future internet

MDPI

Article

N-Trans: Parallel Detection Algorithm for DGA Domain Names

Cheng Yang, Tianliang Lu *, Shangyi Yan, Jianling Zhang and Xingzhan Yu

College of Information and Cyber Security, People's Public Security University of China, Beijing 100038, China; 2020211438@stu.ppsuc.edu.cn (C.Y.); 201621440006@stu.ppsuc.edu.cn (S.Y.); zhangjianling@ppsuc.edu.cn (J.Z.); 2020211477@stu.ppsuc.edu.cn (X.Y.)

* Correspondence: lutianliang@ppsuc.edu.cn

Abstract: Domain name generation algorithms are widely used in malware, such as botnet binaries, to generate large sequences of domain names of which some are registered by cybercriminals. Accurate detection of malicious domains can effectively defend against cyber attacks. The detection of such malicious domain names by the use of traditional machine learning algorithms has been explored by many researchers, but still is not perfect. To further improve on this, we propose a novel parallel detection model named N-Trans that is based on the N-gram algorithm with the Transformer model. First, we add flag bits to the first and last positions of the domain name for the parallel combination of the N-gram algorithm and Transformer framework to detect a domain name. The model can effectively extract the letter combination features and capture the position features of letters in the domain name. It can capture features such as the first and last letters in the domain name and the position relationship between letters. In addition, it can accurately distinguish between legitimate and malicious domain names. In the experiment, the dataset is the legal domain name of Alexa and the malicious domain name collected by the 360 Security Lab. The experimental results show that the parallel detection model based on N-gram and Transformer achieves 96.97% accuracy for DGA malicious domain name detection. It can effectively and accurately identify malicious domain names and outperforms the mainstream malicious domain name detection algorithms.

Keywords: malicious domain name; DGA; parallel detection model; N-gram; Transformer model; N-Trans

Citation: Yang, C.; Lu, T.; Yan, S.; Zhang, J.; Yu, X. N-Trans: Parallel Detection Algorithm for DGA Domain Names. *Future Internet* **2022**, *14*, 209. https://doi.org/10.3390/fi14070209

Academic Editor: Filipe Portela

Received: 28 May 2022
Accepted: 12 July 2022
Published: 13 July 2022

1. Introduction

People's lives are increasingly inseparable from the Internet. By the end of 2021, the number of China's Internet users reached 1.032 billion, with 42.96 million new Internet users compared with December 2020, and the Internet penetration rate reached 73.0% [1]. However, the application of the Internet has enriched people's lives while also increasing security risks. When accessing a website, the domain name entered is resolved to an IP address through a DNS service, which locates the target server to browse the corresponding Web service. Because the domain name system has no security detection mechanism, malware and viruses often use DNS services to communicate with external servers [2].

Malicious domain names are widely used as communication carriers for malware, viruses, and malicious servers by organizations engaged in hacker attacks and APT. In order to meet the attacker's attack needs, prevent the malicious domain names from being discovered by security vendors, and disrupt the attack behavior, attackers often use the DGA algorithm to generate a large number of malicious domain names, which can hide the real malicious server.

The existing malicious domain name detection algorithms can be improved [3]. For example, machine learning algorithms are efficient for training, but require a lot of work up front to find the right features. The recognition rate of detecting malicious domain names still needs to be improved by using deep learning algorithms, such as LSTM and RNN. Given this, a parallel detection model named N-Trans based on N-gram and Transformer is proposed in this paper, and the main work and contributions are as follows:

1. In the data processing stage, flag bits are added at the beginning and end of the domain name, and the dataset is processed using the N-gram algorithm. The location features are added to the features of the phrase elements obtained from the original N-gram processing;
2. In the model training stage, the Transformer model is used to identify the malicious domain name, further enhancing the model's learning of deep features such as location information and text features and improving the accuracy of identification;
3. A parallel combination of the N-gram algorithm and Transformer model to further enrich the text features trained by the model based on a single detection model;
4. Experiments are conducted on the Alexa and 360 Security Lab datasets to compare with the Naive Bayesian algorithm, XGBoost algorithm, RNN model, LSTM model, Bi-LSTM model, etc. The experiments show that the accuracy of the algorithm proposed in this paper reaches 96.97%, and the recall rate reaches 97.07%, which is better than the comparison models.

The arrangement of the paper is as follows: Section 2 introduces the research related to malicious domain detection from machine learning and deep learning perspectives, leading to the model proposed in this paper based on the existing research. Section 3 introduces the N-Trans parallel detection model from data pre-processing and the parallel module. Section 4 presents the experimental environment and dataset details and verifies the effectiveness of the N-Trans model in detecting malicious domain names. Section 5 summarizes the work of this paper and proposes directions for future research.

2. Related Work

Machine learning is commonly used for character features to detect malicious domain names. Based on the difference in character distribution between legitimate and malicious domain names, Yadav et al. [4] detected malicious domain names by looking at the distribution of unigram and bigram features in the same set of IP addresses. Agyepong et al. [5] detected DGA based on a frequency analysis of character distribution and domain name weighted scores. Furthermore, Wang et al. [6] proposed a method based on random forest to identify malicious domain names using features such as domain name length. Meanwhile, Shi et al. [7] combined character features, DNS features, and WHOIS information of websites to identify malicious domain names using an extreme machine learning (ELM) model. The traditional machine learning approach of extracting character features to analyze malicious domain names is laborious and takes too much time [3]. Furthermore, the features extracted by this method may be avoided by DGA users. For example, the dictionary-based domain name generation algorithm generates malicious domain names that are not easily detected by traditional methods.

Deep learning methods are more complex than traditional machine learning methods models, which can mine deeper features of the dataset and derive new features from a limited set of features in the training set. Zhao et al. [8] used the word-hashing technique [9] to convert domain names into binary syntax strings, mapped the domain name to a high-dimensional vector space using a bag-of-words model, and built a five-layer deep neural network to train classification detection on the domain name. The traditional RNN algorithm is difficult to solve in the case of gradient diffusion in malicious domain name experiments. Xu et al. [10] used a bidirectional recurrent neural network to extract effective semantic features and used a recurrent network to effectively solve the problem of gradient diffusion and gradient explosion, which greatly improved the operation efficiency of the algorithm.

It is challenging to detect longer malicious domains using RNN, such as *emotet* and *ranbyus*. To solve this problem, many scholars use LSTM models to detect malicious domains and try to find more features in domain names. Ghosh et al. [11] used an improved LSTM model by adding ALOHA (auxiliary loss optimization for hypothesis augmentation) [12] to the traditional LSTM model to increase the accuracy of detecting malicious domain names based on domain generation. Xu et al. [13] used word vector embedding to

encode characters in domain names and fused convolutional neural networks and long and short-term memory networks. They trained malicious domain name detection models to solve the problems of sparse matrices and dimensional disasters in one-hot encoding, which improved the algorithm's efficiency. Zhang et al. [14] used a CNN network to extract the features of character combinations in domain names to avoid the sparse distribution of features in N-gram and then used LSTM to mine the contextual information in domain name strings. Wang et al. [15] constructed BiLSTM-CNN, a hybrid model of bidirectional long- and short-term memory neural networks and convolutional neural networks, and implemented domain names classification in their self-built dataset. Zhang et al. [16] used LSTM to improve the encoding of the Transformer model to better capture the character location information to solve the high false-positive rate on abbreviated domain names. This model can effectively distinguish between DGA and abbreviated domain names. Although the above scholars achieved better detection results, according to the characteristics of the LSTM model, it cannot mine the location features of characters between domain names, and the effect of domain names with low randomness still needs to be improved. Thus, the N-Trans model is proposed to detect malicious domain names, which can mine the combination and location relationship between characters and train the model in parallel with higher efficiency than the model only based RNN or LSTM.

3. N-Trans

In order to further improve the accuracy of identifying malicious domain names, this paper proposes a detection model based on parallel N-gram and Transformer. It digs deeply into the location information and text features of malicious domain names.

The model first uses the *tldextract* module in Python to process domain names, retaining the main part that can uniquely identify the domain name, and then adds "#" as a flag bit to the beginning and end of each domain name. The domain names are processed in parallel using a combined N-gram and Transformer algorithm, the number of parameters is reduced by a global average pooling layer, the network connections are deactivated by a dropout layer, and the detection results are obtained using sigmoid after activation using the Relu layer.

The detection model is shown in Figure 1.

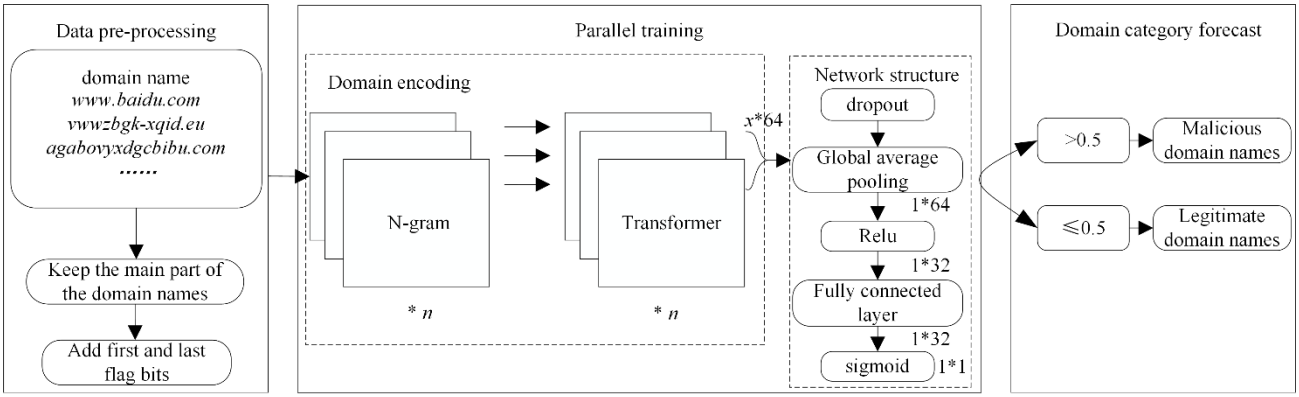


Figure 1. Parallel detection model. *x* is the number of features obtained. It describes the process by which the model detects malicious domains. The model is divided into three stages: data preprocessing, parallel training, and domain name category prediction.

3.1. Data Pre-Processing

The dataset is first processed using the *tldextract* module in Python; we remove the top-level domain and hostname and reserve them for the second-level domain of the domain names. After that, a symbolic flag bit (e.g., "#") is added to each string's first and last part using the word-hashing technique. Although adding the sign bit increases the number of word-hashing vectors after N-gram processing, the increased word-hashing vectors contain

information about the first and last positions of the domain names' character combinations, which enriches the dimensionality of the extracted features. Taking the domain name "www.baidu.com" as an example, processing is shown in Figure 2.

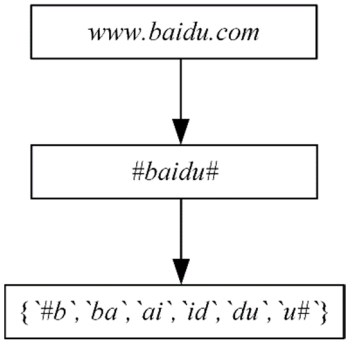


Figure 2. Word-hashing to process "*www.baidu.com*".

For the malicious domain name detection problem, after adding the first and last flag bits, N-gram encoding reduces the loss of memory to a certain extent when building deep learning. The feature model does not need to be modified as the samples increase.

3.2. Parallel Training

Using the parallel combination of N-gram and Transformer, we obtain richer textual features and increase the variety of phrase elements in the features more than using a single N-gram algorithm. This model eliminates the need to select and process features compared to common machine learning detection algorithms. Due to the multi-head attention mechanism in Transformer, the extracted features are more focused on inter-textual character features than deep learning models, such as CNN and LSTM, which improves the effectiveness of detecting malicious domain names.

After experimental validation, a parallel model combining N-gram (N = 2, 3, 4, it represents that bigram, trigram and 4-g methods are used in the model to process domain names.) and Transformer is selected for processing, and the features are spliced and input to the subsequent network for training. The parallel model is shown in Figure 3.

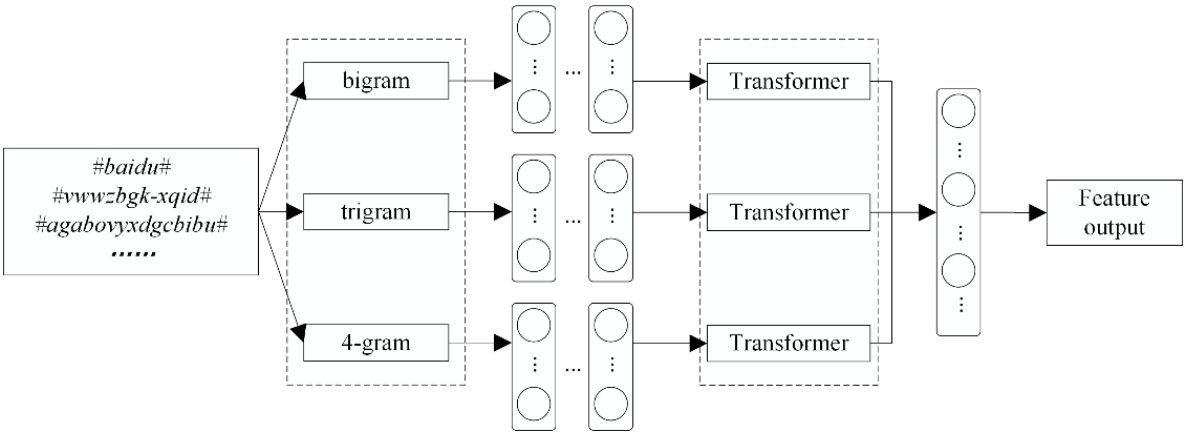


Figure 3. Parallel model structure.

3.2.1. Domain Encoding

In natural language processing problems, one-hot encoding is often used to convert text information into vector form that can be used directly by deep learning, but the converted vectors are very sparse. Facing the sparse matrix generated by one-hot encoding, the word2vec technique is usually used to map the high-dimensional vector space to the low-dimensional vector space. Because the sample units of malicious domain names are

in strings and have no contextual environment, the word2vec technique cannot reduce the dimensionality of the space after converting malicious domain name samples into vectors. The number of phrase elements encoded by the N-gram algorithm is limited, and its dimensionality is not too large. Compared with the huge sparse matrix generated using the one-hot encoding method, the N-gram algorithm can improve resource utilization.

N-gram is a statistical language modeling algorithm that implements "associative" behavior in the process of natural language processing. The N-gram algorithm considers the occurrence of a word to be dependent on several other words, so it can be used to determine whether the composition of a sentence is reasonable. The more information obtained, the more accurate the predicted information is. For a deterministic string, the N-gram algorithm processes all substrings of the string of length N. For the domain name "*www.linkedin.com*", when $N = 2$, the process is shown in Figure 4.

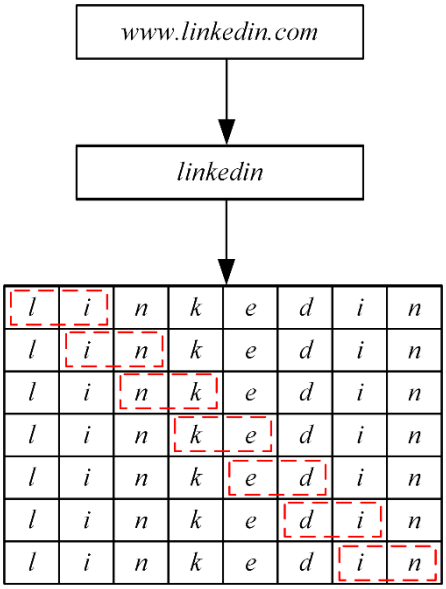


Figure 4. N-gram ($N = 2$) algorithm to process "*linkedin.com*".

The N-gram algorithm is used in malicious domain detection with letters as the smallest unit. To avoid redundancy in the N-gram results, only second-level domains are used as the model input, and top-level domains are ignored. Suppose we now have a domain name $s = (w_1, w_2, \cdots, w_n)$ consisting of n letters; each letter w_i is related to the first letter w_1 to its preceding letter w_{n-1}, as shown in Equation (1).

$$\begin{aligned} p(S) &= p(w_1 w_2 \cdots w_n) \\ &= p(w_1) p(w_2|w_1) \cdots p(w_n|w_{n-1} \cdots w_2 w_1) \end{aligned} \tag{1}$$

Equation (1) introduces too many variables in the operation process, and for the N-gram model, the results obtained are very sparse when N is large. To solve this problem, the probability of occurrence of the ith letter w_i is only related to the first N letters of the current letter by using the assumption of the Markov chain, and the computation of Equation (1) is drastically reduced by the assumption, which is shown in Equation (2).

$$\begin{aligned} p(w_1 w_2 \cdots w_n) &= \prod p(w_i|w_{i-1} \cdots w_1) \\ &\approx \prod p(w_i|w_{i-1} \cdots w_{i-N+1}) \end{aligned} \tag{2}$$

When $N = 1$, it is equivalent to the N-gram method to conduct word frequency statistics, and the common values of N are 2 and 3. Equation (3) shows the process of calculating the frequency statistics for binary phrase elements with $N = 2$ as an example.

$$\begin{aligned} p(S) &= p(w_1 w_2 \cdots w_n) \\ &= p(w_1)p(w_2|w_1) \cdots p(w_n|w_{n-1}) \end{aligned} \tag{3}$$

3.2.2. Transformer Model

The pre-processed phrase elements are fed into the Transformer model. The transformer model [17] is the encoder–decoder model that incorporates the attention mechanism. The common encoder–decoder framework is implemented using RNN networks, but is usually inefficient because its reliance on sequential order features prevents training from progressing in parallel. In the Transformer model, the data can be processed in parallel, and both input and output are from the same sequence, which captures the global information well.

This paper uses the encoder part of the Transformer structure for model training. The parameters of the Transformer structure are as follows: the input dimension of the token is the number of samples (the token is the key parameter in the Transformer model and each domain name is token), and the output dimension is 64; the input dimension of the position part is the number of N-gram-processed features, and the output dimension is 64; the head number of the multi-head Attention is five. There is no sequential structure in the Transformer model like in RNN, so the position information of the elements is needed, and the position information is calculated as shown in Equation (4).

$$P(pos, n) = \begin{cases} \sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}), n = 2i \\ \cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}), n = 2i + 1' \end{cases} \tag{4}$$

where pos is the position of the letter in the domain string, i is the dimension of the vector, and d_{model} is the dimension of *token*, which is processed using the cos function when i is odd and the sin function when i is even.

The core of the Transformer is the multi-head attention mechanism, which consists of several self-attentive mechanisms. The self-attention mechanism is a Q(*Query*)-vector of a phrase element in the domain name and K(*Key*)-vectors of other phrase elements in the domain name multiplied one at a time to get the initial weights. After processing, it is multiplied with V(*Value*) to obtain the sum of weights, as shown in Equation (5).

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \tag{5}$$

where d_k is the dimension of K, and the *softmax* function normalizes the result. N-gram treats domain names as collections of phrase elements. In the study of malicious domain name detection, Q is the phrase element in the domain name, K is the other phrase element in the domain name, and V is the vector representation of the phrase element after N-gram processing. Multiple linear mappings of the inputs Q, K, V are performed to obtain the attention $head_i$, as shown in Equation (6).

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V). \tag{6}$$

$W_i{}^Q$, $W_i{}^K$, and $W_i{}^V$ are the parameter matrices when linearly varied. The final result of multi-head attention is obtained by splicing the multiple $head_i$ for linear variation, as shown in Equation (7).

$$MultiHead(Q, K, V) = Concat(head_1, \cdots, head_n)W^o. \tag{7}$$

In Equation (7), n denotes the number of heads in the multi-head attention mechanism, and W^o is the linearly varying parameter matrix.

3.2.3. Network Structure

The network model processed by the Transformer are shown in Figure 1. Here, the features after Transformer processing are averaged using global averaging pooling for all features of each channel, and the obtained average is transported to the 64-dimensional channel. After the global averaging pooling layer, it can effectively reduce the parameters during the model processing and suppress the overfitting during the model training. The obtained results are randomly deactivated after the dropout layer, randomly reducing some neurons with a 5% probability.

The results go through the 32-dimensional Relu activation function layer, retaining the part of the features with values > 0, and changing the values of features < 0 to 0, removing useless information and data noise. The depth information of the features is not changed, which accelerates the network learning process and simplifies the model. Finally, the features are fed into the fully connected layer, and the prediction results are obtained after the sigmoid function operation.

4. Experiment and Analysis

4.1. Dataset and Test Environment

The dataset used in this paper includes legitimate domain names and malicious domain names. The legitimate domain names are from the collection of the top 1 million domain names collected by Alexa [18]. The malicious domain names are from different DGA families of malicious domain names publicly collected by the 360 Network Security Lab [19] such as *nymaim*, *necurs,* and other well-known malicious domain name families.

The experiments in this paper are based on the deep learning framework of *keras*, and the back-end uses the *Tensorflow* module. The specific experimental environment is shown in Table 1.

Table 1. Experimental environment configuration.

Environment Configuration	Parameters
Operating System	Ubuntu 16.04.12
Memory/GB	128 GB
CPU	2.2 GHz Inter (R)
Python	3.8.10
Tensorflow	2.5.0

4.2. Data Analysis

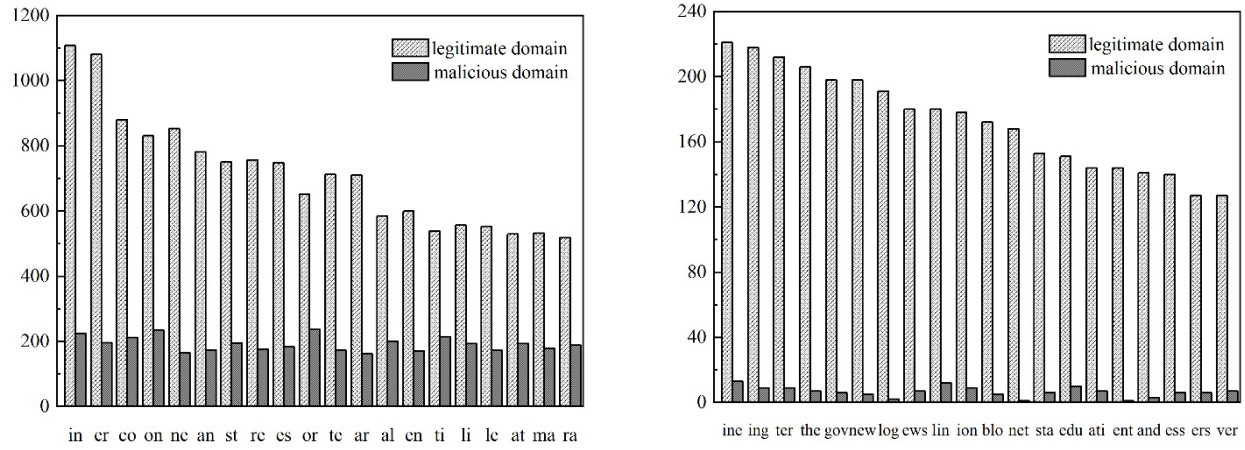
Ten thousand legitimate and ten thousand malicious domain names were randomly selected as the dataset for the experiment. For each domain name, because the characteristics of legitimate and malicious domain names at the first level are not obvious, the first-level domain was discarded and constituted the dataset D. The training set and test set in the experiment were partitioned using the *sklearn* module at 40% for the test set. After mixing the malicious domain names from 360 Network Security Lab, 10,000 data were randomly selected, and each family had no more than 1400 malicious domains.

The selected families of malicious domain names and the number of them are shown in Table 2. Refer to Vranken [20] for a detailed description of the dataset, and refer to the DGALab study to classify the types of DGA. The column 'DGA type' indicates whether the DGA is Static DGA (Static), Seed-based (Seed), or Date-based (Date) [21]. The column 'Number' indicates the count of domain names; the column 'Length' indicates the length (min, max) of the second-level domain names. The last two columns show examples of second-level domain names.

Table 2. Characteristics of DGA families.

DGA Family	DGA Type	Number	Length	Sample1	Sample2
simda	Static	921	8–14	digivehusyd	lymylur
virut	Static	869	6	yvvioe	zuzmoq
rovnix	Static	1092	18	rc7thuhy8agn43zzgi	lryja5lrm835m7byr8
suppobox	Seed	1282	9–18	sharmainewestbrook	arivenice
tinba	Seed	1400	12	nvfowikhevmy	oykjietwrmlw
banjori	Seed	1400	15–26	earnestnessbiophysicalohax	iutcererwyatanb
gameover	Date	791	20–28	14dtuor1aubbmjhgup7915tlinc	2id0lapmam6w1799w7315zaqj5
nymaim	Date	1400	5–12	zzayzoabsi	msfctioj
necurs	Date	845	7–21	wiyqgyiwgm	otenbmgbpuskiasvehxm

For dataset *D*, there were 1417 phrase elements after bigram processing, 31,220 phrase elements after trigram processing, 127,469 phrase elements after 4-g processing, and 138,636 phrase elements after 5-g processing. The N-gram algorithm was used for the legitimate and malicious domain names in *D*. The distribution of the phrase elements of the domain names was obtained as shown in Figure 5a,b.



(**a**) (**b**)

Figure 5. Distribution of legitimate and malicious domain phrase element frequencies. After using N-gram processing, the phrase elements were sorted from highest to lowest frequency. The frequency of the top 20 ranked phrase elements in legitimate domain names and those corresponding to malicious domain names were selected for comparison. (**a**) The processing result when $N = 2$. (**b**) The processing result when $N = 3$.

From the N-gram distribution ($N = 2$ or 3) results of legitimate and malicious domain names, there was a great difference in the frequency of phrase elements of malicious and legitimate domain names. The distribution of phrase elements of malicious domain names was more uniform, while the distribution of phrase elements of legitimate domain names varied greatly, so the phrase elements generated after bigram segmentation had better domain name classification characteristics. When $N = 4$ or 5, legitimate domain names still had a good frequency distribution of phrase elements. Malicious domain names have an increased variety of phrase elements, become more random, and correspond to fewer phrase elements.

The numerical characteristics of the phrase elements after N-gram algorithm processing are analyzed. The data analysis uses mean, variance, skewness, and kurtosis as indicators of numerical characteristics to analyze the results of N-gram processing for two types of domain names. The results are shown in Table 3.

Table 3. N-gram algorithm results analysis. The mean value is an indicator of the concentration trend of the distribution of the phrase elements; the variance shows the trend of the number of phrase elements; the skewness shows whether the distribution of the phrase elements of the domain name is symmetrical; the kurtosis can get the difference of the data distribution of the domain name phrase elements data compared with the normal distribution.

N-Gram	Digital Features	Legitimate	Malicious
$N = 1$	Mean	2007.154	4190.865
	Variance	2146.498	2152.575
	Skewness	1.0524	−0.87408
	Kurtosis	0.1087	−1.19031
$N = 2$	Mean	62.24157	110.4813
	Variance	127.3123	87.4684
	Skewness	3.588488	0.18239
	Kurtosis	16.4035	−1.62526
$N = 3$	Mean	6.768033	4.553918
	Variance	14.68101	3.503929
	Skewness	6.177821	0.848823
	Kurtosis	55.35392	0.086178
$N = 4$	Mean	2.286315	1.13609
	Variance	4.317711	0.428017
	Skewness	12.90284	1.241187
	Kurtosis	303.786	2.659457
$N = 5$	Mean	1.519426	1.024079
	Variance	2.262212	0.249301
	Skewness	16.31627	11.2395
	Kurtosis	445.1127	128.8666

From Table 3, it can be concluded that, when $N = 1$, it is equivalent to performing the frequency statistics of letters, ignoring the order relationship between letters. When $N = 2$, 3, 4, and 5, the phrase element variance of legitimate domain names is larger than that of malicious domain names. The smaller the value, the more uniform the distribution of domain name phrase elements, which is a randomly generated domain name. The larger the variance, the more the frequency distribution of the phrase elements deviates from the mean, which is more likely a domain name generated by human intervention. The skewness of phrase elements of legitimate domain names is larger than that of phrase elements of malicious domain names. In terms of data distribution, malicious domain names have better symmetry than legitimate domain names. In terms of kurtosis, the data distribution of malicious domain names fits more closely to the normal distribution than the data distribution of legitimate domain names.

Taking the data after bigram processing as an example, the experiment uses the word-hashing technique to add the flag bit "#" at the beginning and end of each second-level domain name, and the number of data elements increases from 1417 to 1486. The increase of the first and last flag bits marks part of the location information of the domain name, which enriches the features in the recognition process without causing significant redundancy. Therefore, the data pre-processing method of adding the first and last flag bits and using the N-gram algorithm can effectively extract the character frequency and character location information features of domain names.

4.3. Experimental Results and Analysis

To validate the results of the proposed parallel detection model based on the N-gram algorithm and Transformer to identify malicious domain names, four sets of experiments were designed based on the same dataset.

1. The optimal parallel model was selected to detect malicious domain names by changing the parallelism and combining N-gram algorithms in the parallel detection model;

2. By changing the head value in the Transformer model, the appropriate number of heads was selected as a parameter in the model training;
3. In order to verify the detection effect of the model proposed in this paper, the Naive Bayesian and XGBoost machine learning algorithms and RNN, LSTM, and Bi-LSTM models in deep learning were selected to compare with the model designed in this paper;
4. Through ablation experiments, we compare the number of features, training time, and evaluation metrics to analyze the effects of adding sign bits and L1 regularization in a single detection model and compare the effects of adding sign bits in a parallel detection model.

The following experiments are implemented in the *keras* deep learning framework, *Tensorflow* back-end, and *sklearn* module.

4.3.1. Evaluation Indicators

Accuracy (*Acc*) was used in the experiments to evaluate the classification accuracy of the algorithm for domain names, and *recall* was used to evaluate the model's classification of malicious domain names. The accuracy formula is shown in Equation (8).

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}, \tag{8}$$

where *TP* denotes the number of malicious domain names predicted to be malicious, *FN* denotes the number of malicious domain names predicted to be legitimate, *TN* denotes the number of legitimate domain name predicted to be legitimate, and *FP* denotes the number of legitimate domain names predicted to be malicious.

The formula for calculating the *recall* is shown in Equation (9).

$$recall = \frac{TP}{TP + FN}. \tag{9}$$

From Equation (9), we can see the total number of malicious domain names in the dataset, so the *recall* can calculate the model's effectiveness in detecting malicious domain names. The detection effects of different network models will also be compared using the *precision* and *F1-score*.

4.3.2. Parallel Detection Model Construction Experiments

In order to select a malicious domain detection model with better results, possible combinations of N-gram algorithms in the proposed parallel detection model are analyzed. In the single N-gram experiment when $N \geq 3$, although more features can be captured and the ability to predict characters is significantly improved, a large number of sparse matrices are generated in the model. This can lead to distortion of the calculated probabilities, and a large parameter space can cause a dimensional disaster; thus, the model will have a considerable time and memory loss during training, resulting in the experiment not being carried out smoothly. Based on the original N-gram and Transformer detection models, the features are optimized using a feature selection algorithm based on L1 regularization. After feature gain processing, in addition to achieving the filtering feature purpose, the high-dimensional matrices are also dimensionally reduced, which reduces the experimental time and memory loss, making the experiments feasible.

Different combinations of N were tested to explore the most suitable N-gram combination and Transformer composition for the parallel detection model. The experimental procedure is shown in Table 4.

Table 4. Parallel detection model combination. Experiments were performed for possible N-Trans combinations. The experiments compared the number of features obtained from model training, training time, accuracy, and recall. *N* was too large ($N > 5$) to cause the algorithm to proceed properly, so the corresponding experiments were not performed.

The Values and Combinations of N	Characteristic Number of Gain	Training Time/s	*Acc*	*Recall*
1	35	13.3100	0.9201	0.9210
2	**441**	92.1600	**0.9604**	**0.9599**
3	106	32.7600	0.7591	0.8725
4	28	12.3700	0.5869	0.9955
5	12	8.6600	0.5322	1.0000
1,2	476	60.4457	0.9542	0.9613
1,3	141	27.6050	0.9020	0.8907
1,4	63	13.2781	0.9218	0.9066
1,5	47	10.6808	0.9315	0.9034
2,3	547	82.7185	**0.9573**	**0.9556**
2,4	469	57.7871	0.9353	0.9493
2,5	453	55.7214	0.9403	0.9461
3,4	134	26.2685	0.7603	0.9032
3,5	118	22.5284	0.7637	0.9015
4,5	40	9.6938	0.5820	0.9997
1,2,3	582	63.1049	0.9537	0.9388
1,2,4	504	58.0790	0.9483	0.9070
1,2,5	488	61.4247	0.9535	0.9393
1,3,4	169	31.1952	0.9125	0.9373
1,3,5	153	25.2881	0.9243	0.9264
1,4,5	75	14.7553	0.9145	0.8659
2,3,4	**575**	77.4251	**0.9697**	**0.9707**
2,3,5	559	69.9106	0.9417	0.9257
2,4,5	481	57.6231	0.9472	0.9142
3,4,5	146	25.4514	0.7630	0.9113
1,2,3,4	610	83.7084	0.9465	0.9445
1,2,3,5	594	67.2344	0.9395	0.9040
1,2,4,5	516	53.0182	**0.9582**	0.9479
1,3,4,5	181	29.2953	0.9105	0.9430
2,3,4,5	587	90.5120	0.9368	**0.9578**
1,2,3,4,5	622	73.3525	**0.9545**	**0.9573**

The experimental results show that, for a single N-gram algorithm, the number of phrase elements in the domain names must be reduced because the basic units of the phrase elements are expanding, and the number of phrase elements matching the same category in the domain names must be reduced as the N-value changes. Therefore, the effective features obtained from training are reduced. Taking "*linkedin*" as an example, there are 749 "*li*" phrase elements in the dataset. After trigram processing, there are 192 "*lin*" phrase elements in the dataset, which is less than the result of binary phrase element processing. At the same time, the accuracy of the trained model decreases, and the recall increases, indicating overfitting.

The features extracted by a single algorithm are expanded for the parallel detection model by stitching the features from different N-gram algorithms and Transformer processing. As can be seen from Table 4, the parallel detection model, including bigram, has better detection results and achieves the best results at $N = 2$, 3, and 4. However, it is not the case that the more parallel approaches are included, the better the detection results are. It is found that the model detection accuracy and recall decrease when 5-g is included, but then the detection results become better again if $N = 1$ is included in the model. The domain location and text features are extracted more comprehensively after learning with the multi-head attention mechanism. The model combined with bigram had improved accuracy and recall over the single bigram-based model.

After comparing the experimental results, the model using the parallel combination of bigram, trigram, 4-g, and Transformer at N = 2, 3, 4 was more effective in identifying malicious domain names.

4.3.3. Multi-Head Attention Head Comparison Experiment

In the Transformer model, the number of heads also affects the experiment, which is conducted by changing the heads value; the experimental results are shown in Figure 6.

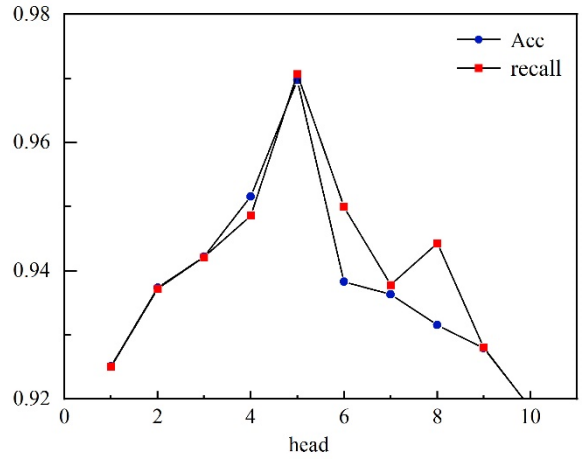


Figure 6. Observe the trends in *Acc* and *recall* by changing the number of heads in Transformer.

From Figure 6, we see that the model achieves the best results regarding accuracy and *recall* when the number of heads is five. At this time, changing the number of heads does not improve the detection effect of the model, and increasing the number of linear mappings to the feature matrix does not allow the extraction of effective features in the increased linear space.

According to the results of this experiment, the head value of five in the detection model optimizes the detection of malicious domain names.

4.3.4. Network Model Comparison Experiment

The Naive Bayesian algorithm, XGBoost algorithm, RNN model, LSTM model, Bi-LSTM model, and the parallel detection model proposed in this paper were selected for comparison. The experimental results are shown in Table 5.

Table 5. Comparison of model training algorithms. The parameters in the experiments are set as follows: the Naive Bayes algorithm and XGBoost algorithm both use features extracted by bigram for processing. The output dimension of the RNN hidden layer was 64, and the output dimension of the Relu activation function was 32; the output dimension of the LSTM hidden layer was 64; the loss rate of the dropout layer was 0.1, and the rest of parameters in the network are default values.

Model	Training Time/s	*Acc*	*Recall*	*Precision*	*F1-Score*
Naive Bayesian	23.2281	0.8000	0.7276	0.8976	0.8037
XGBoost	27.3884	0.9055	0.8666	0.9421	0.9028
RNN	327.4625	0.9152	0.9208	0.9105	0.9156
LSTM	353.3754	0.9252	0.9574	0.8996	0.9276
Bi-LSTM	370.1486	0.9395	0.9074	0.9695	0.9374
Parallel detection model with N = 2,3,4	77.4251	**0.9697**	**0.9707**	**0.9748**	**0.9727**

Machine learning algorithms, such as the Naive Bayesian algorithm and XGBoost algorithm, require human selection of features and the process takes a lot of time. The features extracted with bigram alone cannot accurately distinguish legitimate domain

names from malicious ones. After the word embedding process, RNN, LSTM, and Bi-LSTM improve the accuracy and *recall* compared to machine learning algorithms and focus more on the features of character frequency provided by word embedding. However, they lack the learning of similarity between phrase elements compared to the Transformer model, which introduces a multi-head attention mechanism. As can be seen from the *precision*, the N-Trans model predicts fewer errors in the results for malicious domains. In terms of the speed of feature learning, the Transformer model significantly improves the speed of training compared to the model that relies on the sequential relationship between elements due to its ability to operate in parallel. Combined with the F1 score, the N-Trans model can better distinguish malicious domain names from legitimate domain names compared with other network models.

Since the parallel detection model can effectively extract the location information and multi-text features of domain characters, and the model can capture more features among word combinations, it achieves better results in the malicious domain names classification problem.

4.3.5. Ablation Experiments

In order to further verify the impact of each part of the detection model on the experiments, the single N-gram and Transformer malicious domain name detection models proposed in this paper are first subjected to ablation experiments. Firstly, we use bigram and Transformer experiments, and to verify the effectiveness of word-hashing, we use the word-hashing processed dataset for malicious domain name detection. After that, we compare the models before and after feature gain by adding L1 regularization to verify the effect of feature gain on the model; the experimental results are shown in Table 6.

Table 6. Experimental analysis of single detection model.

Model	Number of Characteristics	Training Time/s	*Acc*	*Recall*
bigram + Transformer	1417	327.46	0.9370	0.9370
Add flag bit + bigram + Transformer	1486	353.37	0.9424	0.9484
Add flag bit + bigram + L1 Regularization + Transformer	**441**	**67.16**	**0.9604**	**0.9599**

When the L1 regularization was removed, the features processed by the parallel detection model increased significantly; the number of features obtained without adding the first and last flag bits was 160,106, and the number of features obtained after adding the first and last flag bits was 181,018, which means the experiment could not be carried out smoothly. Therefore, only the effect of adding the flag bits on the parallel detection model is verified. The ablation experiments were performed with or without adding the flag bits; the results are shown in Table 7.

Table 7. Experimental analysis of parallel detection model.

Model	Number of Characteristics	Training Time/s	*Acc*	*Recall*
L1 Regularization + Parallel Detection	547	73.9109	0.9413	0.9435
Add flag bit + L1 Regularization + Parallel Detection	**575**	**77.4251**	**0.9697**	**0.9707**

Tables 6 and 7 show that adding flag bits can increase the first and last position information in the domain names string, resulting in a small increase in the accuracy and *recall* of the model detection. After adding L1 regularization, the time for model training was significantly reduced; at the same time, the detection effect of the model was optimized, and the accuracy and *recall* of detection were further improved. The results

show that adding the first and last marker bits can enrich the features learned by the model; L1 regularization makes the experiment feasible, reduces the training time and memory consumption, and the features after L1 regularization gain can detect malicious domain names more accurately.

In summary, the parallel detection model based on the N-gram and Transformer framework proposed in this paper can effectively and accurately detect malicious domain names.

5. Conclusions and Future Work

In response to the shortcomings of existing malicious domain name detection models, a parallel detection model based on N-gram and Transformer is proposed, incorporating location information and multi-text features in domain name strings. For the malicious domain name detection problem, aspects of this framework still need to be improved, and the network model is optimized to make it more suitable for the malicious domain name application scenario and to improve the accuracy of identification. The detection model is improved to apply to the future multi-classification problem of malicious domain name families, as well as to explore the deeper reasons for the relationship between the combination of N and malicious domain name detection results. The analysis of numerical characteristics and the selection of evaluation metrics with reference to Iwendi et al. [22] will be applied to the model proposed in the model.

The next step will be to mine other deep features among letters, such as the distribution of adjacent consonants and the statistics of consonant–vowel combinations. The malicious domain names generated by wordlist have low randomness, are more similar to legitimate domain names, and will be studied in depth to detect malicious domain names generated by wordlist in the future. Furthermore, text generation algorithms [23] are continuously researched for adversarial generative networks, and whether the network structure can resist the attack of malicious samples is a challenge for future research.

Author Contributions: Conceptualization, C.Y. and S.Y.; methodology, C.Y. and S.Y.; software, C.Y. and J.Z.; validation, T.L. and X.Y.; investigation, T.L., J.Z. and X.Y.; resources, C.Y. and X.Y.; data curation, S.Y.; writing—original draft preparation, C.Y.; writing—review and editing, T.L., J.Z. and X.Y.; visualization, C.Y., S.Y. and X.Y.; supervision, C.Y. and T.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the 2020 Fundamental Research Funds for the National Key R&D Program (2020AAA0107700) and the 2020 Fundamental Research Funds for the Central Universities of PPSUC (No. 2020JKF101).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository.

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

References

1. China Internet Network Information Center. The 47th Statistical Report on the Development Status of the Internet in China. Available online: http://www.cnnic.cn/hlwfzyj/hlwxzbg/hlwtjbg/202202/P020220407403488048001.pdf (accessed on 1 April 2022).
2. Jiang, J.; Zhuge, J.W.; Duan, H.X.; Wu, J.P. Research on botnet mechanisms and defenses. *J. Softw.* **2012**, *23*, 82–96. [CrossRef]
3. Li, K.; Yu, X.; Wang, J. A Review: How to Detect Malicious Domains. In Proceedings of the International Conference on Artificial Intelligence and Security, Dublin, Ireland, 19–23 July 2021; Springer: Cham, Switzerland, 2021; pp. 152–162.
4. Yadav, S.; Reddy, A.K.K.; Reddy, A.N.; Ranjan, S. Detecting algorithmically generated malicious domain name names. In Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, Melbourne Australia, 1–30 November 2010; ACM: New York, NY, USA, 2010; pp. 48–61.
5. Agyepong, E.; Buchanan, W.J.; Jones, K. Detection of algorithmically generated malicious domain name. In Proceedings of the 6th International Conference of Advanced Computer Science and Information Technology, Zurich, Switzerland, 24–25 June 2017; Computer Science and Information Technology: Omaha, NE, USA, 2018; pp. 91–111.
6. Wang, H.K.; Zhang, X.D.; Yang, W.Y.; Ma, Z.C.; Liao, P.; Huang, Y.B.; Yu, X.W.; Zhang, D.; Xia, W.; Song, W.J. Random Forest Based DGA Domain Name Name Detection Method: China. CN105577660A, 11 May 2016.

7. Shi, Y.; Chen, G.; Li, J. Malicious domain name name detection based on extreme machine learning. *Neural Process. Lett.* **2018**, *48*, 1347–1357. [CrossRef]
8. Zhao, K.J.; Ge, L.S.; Qin, F.L.; Hong, X.G. Deep model for DGA botnet detection based on word-hashing. *J. Southeast Univ. Nat. Sci. Ed.* **2017**, *47*, 30–33.
9. Huang, P.S.; He, X.; Gao, J.; Deng, L.; Acero, A.; Heck, L. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, San Francisco, CA, USA, 27 October–1 November 2013; ACM: New York, NY, USA, 2013; pp. 2333–2338.
10. Xu, Y.; Yan, X.; Wu, Y.; Hu, Y.; Liang, W.; Zhang, J. Hierarchical bidirectional RNN for safety-enhanced b5g heterogeneous networks. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 2946–2957. [CrossRef]
11. Ghosh, I.; Kumar, S.; Bhatia, A.; Vishwakarma, D.K. Using auxiliary inputs in deep learning models for detecting DGA-based domain name names. In Proceedings of the 2021 International Conference on Information Networking, Jeju Island, Korea, 13–16 January 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 391–396.
12. Rudd, E.M.; Ducau, F.N.; Wild, C.; Berlin, K.; Harang, R. ALOHA: Auxiliary Loss Optimization for Hypothesis Augmentation. In Proceedings of the 28th USENIX Security Symposium, Santa Clara, CA, USA, 14–16 August 2019; USENIX: Berkeley, CA, USA, 2019; pp. 303–320.
13. Xu, G.T.; Sheng, Z.W. DGA malicious domain name name detection method based on fusion of CNN and LSTM. *Netinfo Secur.* **2021**, *21*, 41–47.
14. Zhang, B.; Liao, R.J. Malicious domain name name detection model based on CNN and LSTM. *J. Electron. Inf. Technol.* **2021**, *43*, 2944–2951.
15. Wang, T.T.; Liu, X.F. A Phased Malicious Domain Name Detection Algorithm. *J. Chin. Comput. Syst.* **2021**, 1–6. Available online: http://kns.cnki.net/kcms/detail/21.1106.TP.20210818.1358.051.html (accessed on 25 December 2021).
16. Zhang, X.; Cheng, H.; Fang, Y.Q. A DGA domain name name detection method based on Transformer. *Comput. Eng. Sci.* **2020**, *42*, 411–417.
17. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; pp. 5998–6008.
18. Amazon Web Services, Inc. AWS–Alexa Top Sites—Up-to-Date Lists of the Top Sites on the Web. 2021. Available online: https://aws.amazon.com/alexa-top-sites/ (accessed on 10 December 2021).
19. DGA–Netlab OpenData Project. 2022. Available online: https://data.netlab.360.com/dga/ (accessed on 10 December 2021).
20. Vranken, H.; Alizadeh, H. Detection of DGA-Generated Domain Names with TF-IDF. *Electronics* **2022**, *11*, 414. [CrossRef]
21. Chailytko, A.; Trafimchuk, A. DGA clustering and analysis: Mastering modern, evolving threats. *J. Cybercrime Digit. Investig.* **2015**, *1*.
22. Iwendi, C.; Ibeke, E.; Eggoni, H.; Velagala, S.; Srivastava, G. Pointer-based item-to-item collaborative filtering recommendation system using a machine learning model. *Int. J. Inf. Technol. Decis. Mak.* **2022**, *21*, 463–484. [CrossRef]
23. Wu, J.; Lu, T.L.; Du, Y.H. Generation of malicious domain name training data based on improved Char-RNN model. *Netinfo Secur.* **2020**, *20*, 6–11.