple domain names with the same registration information. Therefore, the association relationship among these domain names can be determined according to the registration information. For example, Kaspersky's researcher GReAT [7] found the relationship among them through the WHOIS information of the domain name and determined that these attacks were initiated by the Winnti attack organization. However, for new domain names appearing in the network, it is difficult to obtain real-time judgments from obtaining their WHOIS information to making associations based on WHOIS information. At the same time, GDPR began to be enacted and implemented in 2018. After the implementation, it is impossible to obtain the registration information of the domain name from the domain name registrar. Security platforms such as VirusTotal no longer provide the WHOIS information of the domain name. Therefore, DGA detection and classification by context information such as WHOIS of domain names is no longer feasible.

Therefore, we can only determine whether it is a DGA domain name and which type of DGA it belongs to by analyzing the domain name string. There has been a lot of work based on the textual characteristics of domain names. Yadav et al. [8] perform DGA detection using the distribution characteristics of the characters and the 2-gram character set in the domain name. Antonakakis et al. [9] use domain name length, character frequency, randomness, and other characteristics to unsupervised clustering of domain names to achieve DGA domain name detection. The traditional machine learning method has problems, such as feature extraction, relying on expert experience, which makes it easy for an attacker to bypass. Aiming at the problems existing in the current DGA domain name classification, this paper proposes a DGA domain name classification method based on LSTM with attention mechanism through the research of DGA and DGA domain name. This method neither needs to reverse the malware sample nor uses context information such as WHOIS of the domain name, but only uses the character sequence of the domain name. For the character sequence of the domain name, the LSTM algorithm combined with the attention mechanism is used to construct the DGA domain name classifier, to achieve fast and accurate classification of the domain name.

The contributions of this paper mainly include two aspects:

1.　We only use the character sequence of the domain name for DGA domain name classification to further prove that the character sequence contains DGA features.
2.　We combine LSTM with attention mechanisms and apply them to DGA domain name classifications to prove that the weights of characters in DGA domain names are different.

The remainder of the present paper is organized as follows. In Section 2, we briefly present related work to detect and classify DGA domains. In Section 3, we briefly describe the techniques used in our approach. In Section 4, we describe the architecture of our approach. The experimental results are presented in Section 5, and the summaries of the whole paper are presented in Section 6.

2. Related Work

There has been a lot of work using the dynamic features of domain names, including whether it could be resolved to IP, the geographical distribution of IPs, etc. for DGA detection and classification. In 2010, Antonakakis et al. [10] established a dynamic domain name scoring system that uses three types of features, including network-based features (such as historical IP number of domain names, geographic distribution, AS domain, etc.), domain-based features (such as the length of a domain name, character distribution, etc.), and evidence-based features (including whether it is associated with a known malware family, whether it resolves to a malicious IP, etc.). In the actual environment deployment test, the accuracy of the system is as high as 96.8%. In 2010, Yadav et al. [11] developed a methodology to detect domain fluxes in DNS traffic by looking for patterns inherent to domain names that were generated algorithmically, in contrast to those generated by humans. Moreover, they applied the methodology to packet traces collected at a Tier-1 ISP and showed they could automatically detect domain fluxing, as used by the Conficker botnet, with minimal false positives. In 2011, Bilge et al. [4] proposed a malicious domain name detection technology based on the passive domain name analysis method. They extracted 15 types of features from traffic, including domain name lifetime, period similarity, number of accesses, number of IPs parsed, whether IP is shared by other domain names, digital symbol ratio and length of longest meaningful substring, etc. Finally, the classifier was constructed using the J48 decision tree algorithm. After verification by the actual environment, the detection accuracy of this method is as high as 98%. In 2012, Antonakakis et al. [9] presented a new technique to detect randomly generated domains without reversing. Their insight was that most of the DGA-generated (random) domains that a bot queries would result in Nonexistent Domain (NXDomain) responses, and that bots from the same botnet (with the same DGA algorithm) would generate similar NXDomain traffic. Using a multi-month evaluation phase, they showed that the system could achieve very high detection accuracy. In 2013, Krishnan et al. [12] proposed a method for detecting attack activity using Sequential Hypothesis Testing. They believed that hosts that have been infected by malware will exhibit a domain name scanning behavior. The majority of the scanned domain names could not resolve the IP. This behavior was abnormal. This type of abnormal behavior was detected first, then the domain name requested by the terminal was analyzed, and the malicious domain name was determined using the Zipf filter.

As the dynamic analysis consumed more computational resources and took a long time, many works only used the character and sequence features of the domain name for detection and classification. In 2012, Yadav et al. [8] proposed a DGA domain name detection method. This work was inspired by the observation that the difference in character distribution between the normal domain name and the DGA domain name was quite large. They used the characters in the domain name and the distribution characteristics of the 2-gram character set, and finally used the edit distance and Jaccard distance [13] algorithm. The actual detection rate of the method was as high as 83.87%. In 2012, Antonakakis et al. [9] proposed a method for determining DGA domain names from unresolved domain names. They first used the characteristics of domain name length, character frequency, randomness, and other characteristics to unsupervised clustering of domain names. Then, they used the Markov-based classification model to determine the attack behind the domain name, and filtered out the active domain name, which was the C&C domain name. In 2014, Bilge et al. [14] designed a system, called EXPOSURE, to detect DGA domains in real-time, by applying 15 unique features grouped in four categories. They conducted a controlled experiment with a large, real-world dataset consisting of billions of DNS requests. The results showed that the system worked well in practice, and that it was useful in automatically identifying a wide category of malicious domains and the relationships between them. In 2014, Schiavoni et al. [15] built a DGA domain botnet tracking and intelligence system. First, the character-based and IP-based features were used to identify the DGA and non-DGA domain names, and then the DGA domain names were grouped to identify the botnet to which they belonged; they were tested on more than 1 million domain names, and the detection accuracy was as high as 94.8%. In 2016, Woodbridge et al. [16] presented a DGA classifier that

leveraged long short-term memory (LSTM) networks for real-time prediction of DGAs without the need for contextual information or manually created features. Experimental results showed that the method was significantly better than all state-of-the-art techniques. In 2017, Yu et al. [17] proposed a DGA domain name detection method based on deep learning. The CNN and LSTM algorithms were used to construct the classification model. The accuracy rates were 72.89% and 74.05%, respectively.

3. Theoretical Basis

3.1. Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) [18] with the characteristics of processing historical data and modeling memory is an important branch of deep learning. From a biological perspective, RNN is a simple simulation of the biological neural system ring link, which is suitable for tasks with time series characteristics such as handwriting font recognition, speech recognition, and natural language processing. The original RNN is composed of an input vector, x; a hidden layer state, s; an output vector, h; a weight parameter, U, of the input sequence information; a weight parameter, W, of the hidden layer state; a weight parameter, V, of the output sequence information; and the like.

S_t is calculated based on the hidden layer state s_{t-1} at the previous time and the input x_t at the current time. Let the activation function of the hidden layer state be f, then the current hidden layer state, s_t, is calculated as

$$s_t = f(Ws_{t-1}, Ux_t) \tag{1}$$

Assuming that the output activation function is g, the output is calculated as

$$h_t = g(Vs_t) \tag{2}$$

It can be seen from the formula that the hidden layer state s_t of the RNN has a memory function for the sequence, and the sequence information can be retained by the hidden layer state. The DGA domain name is a sequence of characters that is automatically constructed using algorithms. The DGA domain name can be modeled using RNN for detection or classification.

3.2. Long Short-Term Memory (LSTM)

However, limited by structure, it is difficult for the original RNN to learn valid data in long-term dependent sequence data. Inputs that are far from the current moment cannot contribute to the update of the current time model parameters, the so-called gradient disappearance problem. The length of the DGA domain name is usually very long. For example, the switch domain name of Wannacry is 41 characters long, and, in practice, DGA domain names longer than 70 are encountered often. The most popular solution to the problem of RNN gradient disappearance is using the LSTM [19] structure instead of using the sigmoid activation function in the original RNN.

The LSTM is a special artificial RNN architecture [19] used in the field of deep learning. LSTM has proven to be more effective than traditional RNN models in dealing with long-term dependency problems. LSTM and RNN are similar in timing, but the way to calculate the state of hidden layer neurons is different. Each memory unit of the LSTM includes four main elements: input gate, forgetting gate, output gate, and self-looping connected units. Thus, the output value is controlled between 0 and 1, responsible for describing how much is passed. At time t, x_t represents the input; i_t is the activation value of the input gate; i_t is the activation value of the input gate; f_t is the activation value of the forgetting gate; o_t is the activation value of the output gate; h_t and h_{t-1} are the outputs of the memory cell at times t and $t-1$, respectively; C_t and C_{t-1} are the states of the memory cell at time t and $t-1$, respectively; and C'_t is the candidate state of the memory cell. W_i, U_i, W_c, U_c, W_f, U_f, W_o, U_o, etc. are the weights of the corresponding gate in the main memory unit. b_i, b_c, b_f, b_o, etc. are the offsets of the corresponding gates in the memory unit. σ in the figure is the activation function.

The state of the memory cell at time t is as follows,

$$C_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \times tan(W_c x_t + U_c h_{t-1} + b_c) \\ + \sigma(W_f x_t + U_f h_{t-1} + b_f) \times C_{t-1} \tag{3}$$

The output of the t-memory unit is

$$H_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \times tan(C_t) \tag{4}$$

This mechanism of the LSTM memory unit allows long-term storage and access to sequence information, thereby reducing the problem of gradient disappearance. It is suitable for building DGA domain name detection and classification models. Woodbridge et al. [16] used LSTM constructed DGA domain name detection and classification model to obtain Good results.

3.3. Attention Mechanism

When we analyzed the DGA algorithm, we found that the attacker can control the domain name generated by the DGA; that is, the domain name generated in one cycle cannot be duplicated with the registered domain name of other person, but can hit the attacker to register the domain name, usually adding some restrictions. For example, the Banjori domain name only transforms the first four letters, and the part after the domain name is unchanged. Therefore, as long as the latter part is detected, the domain name can basically be regarded as a DGA and does not need to pay attention to the entire domain name. Therefore, this paper introduces the attention mechanism, which, in the classification model, gives different attention to different parts of the input domain name, effectively improving the classification effect of DGA domain names.

In recent years, attention mechanism has been widely used in various types of deep learning tasks such as machine translation, image recognition, and speech recognition. Compared with simple Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), better results have been achieved. The use of self-attention in the BERT model [20] greatly enhances the effect of machine translation. When human eyes view pictures, they quickly scan the global image to obtain the target area that needs to be focused on, that is, the focus of attention, and then invest more attention resources in this area to obtain the details of the target while suppressing other useless information. The attention mechanism in deep learning is essentially similar to the human selective visual attention mechanism. The goal is to select information that is more critical to the current mission objectives from a variety of information. There are a number of popular attention mechanisms, such as content-based attention [21], attention based on (global/local) location [22], self-focused [23], etc. This paper refers to the global attention [22] and uses a simpler attention mechanism to reduce computational performance.

The most important aspect of the global attention mechanism is the calculation of its position weight, assuming $H = [h_1, h_2, \cdots, h_m]^T$, where $h_i = [h_i^1, h_i^2, \cdots, h_i^n]$, then H is a matrix of $m \times n$:

$$H = \begin{bmatrix} h_1^1 & \cdots & h_1^n \\ \vdots & \ddots & \vdots \\ h_m^1 & \cdots & h_m^n \end{bmatrix} \tag{5}$$

We need to calculate the position weight for h_i; first, transpose H, then H^T is a matrix of $n \times m$:

$$H^T = \begin{bmatrix} h_1^1 & \cdots & h_m^1 \\ \vdots & \ddots & \vdots \\ h_1^n & \cdots & h_m^n \end{bmatrix} \tag{6}$$

Then, use the softmax function for each line of H^T, the result is as follows,

$$S^T = \begin{bmatrix} \frac{h_1^1}{\Sigma_1^m h_j^1} & \cdots & \frac{h_m^1}{\Sigma_1^m h_j^1} \\ \vdots & \ddots & \vdots \\ \frac{h_1^n}{\Sigma_1^m h_j^n} & \cdots & \frac{h_m^n}{\Sigma_1^m h_j^n} \end{bmatrix} = \begin{bmatrix} a_1^1 & \cdots & a_m^1 \\ \vdots & \ddots & \vdots \\ a_1^n & \cdots & a_m^n \end{bmatrix} \tag{7}$$

Then transpose S^T to get the required weight matrix S and the attention matrix:

$$S = \begin{bmatrix} a_1^1 & \cdots & a_1^n \\ \vdots & \ddots & \vdots \\ a_m^1 & \cdots & a_m^n \end{bmatrix} \tag{8}$$

Finally, H is multiplied element-wise by S to achieve the attention mechanism:

$$H \times S = \begin{bmatrix} h_1^1 a_1^1 & \cdots & h_1^n a_1^n \\ \vdots & \ddots & \vdots \\ h_m^1 a_m^1 & \cdots & h_m^n a_m^n \end{bmatrix} \tag{9}$$

The attention mechanism can be placed before the LSTM layer or behind the LSTM layer. This paper demonstrates that the attention mechanism is better behind the LSTM layer. Therefore, this paper puts the attention mechanism behind the LSTM layer.

4. Methodology

4.1. Overview

As can be seen in Figure 1, the method of this paper is composed of training and testing phrases. In both training and testing phrases, the domain names of the train and test dataset must be preprocessed. For each domain name, after DGA string extraction, padding, and embedding, it is converted to a 54×128 matrix. Then, in the training phrase, the matrices of the train dataset are fed into the deep learning network to generate a classification model. Finally, the matrices of the test dataset are tested by the classification model. The specific description is as follows.
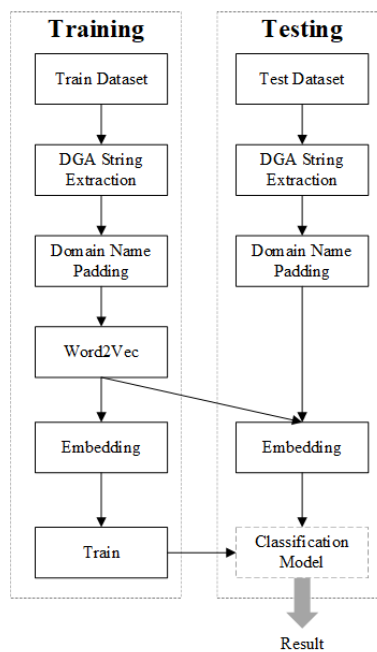
Figure 1. Overview of the Domain Generation Algorithm (DGA) Classification Method.

4.2. DGA String Extraction

In general, to improve controllability and avoid being quickly blocked, most attackers will register their second-level domain name. For example, "h7smcnrwlddsdn34fgv.info" is the DGA domain name registered for the malicious code of Sality. However, registering a second-level domain name has a certain cost. Moreover, even if forged information is used, traceable information, such as IP, may be left, and there is a possibility of being traced by the source. At the same time, some attacks use dynamic domain name services to generate third-level domain names to save attack costs, such as "blackshadespro.no-ip.org". The second-level domain name and third-level domain name of the dynamic domain name service are not necessarily completely separated. Two types of domain names may be used simultaneously in one attack or different attacks launched by the same organization. In the domain name, this paper uses the string generated by DGA for detection and classification, such as "h7smcnrwlddsdn34fgv" in "h7smcnrwlddsdn34fgv.info", "blackshadespro" in "blackshadespro.no-ip.org", and so on. Therefore, we follow the following guidelines to extract the DGA domain name string from the domain name.

1. If it is a second-level domain name, the second-level domain name part is extracted,
2. If it is a third-level domain name, first determine whether the second-level domain name is the domain name of the dynamic domain name service, such as "no-ip.com", "afraid.org", "duckdns.com", "dnsdynamic.org", "dyndns.net", "dynu.com", etc., if so, the third-level domain name part is extracted.
3. If the second-level domain name in the third-level domain name is not the domain name of the dynamic domain name service provider, the longest string is extracted.
4. Otherwise, extract the longest string.

4.3. Domain Name Padding

The length of the domain name is not fixed. Some works use the domain name length [9,10] as one of the characteristics of the DGA domain name detection and classification. Table 1 shows the lengths of 11 common DGA domain names. It can be seen from the table that the lengths of different DGA domain names are usually different; however, the method proposed in this paper requires a fixed length as input.

Table 1. Eleven types of DGA domain name length.

DGA	Length	Example
banjori	10	pdtmstring
corebot	23	a0c4e8sr70oluhsf3t1h1va
cryptolocker	8	rifxkpdx
dircrypt	10	xzdiobjady
kraken	8	iuhqhbmq
locky	17	qqeuxqbetndnsclkm
pykspa	9	folmecyca
qakbot	20	gutkdzfamdgsjbhpuoyb
ramdo	8	kuekesqm
ramnit	9	byqdmekgd
simda	23	jewumerydatyvyjolyvofoh

Usually, the longest length in the dataset is chosen as the fixed length, and then the short domain name is padded. However, considering the scalability of the model, this paper makes statistics on the length of the DGA domain name published on Bambenek Consulting [24]. The result is shown in Figure 2; most DGA domain names are concentrated in the 10–20 interval, and the longest DGA domain name is 44, which covers more possibilities and guarantees the performance of the method.

Adding 10, we use 54 as the input length; all domain names are padded to 54. This paper uses the symbol "*", which is not allowed in the domain name, to complete the domain name.
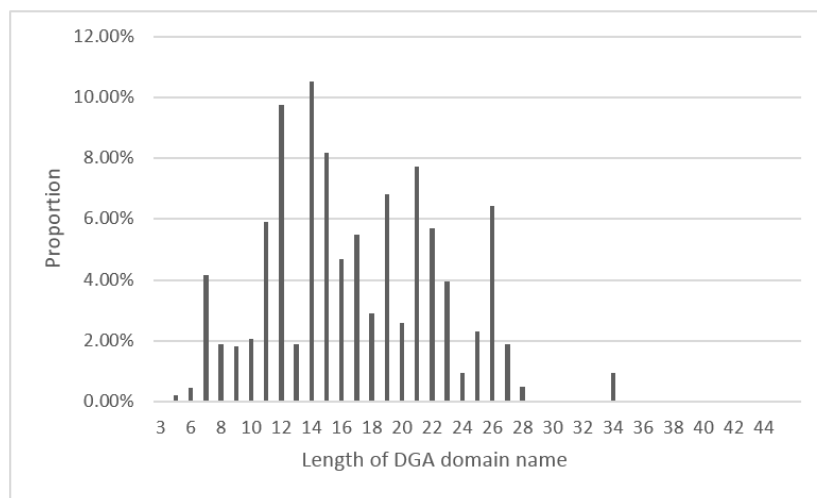

Figure 2. DGA domain name length distribution.

4.4. Embedding

After each domain name is completed, the form is $d = [a_1, a_2, \cdots, a_{54}]$, where the subscript of a indicates the location. Using characters as words, each domain name can be thought of as a sentence composed of characters. Next, using Word2Vec's CBOW model, the word vector of all characters in the domain name is calculated on the complete training set. In this paper, the dimension of the word vector is set to 128, taking into account the number of element in ASCII is 128. The word vector of each character can be expressed as $W^a = [x_1^a, x_2^a, \cdots, x_{128}^a]$, where a represents the character in the domain name. The word vectors are then organized in the order of the characters in the domain name so that each domain name is converted to a 54×128 matrix, as follows,

$$M^d = \begin{bmatrix} W^{a_1} \\ W^{a_2} \\ \vdots \\ W^{a_{54}} \end{bmatrix} = \begin{bmatrix} x_1^{a_1} & \cdots & x_{128}^{a_1} \\ \vdots & \ddots & \vdots \\ x_1^{a_{54}} & \cdots & x_{128}^{a_{54}} \end{bmatrix} \tag{10}$$

4.5. Deep Learning Network Structure

In this paper, we use the LSTM combined with the attention mechanism, as shown in Figure 3; the functions of each layer are as follows,

1. INPUT: Input layer, the domain name is converted to a matrix of dimension 54×128 after the length is padded and embedding, so the input dimension is 54×128.
2. LSTM: LSTM layer, sequence output, and output 54×128 feature vector.
3. ATTENTION: Attention mechanism, according to Section 3.3, and output 54×128 feature vector.
4. FC: Fully connected layer, which stretches the feature vector output by ATTENTION. Each pixel represents a unit. The output feature is 6912 units using the fully connected layer operation, and the probability of DROPOUT is set to 0.5.
5. OUTPUT: Output layer, this layer is fully connected with the FC layer; the output length is the required number of classifications, which represents which classification the extracted features belong to; and the classification function is Softmax.
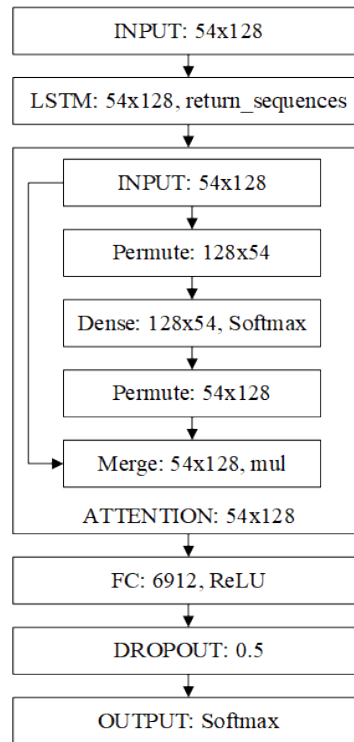
```
┌─────────────────────────────────┐
│        INPUT: 54x128            │
└─────────────────────────────────┘
                 ↓
┌─────────────────────────────────┐
│  LSTM: 54x128, return_sequences │
└─────────────────────────────────┘
                 ↓
┌─────────────────────────────────┐
│  ┌───────────────────────────┐  │
│  │      INPUT: 54x128        │  │
│  └───────────────────────────┘  │
│                ↓                │
│  ┌───────────────────────────┐  │
│  │    Permute: 128x54        │  │
│  └───────────────────────────┘  │
│                ↓                │
│  ┌───────────────────────────┐  │
│  │  Dense: 128x54, Softmax   │  │
│  └───────────────────────────┘  │
│                ↓                │
│  ┌───────────────────────────┐  │
│  │    Permute: 54x128        │  │
│  └───────────────────────────┘  │
│                ↓                │
│  ┌───────────────────────────┐  │
│  │   Merge: 54x128, mul      │  │
│  └───────────────────────────┘  │
│        ATTENTION: 54x128        │
└─────────────────────────────────┘
                 ↓
┌─────────────────────────────────┐
│        FC: 6912, ReLU           │
└─────────────────────────────────┘
                 ↓
┌─────────────────────────────────┐
│        DROPOUT: 0.5             │
└─────────────────────────────────┘
                 ↓
┌─────────────────────────────────┐
│        OUTPUT: Softmax          │
└─────────────────────────────────┘
```

Figure 3. Design of Classification Network Structure Based on Attention Mechanism and LSTM.

The parameters in the network structure are as follows.

- Classifier: Based on the characteristics of DGA domains, we use Softmax classifier to judge which type the domain belongs to. The essence of Softmax function is to map a K-dimensional arbitrary real vector to another *K*-dimensional real vector, where the value of each element in the vector is in the $[0, 1]$ interval, as shown by Formula (11), where v_j is the j element of the vector, and the Softmax value of the element is $softmax(v_j)$,

$$softmax(v_j) = \frac{v_j}{\sum_{k=1}^{K} v_k} \tag{11}$$

- Loss function: When the model is trained, the loss is calculated according to the loss function, and then back-propagation (BP) is used to adjust the parameter adjustment. In this paper, the Categorical Cross-Entropy Loss function is used as the loss function of the model.

$$L(Y, \hat{Y}) = -\sum_i y_i \times \log \hat{y}_i \tag{12}$$

- Activation function: The formula of the ReLU activation function is as follows. This function can satisfy the sparsity in bionics. It activates the units when the input value is higher than a certain number, and can quickly converge in the stochastic gradient descent algorithm. The gradient of the function is 0 or constant, which can alleviate the problem of gradient disappearance, thereby improving the learning precision and speed of the neural network. Therefore, this paper uses ReLU as the activation function in two convolutional layers and two fully connected layers.

$$relu(x) = \max(0, x) \tag{13}$$

At the same time, to prevent overfitting, a dropout layer is added to the network structure. The dropout layer prevents overfitting by preventing the synergy of certain features. At each training,

the units are randomly removed, allowing one unit to appear independent of the other, preventing features from interdepending and reducing the transmission of erroneous information.

5. Experimental Evaluation

In this section, we first describe the dataset used in the experiments. Next, we present an experiment to prove a certain capability of our method. Finally, we compare our method with previous work in many respects.

5.1. DGA Data Set

The OSINT DGA feed from Bambenek Consulting [24] was collected as DGA domains. Then, we filtered out the classes with more than 5000 for training, and finally there were a total of 765,091 DGA domains. At the same time, the first one million domain names of the Alexa [25] website were collected as normal domains. Therefore, a dataset was generated, including normal domains and DGA domains, with a total 1,675,404, as shown in Table 2.

Table 2. The details of the dataset.

DGA	Amount
banjori	439,223
Post	66,000
tinba	65,603
ramnit	47,510
necurs	32,768
qakbot	20,000
murofet	14,260
pykspa	14,215
ranbyus	13,960
simda	13,681
shiotob/urlzone/bebloh	12,521
dyre	7998
Cryptolocker	6000
nymaim	6000
locky	5352
Alex	910,313

5.2. Experimental Results

This paper performed random sampling to generate sets for our evaluations. First, the sample set was randomly divided into 10 parts on average, and one of them was taken as a test set. One of the remaining nine samples was used as a verification set, and the remaining eight were used as a training set. The ratio of the final training set, verification set, and test set is 8:1:1. After training, the test set was used to test the classification model. The results of the experiment are shown in Table 3.

It can be seen from Table 3 that the average precision rate is 95.05%, the average recall rate is 95.14%, and the average F_1 score is 95.48%. The accuracy rate is 95.14%, which is very high. The confusion matrix for the LSTM multiclass classifier with attention mechanism is show in Figure 4. Blocks in the figure represent the fraction of domains belonging to the DGA families on the vertical axis, classified as DGA families on the horizontal axis, where 0 is depicted as white and 1 depicted as black. As can be seen in Figure 4, a large number of Cryptolocker DGAs, Locky DGAs, and Necurs DGAs are classified as Ramnit. We analyzed these DGAs and found that they are very similar. Remi Cohen [26] pointed out that the Ramnit acquired parts of the Zeus code and became a banking trojan after the Zeus source code leak. Limor Kessem [27] supposed that Necurs operators were linked to the very centrum of Zeus elite in the early days, whereas Tom Spring [28] told us that the Locky ransomware roared back to life via Necurs botnet. We could infer that there is some relationship among Cryptolocker,

Locky, Necurs, and Ramnit; that is why some DGAs of Cryptolocker, Locky, and Necurs are classified as Ramnit.

Table 3. Experimental results of our method.

| Domain Type	Precision	Recall	F_1 Score	Support
nymaim	0.3988	0.1115	0.1743	601
ranbyus	0.4672	0.8455	0.6018	1346
murofet	0.7641	0.7207	0.7418	1443
pykspa	0.8972	0.7207	0.7994	1393
locky	0.0000	0.0000	0.0000	574
shiotob	0.9751	0.9251	0.9494	1268
banjori	0.9998	1.0000	0.9999	43,808
necurs	0.6651	0.1722	0.2735	3241
Cryptolocker	0.1000	0.0018	0.0035	562
simda	0.9264	0.9669	0.9462	1418
dyre	1.0000	1.0000	1.0000	797
Post	0.9994	0.9998	0.9996	6644
tinba	0.9259	0.9920	0.9578	6498
qakbot	0.7862	0.5013	0.6122	1973
ramnit	0.4688	0.7525	0.5777	4856
Alex	0.9898	0.9956	0.9927	91,119
avg/total	0.9505	0.9514	0.9458	167,541
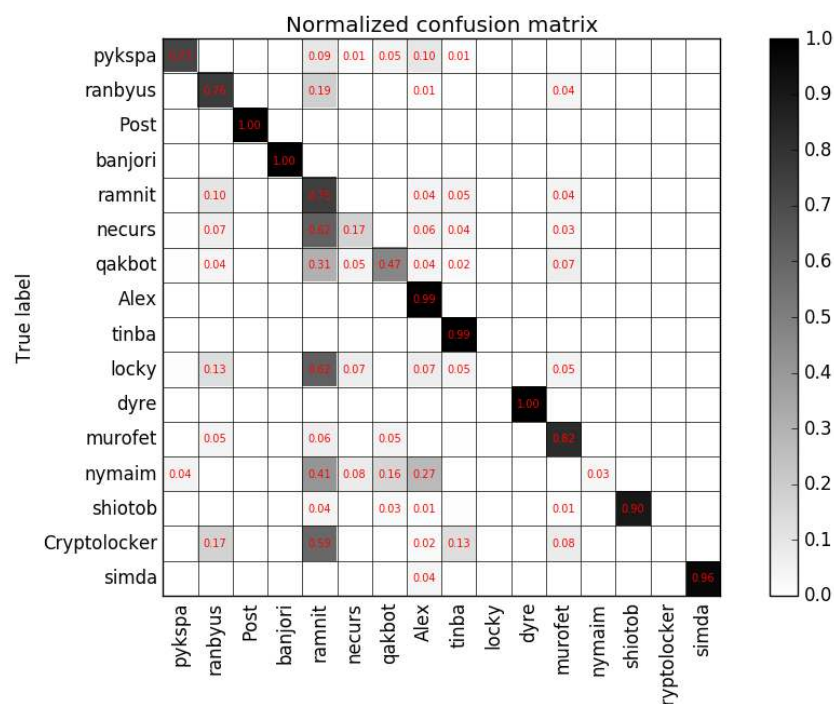
Figure 4. Confusion Matrix for the long short-term memory (LSTM) model with attention mechanism.

5.3. Work Comparison and Discussion

Yadav et al. [8] proposed a DGA domain name detection method. This work was inspired by the observation that the difference in character distribution between the normal domain name and the DGA domain name is quite large. They used the characters in the domain name and the distribution characteristics of the 2-gram character set, and finally used the edit distance and Jaccard distance [13] algorithm. Although Woodbridge et al. [16] proposed a method for DGA detection and classification using LSTM algorithm in 2016, they only detected and classified the strings of domain names, and it has been verified by experiments to have very good detection and classification effects. Compared

with the method in this paper, the attention mechanism is lacking. We compare our method with Yadav et al. [8] and Woodbridge et al. [16], and the results are shown in Table 4:

Table 4. Experimental results of Woodbridge et al. [16].

Domain Type	Yadav et al. [8]			Woodbridge et al. [16]			Our Method			Support
	Precision	Recall	F_1 Score	Precision	Recall	F_1 Score	Precision	Recall	F_1 Score	
Alex	0.9612	0.9863	0.9736	0.9865	0.9970	0.9917	0.9956	0.9956	0.9927	91,119
banjori	0.9741	0.9889	0.9814	0.9998	1.0000	0.9999	1.0000	1.0000	0.9999	43,808
Cryptolocker	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0018	0.0018	0.0035	562
dyre	0.9820	1.0000	0.9909	0.9975	1.0000	0.9987	1.0000	1.0000	1.0000	797
locky	0.0294	0.0038	0.0067	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	574
murofet	0.5139	0.4861	0.4996	0.7441	0.7477	0.7459	0.7207	0.7207	0.7418	1443
necurs	0.2609	0.0253	0.0461	0.6225	0.1725	0.2701	0.1722	0.1722	0.2735	3241
nymaim	0.1080	0.0409	0.0593	0.3621	0.1048	0.1626	0.1115	0.1115	0.1743	601
Post	0.9897	0.9913	0.9905	0.9995	1.0000	0.9998	0.9998	0.9998	0.9996	6644
pykspa	0.7253	0.5690	0.6377	0.9407	0.6827	0.7912	0.7207	0.7207	0.7994	1393
qakbot	0.7160	0.3483	0.4686	0.7970	0.4774	0.5971	0.5013	0.5013	0.6122	1973
ramnit	0.3541	0.5580	0.4333	0.4711	0.7360	0.5745	0.7525	0.7525	0.5777	4856
ranbyus	0.0000	0.0000	0.0000	0.4599	0.8522	0.5974	0.8455	0.8455	0.6018	1346
shiotob	0.9349	0.6584	0.7727	0.9873	0.9211	0.9531	0.9251	0.9251	0.9494	1268
simda	0.8203	0.6839	0.7459	0.9688	0.8533	0.9074	0.9669	0.9669	0.9462	1418
tinba	0.6709	0.8588	0.7533	0.9264	0.9912	0.9577	0.9920	0.9920	0.9578	6498
avg/total	0.8960	0.9127	0.9001	0.9482	0.9504	0.9445	0.9505	0.9514	0.9458	167,541

It can be seen from Table 4 that our method has a much higher precision, recall, and F_1 score compared with the work of Yadav et al. [8]. Compared with the work of Woodbridge et al. [16], the improvement of the method is not obvious.

Therefore, we conducted a 10-fold cross-validation experiment to compare the method of Woodbridge et al. [16] with our method. The sample set was divided into 10 parts on average, and one of them was taken as a test set each time. One of the remaining nine samples was used as a verification set, and the remaining eight were used as a training set. The ratio of the final training set, verification set, and test set is 8:1:1. Each experiment was carried out for 20 epochs of training. After training, the test set was used to test the classification model. The results of 10 experiments are shown in Figure 5. As can be seen in Figure 5, the precision of our method is superior than Woodbridge's method [16] in each experiment. This proves that the weights of different characters in different positions in the DGA domain name are different, and the attention mechanism is necessary for DGA classification.
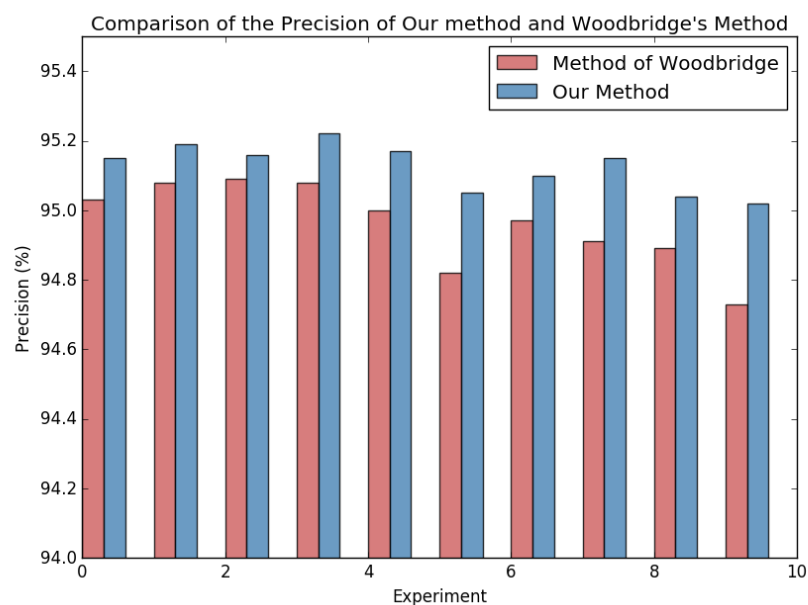

Figure 5. Comparison of the precision of our method and Woodbridge's method.

6. Conclusions

At present, there are many problems in DGA domain name classification. For example, it is difficult to obtain DGA algorithm by reversing malware samples, and the WHOIS information caused by the implementation of GDPR can no longer be easily obtained. Based on the research of DGA algorithm and DGA domain name, we propose a DGA domain name classification method based on LSTM with attention mechanism. This method no longer reverses the malware sample, nor does it use context information such as WHOIS of the domain name, and only uses the character sequence of the domain name. For the character sequence of the domain name, each domain name is converted into a matrix of fixed dimensions by padding and embedding. Then it use the LSTM with attention mechanism algorithm to construct the DGA domain name classification model to achieve fast and accurate classification of domain names. The experimental results show that combining the attention mechanism with the LSTM can effectively classify DGA domain names. Therefore, the DGA domain name is associated with the network attack, and the response time of the attack incident is shortened. The method takes into account the weights of different characters in different positions in the DGA domain name and has higher classification accuracy than the simple LSTM algorithm.

Author Contributions: Conceptualization, Y.Q. and B.Z.; methodology, Y.Q.; software, Y.Q.; validation, Y.Q.; formal analysis, W.Z.; investigation, Y.Q. and H.W.; resources, B.Z.; data curation, Y.Q. and H.W.; writing—original draft preparation, Y.Q.; writing—review and editing, B.Z. and A.K.S.; visualization, H.W.; supervision, B.Z. and W.Z.; project administration, W.Z. and B.Z.; funding acquisition, B.Z. and W.Z.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. McGrath, D.K.; Gupta, M. Behind Phishing: An Examination of Phisher Modi Operandi. In Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, San Francisco, CA, USA, 15 April 2008; USENIX Association: Berkeley, CA, USA, 2008; pp. 4:1–4:8.
2. Ma, J.; Saul, L.K.; Savage, S.; Voelker, G.M. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; ACM: New York, NY, USA, 2009; pp. 1245–1254. [CrossRef]
3. Felegyhazi, M.; Kreibich, C.; Paxson, V. On the Potential of Proactive Domain Blacklisting. In Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More, San Jose, CA, USA, 17 April 2010; USENIX Association: Berkeley, CA, USA, 2010; p. 6.
4. Bilge, L.; Kirda, E.; Kruegel, C.; Balduzzi, M. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In Proceedings of the 18th Network and Distributed System Security Symposium, San Diego, CA, USA, 6 February 2011; Internet Society: Reston, VA, USA, 2011; pp.1–17.
5. Canali, D.; Cova, M.; Vigna, G.; Kruegel, C. Prophiler: A Fast Filter for the Large-scale Detection of Malicious Web Pages. In Proceedings of the 20th International Conference on World Wide Web, Hyderabad, India, 28 March–1 April 2011; ACM: New York, NY, USA, 2011; pp. 197–206. [CrossRef]
6. Zhang, J.; Saha, S.; Gu, G.; Lee, S.; Mellia, M. Systematic Mining of Associated Server Herds for Malware Campaign Discovery. In Proceedings of the 2015 IEEE 35th International Conference on Distributed Computing Systems, Columbus, OH, USA, 29 June–2 July 2015; pp. 630–641. [CrossRef]
7. GReAT. Winnti. More than Just a Game. 2013. Available online: https://securelist.com/analysis/internal-threats-reports/37029/winnti-more-than-just-a-game/ (accessed on 27 June 2019).
8. Yadav, S.; Reddy, A.K.K.; Reddy, A.L.N.; Ranjan, S. Detecting Algorithmically Generated Domain-Flux Attacks With DNS Traffic Analysis. *IEEE/ACM Trans. Netw.* **2012**, *20*, 1663–1677. [CrossRef]

9. Antonakakis, M.; Perdisci, R.; Nadji, Y.; Vasiloglou, N.; Abu-Nimeh, S.; Lee, W.; Dagon, D. From Throw-away Traffic to Bots: Detecting the Rise of DGA-based Malware. In Proceedings of the 21st USENIX Conference on Security Symposium, Bellevue, WA, USA, 8–10 August 2012; USENIX Association: Berkeley, CA, USA, 2012; p. 24.

10. Antonakakis, M.; Perdisci, R.; Dagon, D.; Lee, W.; Feamster, N. Building a Dynamic Reputation System for DNS. In Proceedings of the 19th USENIX Conference on Security, Washington, DC, USA, 11–13 August 2010; USENIX Association: Berkeley, CA, USA, 2010; p. 18.

11. Yadav, S.; Reddy, A.K.K.; Reddy, A.N.; Ranjan, S. Detecting Algorithmically Generated Malicious Domain Names. In Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, Melbourne, Australia, 1–3 November 2010; ACM: New York, NY, USA, 2010; pp. 48–61. [CrossRef]

12. Krishnan, S.; Taylor, T.; Monrose, F.; McHugh, J. Crossing the threshold: Detecting network malfeasance via sequential hypothesis testing. In Proceedings of the 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Budapest, Hungary, 24–27 June 2013; pp. 1–12. [CrossRef]

13. Jain, A.K.; Dubes, R.C. *Algorithms for Clustering Data*; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1988.

14. Bilge, L.; Sen, S.; Balzarotti, D.; Kirda, E.; Kruegel, C. Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains. *ACM Trans. Inf. Syst. Secur.* **2014**, *16*, 14:1–14:28. [CrossRef]

15. Schiavoni, S.; Maggi, F.; Cavallaro, L.; Zanero, S. Phoenix: DGA-Based Botnet Tracking and Intelligence. In *Detection of Intrusions and Malware, and Vulnerability Assessment*; Dietrich, S., Ed.; Springer International Publishing: Cham, Switzerland, 2014; pp. 192–211.

16. Woodbridge, J.; Anderson, H.S.; Ahuja, A.; Grant, D. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. *arXiv* **2016**, arXiv:1611.00791.

17. Yu, B.; Gray, D.L.; Pan, J.; Cock, M.D.; Nascimento, A.C.A. Inline DGA Detection with Deep Networks. In Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW), New Orleans, LA, USA, 18–21 November 2017; pp. 683–692. [CrossRef]

18. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]

19. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]

20. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.

21. Graves, A.; Wayne, G.; Danihelka, I. Neural Turing Machines. *arXiv* **2014**, arXiv:1410.5401.

22. Luong, M.T.; Pham, H.; Manning, C.D. Effective approaches to attention-based neural machine translation. *arXiv Preprint* **2015**, arXiv:1508.04025.

23. Cheng, J.; Li, D.; Lapata, M. Long Short-Term Memory-Networks for Machine Reading. *arXiv* **2016**, arXiv:1601.06733.

24. Bambenek, J. OSINT Feeds from Bambenek Consulting. 2019. Available online: http://osint.bambenekconsulting.com/feeds/ (accessed on 10 September 2019).

25. Alex. Keyword Research, Competitive Analysis, & Website Ranking | Alexa. 2019. Available online: https://www.alexa.com/ (accessed on 10 September 2019).

26. Cohen, R. Banking Trojans: A Reference Guide to the Malware Family Tree. 2019. Available online: https://www.f5.com/labs/articles/education/banking-trojans-a-reference-guide-to-the-malware-family-tree (accessed on 3 September 2019).

27. Kessem, L. The Necurs Botnet: A Pandora's Box of Malicious Spam. 2017. Available online: https://securityintelligence.com/the-necurs-botnet-a-pandoras-box-of-malicious-spam/ (accessed on 3 September 2019).

28. Spring, T. Locky Ransomware Roars Back to Life Via Necurs Botnet. 2017. Available online: https://threatpost.com/locky-ransomware-roars-back-to-life-via-necurs-botnet/125156/ (accessed on 3 September 2019).