applied sciences

MDPI

Article

Detection of Algorithmically Generated Malicious Domain Names with Feature Fusion of Meaningful Word Segmentation and N-Gram Sequences

Shaojie Chen [1,*], Bo Lang [1,2], Yikai Chen [1] and Chong Xie [1]

[1] State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China; langbo@buaa.edu.cn (B.L.); yk_chen@buaa.edu.cn (Y.C.); xiechong@buaa.edu.cn (C.X.)
[2] Zhongguancun Laboratory, Beijing 100191, China
* Correspondence: chenshaojie@buaa.edu.cn

Abstract: Domain generation algorithms (DGAs) play an important role in network attacks and can be mainly divided into two types: dictionary-based and character-based. Dictionary-based algorithmically generated domains (AGDs) are similar in composition to normal domains and are harder to detect. Although methods based on meaningful word segmentation and n-gram sequence features exhibit good detection performance for AGDs, they are inadequate for mining meaningful word features of domain names, and the performance of hybrid detection of character-based and dictionary-based AGDs needs to be further improved. Therefore, in this paper, we first describe the composition of dictionary-based AGDs using meaningful word segmentation, introduce the standard deviation to better measure the word distribution features, and construct additional 11-dimensional statistical features for word segmentation results as a supplement. Then, by combining 3-gram and 1-gram sequence features, we improve the detection performance for both character-based and dictionary-based AGDs. Finally, we perform feature fusion of the above four kinds of features to achieve an end-to-end detection method for both kinds of AGDs. Experimental results showed that our method achieved an accuracy of 97.24% on the full dataset and better accuracy and F1 values than existing methods on both dictionary-based and character-based AGD datasets.

Keywords: AGD detection; meaningful word segmentation; n-gram; LSTM; feature fusion

Citation: Chen, S.; Lang, B.; Chen, Y.; Xie, C. Detection of Algorithmically Generated Malicious Domain Names with Feature Fusion of Meaningful Word Segmentation and N-Gram Sequences. *Appl. Sci.* **2023**, *13*, 4406. https://doi.org/10.3390/app13074406

Academic Editors: Tarek Gaber, Shu-Chuan Chu and Chin-Shiuh Shieh

Received: 27 February 2023
Revised: 17 March 2023
Accepted: 27 March 2023
Published: 30 March 2023

1. Introduction

The domain name system (DNS) is a decentralized system that is mainly used to build the mapping relationship between domain names and internet protocols (IPs) and to convert readily memorized domain names to numerical IP addresses. For network attacks such as botnets, to improve the reliability of the communication between the command and control (C&C) server and infected devices, attackers use the domain generation algorithm (DGA) to generate many random domain names, from which one or more domain names are selected to be registered as the domain names of the C&C server. With the use of many random domain names, i.e., algorithmically generated domains (AGDs), attackers may prevent communication from being blocked by blacklists to evade the detection of antiviruses. Therefore, DGA plays an important role in network attacks.

DGA usually generates a set of random strings with the help of random seeds and combines them with specific top-level domain names (TLDs) to generate a set of AGDs. Therefore, AGDs usually exhibit randomness in the character sequences. In addition, the infected devices attempt to establish connections with the C&C server by querying the randomly generated AGDs, which results in many nonexistent domains (NXDomain) during the requests. Compared with normal DNS traffic, DGA traffic exhibits abnormalities in both domain names and traffic behaviors. Therefore, common DGA attack detection

methods can be generally divided into two types: domain name-based methods and traffic-based methods. Domain name-based methods usually extract features such as character entropy, domain name length, ratio of vowels and consonants from a single DNS domain name [1–3] and use machine learning models [1–3], deep learning networks [4–6], or clustering methods [7,8] to measure the randomness of the characters in domain names and detect the AGDs. Common models used in these methods include support vector machines (SVMs) [1,9], random forests (RFs) [1,10], recurrent neural networks (RNNs) [4,5], convolutional neural networks (CNNs) [6,11], K-means [7] and DBSCAN [8]. Traffic-based methods [12–16] usually obtain the traffic of a network device within a period and detect DGA traffic by analyzing the behavior characters, such as the ratio of the NXDomain or the similarity of queried domain names. Traffic-based methods are not limited by the training samples and may achieve good detection effects on new DGA attacks. However, in both the training and detection processes, they must buffer traffic for a certain period, which makes it difficult to achieve real-time detection and makes the training process relatively complicated. For domain name-based methods, the detection unit is usually a single domain name, and only a domain name dataset are needed when training the model. These methods achieve a better effect in real-time detection and are currently a popular topic of research in DGA attack detection. Since domain name-based detection methods mainly aim to detect AGDs, in this paper, AGD detection has the same meaning as domain name-based detection of DGA attacks.

Traditional AGDs generated with random characters are quite different from normal domain names, making them easy to detect. Therefore, to generate AGDs more like normal domain names, attackers have developed the dictionary-based DGA (or wordlist-based DGA). In the dictionary-based DGA, a dictionary or wordlist is first established and then random domain names are generated by concatenating words selected from the dictionary with random seeds. Such AGDs appear to have practical semantics, which is challenging for domain name-based detection methods.

There are currently 5 main types of methods for dictionary-based AGD detection. (1) Graph methods for discovering AGD words [9,17–19]: In terms of word selection, dictionary-based DGAs tend to use different words from normal domain names. Therefore, dictionary-based AGDs can be found by constructing word relationship graphs. This type of method is usually used for the detection of dictionary-based AGDs, and like traffic-based methods, a certain amount of data is required to perform detection, while a single domain name cannot be detected in real-time. (2) Methods based on meaningful word segmentation [10,20–24]: Like the randomness of characters in character-based AGDs, dictionary-based AGDs exhibit randomness in the words selected. Meaningful word segmentation methods split the concatenated string into a set of meaningful English words to obtain the component words of dictionary-based AGDs. Then, features, such as statistical features and word distribution features are extracted from the word sets and used for classification. This type of method performs well for dictionary-based AGDs, but the methods are usually based on some distribution features, such as the average mean pooling of word features [20] or the part-of-speech distribution features [25], without considering the word correlation in a domain name. Therefore, the detection performance of dictionary-based AGDs should be improved, and the detection performance of character-based AGDs is lower than that of other types of methods. (3) Methods based on n-gram sequences [10,26–32]: Methods of this type usually perform classification using the n-gram sequences obtained by the segmentation of the domain name by the n-gram algorithm and extracting statistical features or extract sequence features using deep neural networks such as RNN. Increasing the value of n improves the description of the character fragment sequence but increases the complexity exponentially. Therefore, the maximum value of n is usually 3. This kind of method can detect both character-based and dictionary-based AGDs, but it is insufficient to describe the characteristics of the component units (i.e., words) in dictionary-based AGDs. (4) Methods based on local feature extraction using CNN models [32–37]: Methods of this type use a CNN model to extract local features, which are similar to the features of n-grams. Then,

the sequence composed of local features is obtained, and sequence detection methods such as the long short-term memory (LSTM) model are used to perform classification. These methods can also detect both character-based and dictionary-based AGDs, but they cannot describe the word characteristics of dictionary-based AGDs. (5) Methods that introduce external information [38]: When extracting the features of domain names, some external features, such as WHOIS information, are extracted as auxiliary features for AGD detection. Methods of this type can also detect both kinds of AGDs. However, these methods usually require the internet for detection, making them difficult to use on offline devices.

In summary, most current AGD detection methods use meaningful word segmentation features or sequence features for detection. The word distribution features of domain names can be effectively used to detect dictionary-based AGDs; character sequence features based on n-grams can effectively detect character-based AGDs, and n-gram sequence features can also help improve the detection performance of dictionary-based AGDs. However, in previous studies, the methods are insufficient for mining meaningful word features of the domain names, and the performance of hybrid detection of character-based and dictionary-based AGDs needs to be further improved.

In this paper, we first enhance the meaningful word segmentation features based on previous studies and analyze the n-gram sequence features under different values of n. To improve the detection performance of dictionary-based AGDs, the word distribution features, statistical features and 3-gram sequence features of the domain names are used to describe the word features and sequence features of dictionary-based AGDs. The 1-gram sequence features of the domain names are used to ensure the detection of the character-based AGDs. For meaningful word segmentation features, we first calculate the context-sensitive word embedding features based on ELMo [39] after meaningful word segmentation, which is similar to the method of Koh et al. [20]. To measure the word correlations in a domain name, we calculate the standard deviation (Std) of each dimension of word features in addition to the mean value of each dimension, and the concatenation of mean and Std features are used as the final word distribution features. Statistical features are widely used in the detection of AGDs, and the part-of-speech distribution features of words are valuable for the detection of dictionary-based AGDs [25]; therefore, we incorporate 3 additional kinds of statistical features, with a total of 11 dimensions, including domain length features, character ratio features, and word statistical features. For the n-gram sequence features, to mutually ensure the detection effects of character-based and dictionary-based AGDs, we compare the performance of 1-gram, 2-gram, and 3-gram sequence features and demonstrate that the combination of 1-gram and 3-gram sequence features exhibit the best detection performance through experiments. Finally, we perform feature fusion of the 4 kinds of features and perform classification to achieve end-to-end detection of the two kinds of AGDs and effectively improve the detection performance.

In summary, our method makes the following contributions:

1. Based on meaningful word segmentation methods, we propose using the standard deviation and mean value of word embedding features to form the word distribution features. We also design multiple-dimensional statistical features and integrate n-gram sequence features to improve the detection performance of dictionary-based AGDs.

2. We determine the best combination of n-gram sequence features in experiments, use a 1-gram sequence to describe the character sequence features of character-based AGDs and use a 3-gram sequence to describe the character fragment sequence features of dictionary-based AGDs. By combining 1-gram and 3-gram sequence features, the detection effects of character-based and dictionary-based AGDs can be jointly considered.

3. Our method achieves accuracies of 96.33% and 98.64% in the dictionary-based AGD dataset and character-based AGD dataset, respectively, and achieves an accuracy of 97.24% in the mixed dataset, exhibiting significantly better performance than the existing methods.

The remainder of the paper is organized as follows. Section 2 summarizes the previous work related to DGA attack detection. Section 3 introduces the motivation and framework of the detection method based on the feature fusion of meaningful word segmentation and

n-gram sequences. In Section 4, we explain how the datasets were constructed and describe the application of the model to the datasets, comparing our method with the state-of-the-art method. Section 5 concludes the paper.

2. Related Work

In recent years, researchers have conducted many studies on DGA attacks. Most of the studies focused on detecting AGDs more effectively, while some studies focused on designing a more covert DGA that makes it harder to detect [40–45]. Detection methods can be divided into domain name-based detection methods and traffic-based detection methods according to their analysis units. Traffic-based methods divide the traffic into groups and classify the groups by extracting behavior features to detect DGA attack events [12–16]. The domain name-based methods only analyze the queried domain names in DNS traffic to detect AGDs. Common DGAs include arithmetic-based methods, hash-based methods, permutation-based methods and dictionary-based methods. The domain names generated by arithmetic-based methods, hash-based methods, and permutation-based methods usually exhibit randomness in character sequences, which are rarely meaningful in English. Here, the AGDs generated by the three DGA methods are combined and classified as character-based AGDs. The domain names generated by dictionary-based DGA methods are usually composed of English words, which are referred to as dictionary-based AGDs. The two kinds of AGDs are quite different and the features for detection must be designed accordingly. Due to the differences in character-based and dictionary-based AGDs, domain name-based detection methods are further divided into character-based AGD detection methods and dictionary-based AGD detection methods. Our method is a domain name-based detection method, and we mainly focus on the domain name-based detection methods in this section.

2.1. Domain Name-Based Methods for Character-Based AGDs

For character-based AGD detection, domain name-based methods could be divided into methods based on feature engineering, methods based on deep learning-based methods, and methods based on clustering, according to the model they use.

Methods based on feature engineering usually construct features manually according to the characteristics of AGDs and use machine learning models for detection [1–3]. Schüppen et al. [1] analyzed the domain names of the NXDomain, extracted three kinds of features, including structural features, linguistic features and statistical features, and used random forest and SVM models for classification. Sivaguru et al. [2] extracted 11 types of features from domain names, including character entropy, n-gram median and consecutive character ratio, and trained a binary classification model (B_RF) for AGD detection and two multiclass classification models (M-RF and OVA-RF) for family classification based on a random forest model. Feature engineering-based methods are difficult to implement in feature design, and attackers can design new DGA categories to avoid detection.

For methods based on deep learning [4–6], common detection models include RNNs and convolutional neural networks (CNNs). Woodbridge et al. [4] first proposed detecting AGDs through the traditional LSTM model. They used the LSTM model to realize binary classification for AGD detection and multiclass classification for DGA family classification. Later, Tran et al. [5] further proposed LSTM.MI model to address the problem of imbalanced samples by adding different weights to different DGA families. Similarly, Yu et al. [6] used LSTM, CNN models, etc., to extract features and perform classification based on the character sequences of the domain names.

In addition to supervised machine learning methods, some studies used clustering methods to detect AGDs [7,8]. Tong et al. extracted features such as bigram frequency, n-gram score, and entropy from the domain names and used K-means methods for clustering and detecting AGDs according to Mahalanobis distance. The PHOENIX method proposed by Schiavoni et al. [8] extracted a set of linguist features, used the DBSCAN model for clustering to obtain a domain name set, and detected AGDs by calculating the Mahalanobis

distance between the new domain name and the centroids of clusters. The accuracy of the clustering-based method depends on the effect of clustering, and the accuracy varies in different types of botnets.

The character-based LSTM method can effectively extract the character sequence features of the domain names. Therefore, we also used the LSTM model to detect character-based AGDs. In addition, some statistical features are valuable for AGD detection, and this information is difficult to be reflected in deep learning networks. Therefore, we propose multidimensional statistical features and fuse them with deep learning features to improve the effectiveness of the features.

2.2. Domain Name-Based Methods for Dictionary-Based AGDs

Currently, domain name-based methods for detecting dictionary-based AGDs are mainly divided into the following categories: (1) graph methods for discovering AGD words, (2) methods based on meaningful word segmentation, (3) methods based on n-gram sequences, (4) methods based on local feature extraction using CNN models, and (5) methods that introduce external information.

Dictionary-based DGAs tend to use different words from benign domain names, and dictionary-based AGDs can be found by constructing word relationship graphs [9,17–19]. Pereira et al. [17] first obtained words from the domain names and constructed a graph, named WordGraph, based on the co-occurrence relationship of words in the same domain name. In WordGraph, words in AGDs are usually used repeatedly, resulting in more degrees than words in benign domain names. Therefore, the authors used the connected components in WordGraph as units to extract the structural features, including degrees, to determine their categories. When testing, a domain name was classified according to whether it is an AGD by the word categories in the domain name. Shen et al. [18] also constructed a word graph based on the co-occurrence relationship of words and used the Infomap algorithm to perform community detection on the constructed word graph. Then, they used communities as units and classified them with the decision tree method. When testing, if all the words of a domain name belong to a word community, the domain name was classified as a dictionary-based AGD of this community. This type of method can adapt better to the detection of new AGDs, but like traffic-based DGA attack detection methods, these methods require many samples to build the graph, which makes it hard to detect a single AGD. This type of method is usually used for the detection of only dictionary-based AGDs and typically does not apply to character-based AGDs.

Since the components of dictionary-based AGDs are English words, some studies obtained the word sequences of a domain name based on meaningful word segmentation [10,20–24]. Koh et al. [20] used Wordninja [46] to perform meaningful word segmentation on domain names without TLD and calculated the context-sensitive word embedding features of each word through the ELMo [39] model. After calculating the mean pooling of the word features, the final features of the domain names are obtained and classified although fully-connected networks. Overall, the performance of methods based on meaningful word segmentation for character-based AGDs has been relatively poor. Some methods further split words on the basis of word segmentation, which is compatible with the detection of character-based AGDs. Zhou et al. [21] used WordSegment [47] methods [47] for meaningful word segmentation and constructed a word frequency dictionary. A word was further split into characters if the word was not in the frequency dictionary. After calculating the word embedding features of each word, a CNN model was used for classification. Methods based on meaningful word segmentation are intended to restore the smallest components of AGDs as much as possible; the detection performance is more affected by the word segmentation method, and the word sequences obtained are usually short, so it is difficult to extract the sequence features from them.

In addition, the n-gram algorithm is often applied to the detection of dictionary-based AGDs. The n-gram algorithm usually takes every n characters of the domain name as a window, extracts an n-gram sequence by the sliding window method, and performs

classification by sequence detection methods [10,26–32]. Xu et al. [26] first removed the TLD from the domain name and split it by the n-gram algorithm, in which n was set to 2 and 3. Then, each n-gram was encoded with one-hot encoding to form a feature matrix, and the CNN model was used for classification. Morbidoni et al. [27] extracted n-gram sequences and calculated embedding vectors for each n-gram, then used the LSTM model for binary classification, and further performed family classification for the domain names that were classified as AGDs. They used unlabeled samples to pre-train the n-gram embedding model, thereby solving the problem of insufficient labeled samples in the classifier training stage. Overall, the methods based on n-gram sequences usually deal with both character-based and dictionary-based AGDs at the same time. However, when encoding the n-grams, the number of distinct n-grams grows exponentially with the value of n, so usually, 3 is the maximum value of n. To address this problem, Selvi et al. [28] proposed a masked n-gram method, which first replaced the characters in the domain name with the 4 character types and used the n-gram algorithm for segmentation. In this way, the number of distinct n-grams was significantly decreased by the replacement, and a larger n-gram (i.e., 4 grams) was extracted. After constructing features based on the occurrence of each distinct n-gram, along with 18-dimensional lexical features, they used a random forest model for classification. However, the performance of methods based on the n-gram sequence was largely limited by the value of n, and it was insufficient for these methods to be able to describe the characteristics of the component unit (i.e., words) of dictionary-based AGDs.

Some researchers used a CNN model to extract local features to detect dictionary-based AGDs [32–37], which has features similar to those of n-grams, and can extract a larger range of local features than methods based on n-gram sequences. Yang et al. [33] first transformed domain names into feature sequences according to the character embeddings, extracted local features in parallel through a CNN network with multiple convolution kernels of different sizes, and extracted character sequence features through a bidirectional LSTM model. Then, the features from the CNN and bidirectional LSTM were concatenated for classification. Ren et al. [34,35] proposed the ATT-CNN-BiLSTM model based on CNN, bidirectional LSTM and the attention mechanism. Feature sequences were first obtained by character embeddings of domain names without TLD, and CNN layers were used to extract local features from the feature sequences. Then, sequence features were extracted by bidirectional LSTM layers, and an attention layer was used to allocate the corresponding weight. Finally, the features were input into fully-connected layers for classification. Overall, this kind of method can effectively extract the features from the sequence constructed by local features from the domain names. However, like n-gram sequence-based methods, these methods are insufficient for describing the characteristics of the words of dictionary-based AGDs. In addition to features from the domain names themselves, some methods obtain additional external information, such as WHOIS information [38], as supplementary data. However, the acquisition of this external information requires the use of the internet, which makes the methods difficult to use on offline devices.

Based on the above findings, we chose to combine the meaningful word segmentation and n-gram sequence features of the domain names to improve the detection effect of dictionary-based AGDs and take into account the detection effect of character-based AGDs by n-gram sequence features.

3. Method

3.1. Motivation

The domain names in normal DNS traffic can be regarded as semantical sequences, while the AGDs generated by the pseudorandom method often exhibit a certain degree of randomness. A domain name can be defined as “subdomain.2LD.TLD”, meaning the top-level domain (TLD) [48] is managed by the root nameserver, and the 2LD is the next-level label of TLD and is usually registered by a company or a group below a certain TLD. Therefore, the random string of an AGD usually appears in 2LD, and common AGDs do not

contain subdomain parts. However, since some domain names also support the registration of subdomains, such as "github.io", the random string of an AGD may also occur in the subdomain. Therefore, we mainly use the "subdomain.2LD" as the main detection unit. In addition, some TLDs named country code TLDs (ccTLDs) often have subdomains such as "com.cn", "net.uk", "gov.au", and "edu.co". Such 2LD.TLDs are not registered by a company or a group and are also regarded as TLDs. In this situation, these 2LD.TLDs are considered TLDs, and the 3rd level label of the domain name is considered 2LD. The 2LD.TLDs that can be regarded as TLDs are listed in [49].

The AGDs generated by character-based and dictionary-based methods are quite different and features for detection must be designed accordingly. Our method mainly considers two types of features: meaningful word segmentation features and n-gram sequence features. The overall design of our method is shown in Figure 1. For meaningful word segmentation features, since the components of a dictionary-based AGD are English words, the component units (i.e., words) of a dictionary-based AGD can be restored by meaningful word segmentation. However, the number of words obtained is relatively small (usually less than 10), making it difficult to extract the word sequence features; therefore, after the context-sensitive word embedding feature extraction, the word features are summarized to construct word distribution features. In addition, based on the domain names and the segmented words, we additionally extract 11-dimensional statistical features, including word part-of-speech distribution, to improve the detection ability for dictionary-based AGDs.

For n-gram sequence features, we use the n-gram algorithm with the LSTM model to characterize the morpheme correlation in domain names. The randomness of the character sequence is weak in dictionary-based AGDs and strong in character-based AGDs. Therefore, for sequence feature extraction of dictionary-based AGDs, we use the LSTM model of a 3-gram sequence that contains more characters in an n-gram. For sequence feature extraction of character-based AGDs, we use the LSTM model of a 1-gram sequence (i.e., character LSTM model) that contains fewer characters in an n-gram.

Finally, we fuse the 4 kinds of obtained features based on the fully-connected layers and perform classification to realize an end-to-end detection method for the detection of both character-based and dictionary-based AGDs.

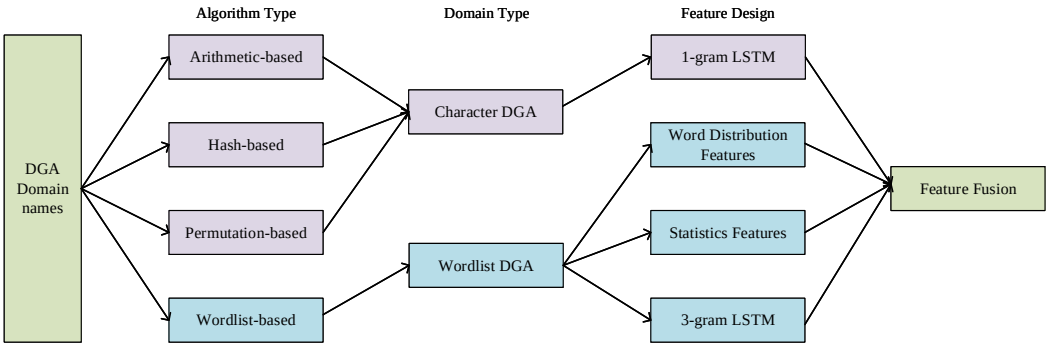

Figure 1. Overall concept of the detection method.

3.2. Model Structure

The structure of our AGD detection method is shown in Figure 2. Our method is divided into 3 parts: data preprocessing, feature extraction, and feature fusion and classification.

First, during data preprocessing, the TLD is removed from the domain name, and the remaining sections are mainly used as the basic unit for classification. We use Wordninja [46] for meaningful word segmentation and use n-gram algorithms to construct n-gram sequences for the domain name without TLD. Here, we construct 1-gram and 3-gram sequences, as discussed in Section 3.4.

The next step is feature extraction. For the extraction of meaningful word segmentation features, we first calculate the context-sensitive word embedding features using

ELMo [39] and summarize the word features to obtain 128-dimensional word distribution features, as discussed in Section 3.3. Next, based on the domain name and the segmented words, we extract 11-dimensional statistical features and transfer them into 8-dimensional statistical features, as discussed in Section 3.5. For n-gram sequence feature extraction, based on the results of 1-gram and 3-gram sequences, we use the LSTM model to calculate the 128-dimensional 1-gram and 3-gram sequence features, respectively, as discussed in Section 3.4. Finally, the four parts of features are concatenated, and 392-dimensional features (128 + 8 + 128 * 2) are obtained.

In the feature fusion and classification step, we use two fully-connected layers to fuse the 4 features and use the sigmoid activation function to calculate the classification result, which is a value in the interval [0, 1]. When the value is greater than 0.5, the domain name is classified as an AGD, and when the value is less than 0.5, the domain name is classified as a benign domain name.

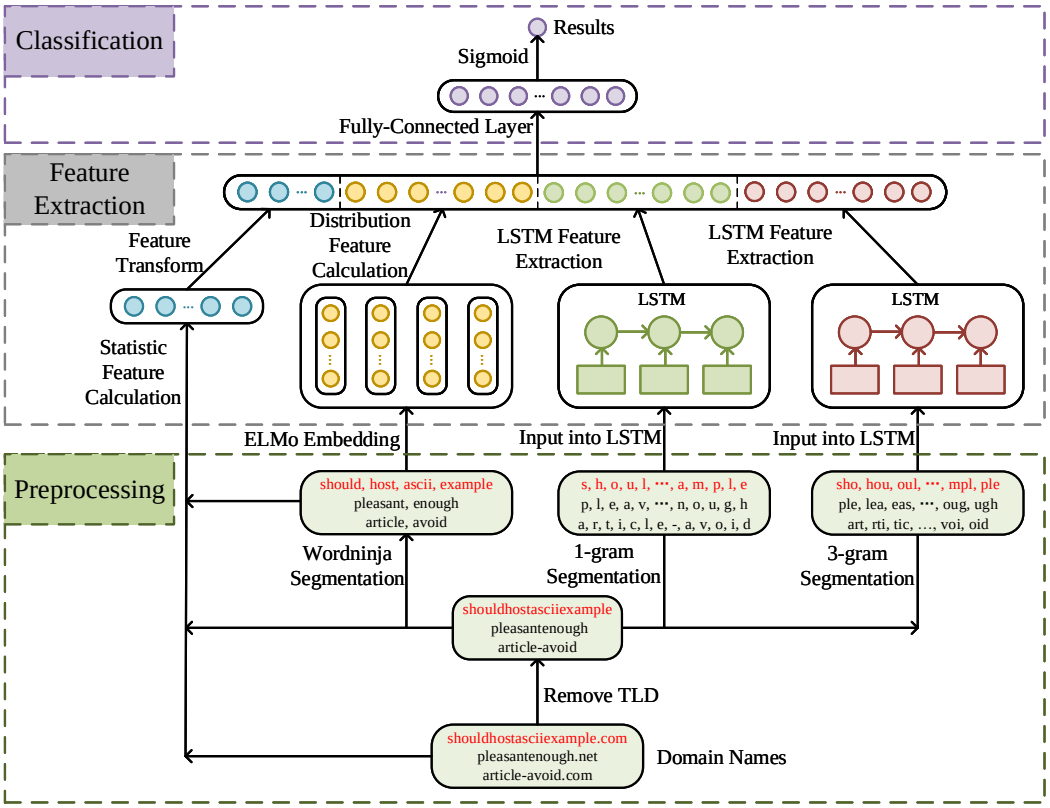

Figure 2. Structure of the AGD detection method.

3.3. Word Distribution Feature Extraction

This section describes how word distribution features are extracted based on meaningful word segmentation, which primarily focuses on the detection of dictionary-based AGDs. Since dictionary-based DGAs use English words to randomly generate the AGDs, we adopt the same approach as Koh et al. [20], by restoring the component words of a domain name through meaningful word segmentation to improve the detection effect of such AGDs. For English word segmentation methods, Wordninja [46] and WordSegment [47] are commonly used, and based on our experiments, we believe that Wordninja performs better in this scenario.

The process of word distribution feature extraction is shown in Figure 3. We first use Wordninja to obtain the word sequence of the domain name and use the ELMo [39] model to extract a 1024-dimensional context-sensitive word embedding feature vector for each word. For benign domain names, the words contained in a domain name usually have a certain semantic correlation, while for dictionary-based AGDs constructed by random concatenate words, the semantic correlation is relatively low. Therefore, we use

the correlation between words to detect the dictionary-based AGDs. Since the output of the ELMo model is a 1024-dimensional feature vector, we calculate the standard deviation (Std) of each dimension of the word features, which obtains 1024-dimensional Std features to measure the correlations. In addition, we extract the mean value of each dimension of word features to represent the distribution of the features, which is the same as the Koh method [20]. Since the original ELMo model is trained based on natural languages, it is difficult to fully adapt to the words in domain names, but if the ELMo model is iterated, it will result in a very large training time cost. Therefore, we add a fully-connected layer after ELMo to transform the output features of ELMo. After that, the mean features and Std features are calculated and concatenated to obtain 2048-dimensional distribution features. Last, through a fully-connected layer, the features are transformed into 128-dimensional feature vectors, which are regarded as the final word distribution features.

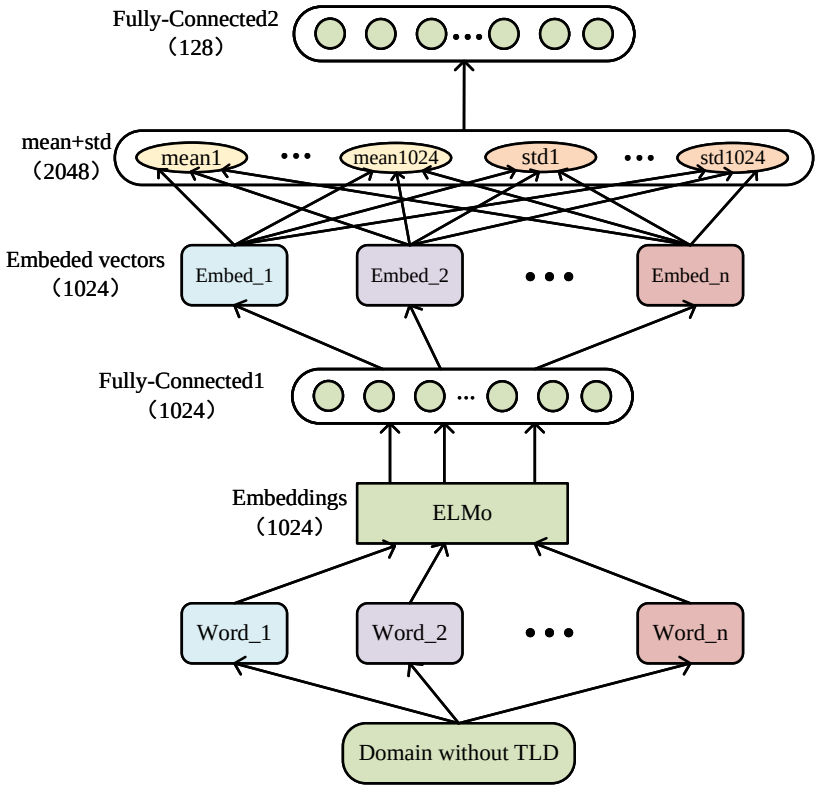

Figure 3. Process of word distribution feature extraction.

3.4. N-Gram Sequence Feature Extraction

This section describes how n-gram sequence features are extracted from the domain names using the LSTM model. After analysis, we select 1-gram and 3-gram sequence features; 1-gram sequence features are mainly used for the detection of character-based AGDs, and 3-gram features are mainly used for the detection of dictionary-based AGDs.

According to the RFC specification of the DNS [50], the common characters in the domain names are digits (0–9), letters (a–z and A–Z), hyphen ("-") and dot ("."). Since the domain name resolution process is case-insensitive, a total of 38 common characters are considered. In addition, some domain names in the dataset may contain underlines ("_"). To calculate the string embedding vector without throwing an exception, 39 valid characters are considered here. Therefore, the number of distinct 1-grams is 39, 2-grams may be 39^2 = 1521, 3-grams may be 39^3 = 59,319, and 4-grams may be 39^4 = 2,313,441. Since the LSTM model needs to calculate the embedding vector for each n-gram, there are too many distinct 4-grams, so only 1-grams, 2-grams, and 3-grams are considered here. When the value of n increases, the n-gram better describes the character fragment sequence, which is suitable for the detection of dictionary-based AGDs; in contrast, the

1-gram can better describe the character sequence, which is more suitable for the detection of character-based AGDs. Therefore, for character-based AGDs, the relative detection performance is as follows: 1-gram > 2-gram > 3-gram. For dictionary-based AGDs, the detection performance is as follows: 3-gram > 2-gram > 1-gram. This is demonstrated in experiments. We fuse 1-gram and 3-gram sequence features to detect both character-based and dictionary-based AGDs, as shown in Figure 2. We experimentally show that the fusion of 1-gram and 3-gram sequence features exhibits the best performance.

For each n-gram sequence, the character sequence of the domain name without TLD is split according to the step size of 1 character and then truncated or padded to a fixed length. According to the RFC specification of the DNS, the maximum length of each label in a domain name is 63, and the random string of an AGD mostly appears at the first label of the domain name, so we set the sequence length to 64. Last, based on the sequences obtained by 1-gram and 3-gram, we extract sequence features, although the LSTM model and output 128-dimensional features separately.

3.5. Statistical Feature Extraction

In the word distribution feature extraction, we consider the mean and Std of word features as the final word distribution features. In addition, we believe that the statistical values of segmented words and the distribution of part-of-speech are also of great value. Therefore, after meaningful word segmentation, we additionally extract statistical features to supplement the above deep learning features to enrich the feature types and improve the detection effect. We extract statistical features with a total of 11 dimensions from three perspectives: domain length features, character ratio features, and word statistical features.

The first type of feature includes domain length features. Since the main detection unit is the domain name without a TLD, the length of the original domain name and the length of the domain name without a TLD are considered here. The second type of feature is the character ratio feature. We use the ratio of vowel, consonant and digit characters in the domain without TLD as features. In addition, some dictionary-based DGAs use hyphens ("-") to concatenate the selected words from the dictionary (such as the Matsnu family), so we take the occurrence of hyphens "-" as one dimension of the features. The third type of feature is word statistical features. We mainly focus on the number of words and the average length of words after meaningful word segmentation. In addition, the part-of-speech distribution of words plays an important role in dictionary-based AGD detection [25], so we use the part-of-speech ratio of the words as features. The part-of-speech of words are extracted using the NLTK package of Python language. Since each domain name contains a small number of words, we only consider the ratio of adjectives (JJ), nouns (NN), and verbs (VB) and do not consider the subtypes of part-of-speech. For instance, adjectives (JJ), comparatives adjectives (JJR), and superlative adjectives (JJS) are all considered the same type.

Since the values of the four dimensions of the statistical features are integers, including the length of the original domain name, the length of the domain name without TLD, the number of words, and the average length of words, we divide them by the maximum value in their dataset for normalization. The values of the other 7-dimensional features already belong to the interval [0, 1], and there is no need for further normalization. Finally, we use a fully-connected layer to transform the 11-dimensional statistical features into 8-dimensional features. The details of the 11-dimensional statistical features are shown in Table 1.

Table 1. Summary of statistical features.

Feature Type	Feature Name	Description	Whether Need Normalization
Domain length features	Len_total	Total length of original domain name	Yes
	Len_no_tld	Total length of domain name without TLD	Yes
Character ratio features	Ratio_vow	Ratio of vowel characters in domain name without TLD	No
	Ratio_con	Ratio of consonant characters in domain name without TLD	No
	Ratio_dig	Ratio of digit characters in domain name without TLD	No
	If_contain _hypen	Whether to contain hyphen ("-"). 1 indicates yes and 0 indicates no	No
Word statistical features	Num_words	The total number of words after meaningful word segmentation	Yes
	Avg_word _len	The average length of words after meaningful word segmentation	Yes
	Ratio_JJ	The ratio of adjectives in a word after meaningful word segmentation	No
	Ratio_NN	The ratio of nouns in a word after meaningful word segmentation	No
	Ratio_VB	The ratio of verbs in a word after meaningful word segmentation	No

4. Experiments

4.1. Datasets and Indicators

This paper builds datasets based on Alexa [51] and UMUDGA [52]. Alexa contains nearly 1,000,000 benign domain names, and UMUDGA was released in 2020 and contains 50 families of AGDs, including 6 families with 11 variants of dictionary-based AGDs and 32 families with 39 variants of character-based AGDs. Domain names in some character-based DGA families are also similar to dictionary-based AGDs, such as Vawtrak_v3, but most of the words do not have English meanings, so they are also regarded as character-based AGDs. In our experiments, we regarded each variant as an independent DGA family.

Earlier papers usually used DGArchive [53] and Netlab 360 DGA [54] to construct their datasets, but these AGDs are derived from real network traffic, and the number of samples of each family is quite different in the two datasets, which makes them difficult to use. UMUDGA is a relatively standardized dataset that has been used for experiments in many papers since its release in 2020. Here, we selected 500,000 domain names from Alexa and 500,000 AGDs from UMUDGA (10,000 for each DGA family) to build our datasets. We constructed 3 datasets: a dictionary-based dataset that contained 210,000 samples, a character-based dataset that contained 790,000 samples, and a full dataset that contained 1,000,000 samples. For each dataset, 70% of the samples were used as the training set, 10% as the validation set, and the other 20% as the testing set. The validation set was used to estimate the fitting state of the model during the training stage. When the accuracy of the validation set decreased significantly, the epoch with the highest accuracy on the validation set was selected as the final model, and the performance of the model on the testing set was recorded as the final result.

In addition, we constructed 3 generalization testing sets based on the AGDs from DGArchive and Netlab 360 to measure the real-world detection performance of our final model and the three comparison methods, which were trained on the full dataset. First, we constructed 2 generalization testing sets based on the AGDs from DGArchive. We selected up to 5,000 AGDs from each family, based on whether or not the family was included in the UMUDGA, the AGDs were divided into two generalization testing sets. Then, we selected up to 1,000 AGDs for each family after October 2022 from Netlab 360 and built the third generalization dataset. The details of our final datasets are shown in Table 2.

Table 2. Descriptions of datasets.

Dataset	Composition	Description
Dictionary-based dataset	Total: 210,000 AGD: 110,000 Benign: 100,000	11 families of dictionary-based AGDs from UMUDGA and random selected 100,000 benign domain names from Alexa
Character-based dataset	Total: 790,000 AGD: 390,000 Benign: 400,000	39 families of character-based AGDs from UMUDGA and random selected 400,000 benign domain names from Alexa
Full dataset	Total: 1,000,000 AGD: 500,000 Benign: 500,000	Equal to the union of the dictionary-based dataset and the character-based dataset
DGArchive in UMUDGA	AGD: 146,648	Families contained in UMUDGA
DGArchive not in UMUDGA	AGD: 137,457	Families not contained in UMUDGA
Netlab 360	AGD: 20,950	Latest AGDs from Netlab 360 DGA

For the three datasets, the number of positive samples (AGD) and negative samples (benign) were roughly equal, so the accuracy, precision, recall and F-measure could be used to evaluate the model effects. Furthermore, the three generalization testing sets only contained AGDs, so only the recall rate was considered. The definitions of the indicators are as follows:

Accuracy (*A*): The ratio of the correctly classified samples to the total samples, which represents the confidence of the classification.

$$A = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

Precision (*P*): The ratio of the true positives (TPs) to the sum of the TPs and the false positives (FPs), which represents the confidence of the classification.

$$P = \frac{TP}{TP + FP} \tag{2}$$

Recall (*R*): The ratio of the TPs to the sum of the TPs and the false negatives (FNs), which represents the completeness of the classification.

$$R = \frac{TP}{TP + FN} \tag{3}$$

F-Measure (*F1*): The harmonic mean of the precision and recall, which represents the synthesis of the performance of the classification.

$$F1 = \frac{2 * P * R}{P + R} \tag{4}$$

where *TP* represents the number of samples correctly classified as positive, *FN* represents the number of samples incorrectly classified as negative, *FP* represents the number of samples incorrectly classified as positive, and *TN* represents the true negatives that are correctly classified as negative.

4.2. Experimental Settings

In this paper, three existing methods were selected for comparison: the Koh method [20] based on meaningful word segmentation, the ATT-CNN-BiLSTM method [34] based on local features extracted by CNN and LSTM.MI method [5] based on character LSTM. (1) Koh et al. (2018) [20] used Wordninja [46] for word segmentation and extracted context-sensitive word embedding features through ELMo [39]. The mean values of each dimension of word features were calculated for classification. This method is suitable for the detection of dictionary-based AGDs. (2) ATT-CNN-BiLSTM (2020) [34] used a CNN to extract local features and used bidirectional LSTM to extract sequence features for classification. This method is suitable for the detection of both dictionary-based and character-based AGDs. (3) LSTM.MI (2018) [5] used the LSTM model for unbalanced AGD classification. Since the datasets we constructed were roughly balanced, this method is basically the same as the method of Woodbridge (2016) [4], and this method is suitable for the detection of character-based AGDs.

Our experiments could be divided into three categories: the comparison experiments with existing detection methods, the ablation experiments, and the comparison experiment with detection engines.

(1) Comparison experiments with existing detection methods

We conducted 4 groups of compassion experiments, to compare the overall detection effects, the recall rate of certain DGA families, the detection ability on real-world samples, and the detection ability against adversarial DGA families between our method and the three comparison methods. The details of the experiments are as follows.

- **Comparison experiment for detection effects.** We compared our method with the Koh method, ATT-CNN-BiLSTM method, and LSTM. MI method on the three datasets.
- **Comparison experiment for family recall.** We compared the recall rate of each DGA family of our method with the Koh method, ATT-CNN-BiLSTM method, and LSTM.MI method. For each method, we used the final model trained on the full dataset and calculated the recall rate of each DGA family in the full dataset to show the detection ability of different methods for each DGA family.
- **Comparison experiment for generalization recall.** We compared the recall rate of each generalization dataset of our method with the Koh method, ATT-CNN-BiLSTM method, and LSTM.MI method. For each method, we used the final model trained on the full dataset to test the recall rate on the three generalization datasets constructed based on DGArchive and Netlab 360 to show the detection ability on real-world samples.
- **Comparison experiment for adversarial DGA families.** Based on adversarial DGA families named *deception* and *deception2* proposed by Spooren et al. [42], we reconstructed a new dataset to compare the detection effects on such kinds of DGA families with the Koh method, ATT-CNN-BiLSTM method, and LSTM.MI method.

(2) Ablation experiments

Our method fuses a variety of features. We conducted 3 groups of ablation experiments for the selection of word distribution features, the selection of n-gram sequence features, and feature fusion. The details of the experiments are as follows.

- **Effectiveness experiment for word distribution features.** We compared the detection effects of two-word segmentation methods, including Wordninja [46] and WordSegment [47], and the detection effects of two kinds of features, including Mean and Std, to select the best word distribution features.

- **Effectiveness experiment for n-gram sequence.** We compared the detection effects of 1-gram, 2-gram and 3-gram sequence features and the detection effects of different combinations of n-gram sequence features to select the best combination of n-grams.
- **Effectiveness experiment for feature fusion.** We added each type of feature in sequence and observed the changes in the detection effect to verify the effectiveness of each type of feature.

(3) Comparison experiment with detection engines

Based on Netlab 360 generalization set, we compared the recall rates of our method with the detection engines in VirusTotal [55].

4.3. Results

4.3.1. Comparison Experiments with Existing Detection Methods

(1) Comparison experiment for detection effects

The comparison methods we implemented were the Koh method [20] based on meaningful word segmentation, the ATT-CNN-BiLSTM method [34] based on local features extracted by CNN, and the LSTM.MI method [5] based on character LSTM. Our method was implemented based on the PyTorch and Allennlp packages. The initial learning rate was set to 0.001, and Adam was selected as the optimizer. During the training stage, a total of 25 epochs were iterated, and the learning rate decayed to 0.1 of the previous rate every 5 iterations. The source code for the LSTM.MI method was released based on Keras, and we directly used the original implementation for experiments. For the Koh method, we reimplemented the method in the paper. For the implementation of ELMo, we chose two methods: one was based on TensorFlow_hub [56] as described in the paper, and the other was based on Allennlp with PyTorch, which was the same as our method. We found that the effect of ELMo implemented based on PyTorch was better in experiments, so we chose it as the final implementation. For the ATT-CNN-BiLSTM method, we referred to the source code of LSTM.MI and reimplemented the method based on Keras according to the description in the paper. The performances of the four methods are shown in Table 3.

Table 3. Performance of comparison experiments.

Datasets	Indicators	Ours	Koh	ATT-CNN-BiLSTM	LSTM.MI
Dictionary-based dataset	Accuracy	96.33%	94.39%	94.13%	93.09%
	Precision	96.34%	94.34%	93.61%	92.88%
	Recall	96.64%	94.95%	95.25%	93.96%
	F1	96.49%	94.64%	94.43%	93.42%
Character-based dataset	Accuracy	98.64%	96.79%	98.13%	98.42%
	Precision	98.68%	96.72%	98.53%	98.57%
	Recall	98.56%	96.78%	97.68%	98.23%
	F1	98.62%	96.75%	98.10%	98.40%
Full dataset	Accuracy	97.24%	94.63%	96.14%	96.12%
	Precision	97.22%	94.78%	96.16%	96.72%
	Recall	97.25%	94.46%	96.10%	95.47%
	F1	97.23%	94.62%	96.13%	96.09%

In all tables of the experimental results, the best result is shown in red, and the second-best result is shown in blue. Based on the results of the comparison, we found that our method achieved the highest detection performance on all three datasets. The Koh method uses the mean values of the context-sensitive word embedding features of the words as the final features, and it achieved better detection performance for dictionary-based AGDs,

while the detection effect of character-based AGDs was relatively poor. The LSTM.MI method uses the traditional character sequence detection method, so it exhibited relatively good detection performance for character-based AGDs. The ATT-CNN-BiLSTM method achieved better results on the full dataset by fusing local features and sequence features.

By the effective fusion of the multiple features, our method significantly improved the performance of both the dictionary-based dataset and the full dataset. Since only 1-gram sequence features in our method were specifically for the detection of character-based AGDs and the gain of other features on character-based AGDs was relatively small, the performance of our method was only slightly better than that of the LSTM.MI method.

(2) Comparison experiment for family recall

We compared the recall rate of each DGA family of our method with that of the Koh method, ATT-CNN-BiLSTM method, and the LSTM.MI method. For each method, we used the final model trained on the full dataset and calculated the recall rate of each DGA family in the full dataset to show the detection ability of different methods for each DGA family. The recall rate of each method for each family is shown in Table 4. In Table 4, the first 11 families, are dictionary-based DGA families, and the rest are character-based DGA families.

Table 4. Comparison of recall of different families.

Family	Recall			
	Ours	**Koh**	**ATT-CNN-BiLSTM**	**LSTM.MI**
Gozi_gpl	94.55%	91.50%	87.70%	86.30%
Gozi_luther	93.00%	82.70%	88.40%	85.35%
Gozi_nasa	94.00%	87.60%	89.35%	87.30%
Gozi_rfc4343	94.90%	88.55%	88.50%	85.80%
Matsnu	88.75%	81.15%	84.95%	79.80%
Nymaim	67.20%	52.60%	58.30%	54.85%
Pizd	98.54%	97.28%	97.91%	95.40%
Rovnix	97.90%	97.70%	95.30%	92.70%
Suppobox_1	98.54%	98.06%	97.38%	95.45%
Suppobox_2	97.35%	95.35%	94.40%	94.85%
Suppobox_3	99.90%	97.65%	99.50%	99.50%
Alureon	98.20%	97.95%	97.65%	96.65%
Banjori	100.00%	100.00%	100.00%	100.00%
Bedep	99.55%	99.40%	99.45%	99.35%
Ccleaner	100.00%	99.80%	99.90%	100.00%
Chinad	100.00%	99.85%	100.00%	99.90%
Corebot	100.00%	100.00%	100.00%	99.95%
Cryptolocker	99.30%	98.70%	98.75%	98.80%
Dircrypt	98.55%	98.20%	98.25%	98.20%
Dyre	100.00%	100.00%	100.00%	100.00%
Fobber_v1	99.85%	99.90%	99.75%	99.75%
Fobber_v2	97.90%	97.15%	97.50%	96.85%
Kraken_v1	99.95%	99.65%	99.85%	99.80%
Kraken_v2	98.60%	97.60%	97.95%	97.20%

Table 4. *Cont.*

Family	Recall			
	Ours	**Koh**	**ATT-CNN-BiLSTM**	**LSTM.MI**
Locky	97.10%	95.65%	96.25%	95.90%
Murofet_v1	100.00%	100.00%	100.00%	100.00%
Murofet_v2	99.80%	99.70%	99.85%	99.65%
Murofet_v3	100.00%	100.00%	100.00%	100.00%
Necurs	98.55%	97.55%	98.05%	97.65%
Padcrypt	99.65%	98.70%	99.40%	99.55%
Proslikefan	91.65%	87.85%	89.60%	88.55%
Pushdo	95.25%	86.15%	95.15%	93.90%
Pykspa	92.20%	88.35%	90.70%	89.70%
Pykspa_noise	92.40%	89.20%	91.70%	90.60%
Qadars	99.40%	98.80%	99.10%	98.95%
Qakbot	99.25%	98.70%	99.10%	98.65%
Ramdo	99.65%	99.25%	99.80%	99.85%
Ramnit	98.60%	97.45%	98.15%	97.50%
Ranbyus_v1	99.55%	99.45%	99.55%	99.55%
Ranbyus_v2	99.80%	99.80%	99.75%	99.85%
Shiotob	99.05%	98.90%	98.65%	98.70%
Simda	94.80%	80.60%	96.15%	96.15%
Sisron	100.00%	100.00%	100.00%	100.00%
Symmi	100.00%	99.95%	100.00%	99.80%
Tempedreve	94.20%	90.75%	92.90%	92.75%
Tinba	99.80%	99.15%	99.20%	99.30%
Vawtrak_v1	96.10%	92.45%	94.40%	94.25%
Vawtrak_v2	99.65%	86.60%	98.15%	99.70%
Vawtrak_v3	99.80%	79.90%	99.00%	99.00%
Zeus-newgoz	100.00%	100.00%	100.00%	100.00%

From the comparison results, among the 50 families, our method achieved the highest recall rates in 44 families; the recall rates of some families were slightly lower than other methods, but the gap was relatively small. In addition, our method achieved relatively high recall rates for most families, but the recall rates of some families were slightly lower, such as the recall rates of two families of dictionary-based AGDs, Nymaim and Matsnu, which were only 67.20% and 88.75%, respectively, but were still higher than the three comparison methods. Overall, the recall rates of character-based AGDs were higher than those of dictionary-based AGDs. In addition, the average recall rate of 11 character-based DGA families was only 93.15%, demonstrating that after the character-based AGDs were added to the dataset, the recall rates of dictionary-based AGDs decreased slightly.

(3) Comparison experiment for generalization recall

In this experiment, we compared the detection performance of our final model and the 3 comparison methods on other datasets to measure the detection performance in real-world detection. We used the three generalization testing datasets constructed from

DGArchive and Netlab 360 to compare the recall rates of the final models trained on the full dataset. The experimental results are shown in Table 5.

Table 5. Comparison of generalization abilities.

Dataset	Total Samples	Recall			
		Ours	Koh	ATT-CNN-BiLSTM	LSTM.MI
DGArchive in UMUDGA	146,648	92.99%	90.98%	91.24%	89.78%
DGArchive not in UMUDGA	137,457	84.46%	82.11%	80.94%	79.35%
Netlab 360	20,950	92.27%	90.87%	89.50%	88.89%

Based on the comparison results, we found that our method achieved the best recall rates on the three generalization datasets, which were slightly higher than those of the other three comparison methods, demonstrating that our method has the ability to detect AGDs in real traffic and the ability to detect latest AGD samples.

(4) Comparison experiment for adversarial DGA families

In recent years, many researchers use adversarial methods to construct AGDs, which makes the detection even harder [40–45]. In order to test the detection ability of our method against such DGA families, we constructed a dataset for the experiment. We separately selected 10,000 AGDs generated by adversarial methods named *deception* and *deception2* proposed by Spooren et al. [42], and randomly selected 50,000 AGDs and 70,000 benign domain names from the full dataset. Finally, a dataset containing 140,000 domain names was constructed and split into training, validation, and testing sets. The experimental results are shown in Table 6.

Table 6. Comparison of adversarial DGA families.

Indicators	Ours	Koh	ATT-CNN-BiLSTM	LSTM.MI
Accuracy	89.13%	87.10%	84.33%	84.30%
Precision	89.32%	86.90%	86.21%	86.43%
Recall	88.87%	87.37%	81.73%	81.37%
F1	89.10%	87.13%	83.91%	83.83%
Recall of *Deception*	74.42%	72.63%	61.24%	59.31%
Recall of *Deception2*	72.24%	66.50%	50.73%	49.97%

The results in Table 6 show that our method achieved the best detection performance, and the recall rates of the two adversarial DGA families are more than 70%, which proves that our method also has detection ability for Adversarial DGA samples.

4.3.2. Ablation Experiments

(1) Effectiveness experiment for word distribution features

For the detection of dictionary-based AGDs, we first performed meaningful word segmentation and calculated the context-sensitive word embedding features. Then, we calculated the standard deviation (Std) of each dimension of word features to measure the correlation between words based on the Koh method. The mean of each dimension (Mean) was calculated to represent the distribution of word features. The word segmentation methods we considered were WordSegment and Wordninja. We assessed the performance of six settings: WordSegment (Mean), WordSegment (Std), WordSegment (Mean+Std), Wordninja (Mean), Wordninja (Std), and Wordninja (Mean+Std). For each setting of the experiments, two fully-connected layers were used for classification after the word distribution features were obtained. The experimental results are shown in Table 7.

Table 7. Performance of different word distribution features.

Datasets	Indicators	WordSegment (Mean)	WordSegment (Std)	WordSegment (Mean+Std)	Wordninja (Mean)	Wordninja (Std)	Wordninja (Mean+Std)
Dictionary -based dataset	Accuracy	93.09%	93.80%	94.73%	93.52%	94.90%	95.00%
	Precision	93.47%	94.63%	94.93%	93.56%	94.69%	94.90%
	Recall	93.27%	93.41%	94.97%	94.06%	95.60%	95.56%
	F1	93.37%	94.02%	94.95%	93.81%	95.14%	95.23%
Character -based dataset	Accuracy	95.16%	89.71%	95.99%	96.47%	97.22%	97.41%
	Precision	94.92%	84.19%	95.32%	96.53%	97.47%	97.57%
	Recall	95.29%	97.45%	96.62%	96.30%	96.89%	97.18%
	F1	95.10%	90.34%	95.97%	96.42%	97.18%	97.37%
Full dataset	Accuracy	93.19%	87.82%	94.13%	94.38%	95.20%	95.46%
	Precision	93.47%	82.96%	94.23%	94.95%	96.04%	95.48%
	Recall	92.86%	95.17%	94.01%	93.74%	94.28%	95.44%
	F1	93.16%	88.64%	94.12%	94.34%	95.15%	95.46%

Based on the results, the performance of Wordninja was significantly better than that of WordSegment. In addition, for both Wordninja and WordSegment, the fusion of the Mean and Std features effectively improved the detection effect of the model compared to using only Std or Mean as features. Therefore, Wordninja (Mean+Std) was selected as the final feature in this experiment. In addition, Wordninja (Mean) was similar to the Koh method, and the Wordninja (Mean) method had two more fully-connected layers than the Koh method. In contrast, the performance of Wordninja (Mean) was slightly worse than that of the Koh method. Based on our analysis, we believe that it affects the detection ability after increasing the complexity of the model when only the mean values are used as features. However, after fusing the mean and Std features, the detection performance was significantly better than that of the Koh method, which demonstrated the effectiveness of the fusion of the two kinds of features.

(2) Effectiveness experiment for n-gram sequence

Common n-gram sequence methods mainly include 1-gram, 2-gram and 3-gram methods. We used the three n-gram algorithms to obtain n-gram sequences and used the LSTM model for sequence feature extraction and classification. Here, we compared the detection effect of each single n-gram sequence and the detection effects of two feature fusion schemes using 1+3-grams and 1+2+3-grams to select the n-gram sequence features. The experimental results are shown in Table 8.

Table 8. Performance of different n-gram sequence features.

Datasets	Indicators	1-Gram	2-Gram	3-Gram	1, 3-Gram	1, 2, 3-Gram
Dictionary -based dataset	Accuracy	94.56%	94.72%	95.19%	95.24%	95.26%
	Precision	94.48%	94.30%	95.26%	95.33%	94.94%
	Recall	95.14%	95.67%	95.53%	95.56%	96.03%
	F1	94.81%	94.98%	95.40%	95.45%	95.48%
Character -based dataset	Accuracy	98.45%	98.37%	98.01%	98.50%	98.51%
	Precision	98.48%	98.43%	98.28%	98.50%	98.75%
	Recall	98.38%	98.27%	97.68%	98.47%	98.22%
	F1	98.43%	98.35%	97.98%	98.48%	98.48%
Full dataset	Accuracy	96.43%	96.39%	96.45%	96.62%	96.63%
	Precision	96.62%	97.08%	96.75%	96.64%	97.10%
	Recall	96.22%	95.65%	96.13%	96.60%	96.12%
	F1	96.42%	96.36%	96.44%	96.62%	96.61%

Based on the results, the 1-gram method performed better on the character-based dataset, while the 3-gram method performed better on the dictionary-based dataset. We believe that when the value of n in the n-gram increases, the detection performance of dictionary-based AGDs increases, while the performance of character-based AGDs decreases. Therefore, we achieved good detection performance for the two kinds of AGDs at the same time by fusing multiple kinds of n-grams.

Comparing the results of 1+3-gram and 1+2+3-gram, we found that 1+3-gram effectively detected both kinds of AGDs. After adding 2 grams, the accuracy and F1 value were only slightly improved, which did not significantly improve the overall performance of the model but greatly increased the complexity of the model. Therefore, in the final model, we gave priority to fusing 3-grams and 1-grams and finally considered 2-grams.

(3) Effectiveness experiment for feature fusion

In this experiment, we compared the detection performance of different feature fusion methods. The features we chose included word distribution features, statistical features, and n-gram features. Based on the results in experiments (1) and (2), we first used Wordninja (Std+Mean), which was the best word distribution feature, as the basic method, and then we sequentially added statistical features, 3-gram sequence features, 1-gram sequence features, and 2-gram sequence features. We compared the performance before and after each feature was added, and then the optimal feature fusion method was selected. The experimental results are shown in Table 9.

The results indicated that after adding statistical features to the word distribution features based on Wordninja, the indicators on the three datasets were improved, demonstrating that the statistical features can effectively improve the detection effect. Then, after adding 3-gram sequence features, the detection ability for the two kinds of AGDs improved, demonstrating the effectiveness of 3-gram sequence features. After that, we continued adding 1-gram sequence features, and the detection performance on the three datasets also significantly improved, so we also retained the 1-gram sequence features. Finally, after adding the 2-gram sequence features on this basis, the overall detection performance decreased due to the increase in the complexity of the model. The introduction of 2-grams led to a decrease in the effect of the model, so we did not consider the 2-gram sequence features in the final model. That is, Wordninja (Std + Mean) + 1,3-gram + statistical features were finally selected as our detection method.

Table 9. Performance of different feature fusion methods.

Datasets	Indicators	Wordninja	Wordninja + Statistics	Wordninja + 3-Gram + Statistics	Wordninja + 1, 3-Gram + Statistics	Wordninja + 1, 2, 3-Gram + Statistics
Dictionary-based dataset	Accuracy	95.00%	95.17%	96.17%	96.33%	96.19%
	Precision	94.90%	95.20%	96.12%	96.34%	96.32%
	Recall	95.56%	95.56%	96.55%	96.64%	96.38%
	F1	95.23%	95.38%	96.34%	96.49%	96.35%
Character-based dataset	Accuracy	97.41%	97.55%	98.36%	98.64%	98.57%
	Precision	97.57%	97.60%	98.27%	98.68%	98.74%
	Recall	97.18%	97.44%	98.42%	98.56%	98.36%
	F1	97.37%	97.52%	98.34%	98.62%	98.55%
Full dataset	Accuracy	95.46%	95.43%	96.89%	97.24%	97.10%
	Precision	95.48%	96.10%	96.85%	97.22%	97.45%
	Recall	95.44%	94.68%	96.92%	97.25%	96.72%
	F1	95.46%	95.39%	96.89%	97.23%	97.08%

4.3.3. Comparison Experiment with Detection Engines

VirusTotal [55] can determine whether a domain name is an AGD or not; in addition, it also integrates detection results of many independent detection engines which determine whether a domain name is malicious. We obtained the recall rates of each detection engines in VirusTotal and our method on the Netlab 360 generalization testing sets. The detection effects are shown in Table 10. For the detection engines in VirusTotal, the recall rate represents the proportion of domain names predicted as malicious; and for VirusTotal itself, the recall rate represents the proportion of domain names predicted as "DGA". We only present engines with the recall rates over 30%.

Table 10. Recall rates of different open access detection engines.

Indicator	Ours	VirusTotal	Heimdal Security	Antiy-AVL	Seclookup
Recall	92.27%	85.75%	83.91%	43.40%	31.23%

The results in Table 10 show that, our method achieved higher recall rate than the detection engines in VirusTotal.

5. Conclusions

In this paper, we proposed an AGD detection method based on the fusion of meaningful word segmentation and n-gram sequence features. We used meaningful word segmentation to describe the composition of dictionary-based AGDs and used the mean and standard deviation to describe the word distribution of the domain name. We constructed additional 11-dimensional statistical features based on the word segmentation results. In addition, we used 3-gram and 1-gram sequences to characterize the sequence features of dictionary-based AGDs and character-based AGDs, respectively. Finally, we fused these four kinds of features to realize an end-to-end detection of AGDs, which significantly improved the detection effect of dictionary-based AGDs and accounted for the detection performance of character-based AGDs. We conducted experiments on three datasets, which separately contained only dictionary-based AGDs, only character-based AGDs, and both character-based and dictionary-based AGDs. The testing results demonstrated that our method achieved the highest accuracy on each of the three datasets and reached an accuracy of 97.25% on the full dataset, which was higher than that of the state-of-the-art methods.

The recall rate of some dictionary-based DGA families was relatively low on the full dataset. We will further analyze these families to improve the detection performance, and then we will deploy them in the real-world network to detect the AGDs.

Author Contributions: Conceptualization, S.C. and B.L.; methodology, S.C. and C.X.; validation, S.C., Y.C. and C.X.; writing—original draft preparation, S.C.; writing—review and editing, B.L. and Y.C.; supervision, B.L.; funding acquisition, B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by State Key Laboratory of Software Development Environment grant number SKLSDE-2020ZX-02.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

DGA	Domain generation algorithm
AGD	Algorithmically generated domains
DNS	Domain name system
IP	Internet protocol
C&C	command and control
TLD	Top-level domain name
NXDomain	Non-existent domains
SVM	Support vector machines
RF	Random forests
RNN	Recurrent neural network
CNN	Convolutional neural network
LSTM	long short-term memory
Std	standard deviation
ccTLD	Country code TLD

References

1. Schüppen, S.; Teubert, D.; Herrmann, P.; Meyer, U. FANCI: Feature-based Automated NXDomain Classification and Intelligence. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 1165–1181.
2. Sivaguru, R.; Choudhary, C.; Yu, B.; Tymchenko, V.; Nascimento, A.; De Cock, M. An evaluation of DGA classifiers. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 5058–5067.
3. Mac, H.; Tran, D.; Tong, V.; Nguyen, L.G.; Tran, H.A. DGA botnet detection using supervised learning methods. In Proceedings of the Eighth International Symposium on Information and Communication Technology, Nha Trang, Vietnam, 7–8 December 2017; pp. 211–218.
4. Woodbridge, J.; Anderson, H.S.; Ahuja, A.; Grant, D. Predicting domain generation algorithms with long short-term memory networks. *arXiv* **2016**, arXiv:1611.00791.
5. Tran, D.; Mac, H.; Tong, V.; Tran, H.A.; Nguyen, L.G. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing* **2018**, *275*, 2401–2413. [CrossRef]
6. Yu, B.; Pan, J.; Hu, J.; Nascimento, A.; De Cock, M. Character level based detection of DGA domain names. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
7. Tong, V.; Nguyen, G. A method for detecting DGA botnet based on semantic and cluster analysis. In Proceedings of the Seventh Symposium on Information and Communication Technology, Ho Chi Minh, Vietnam, 8–9 December 2016; pp. 272–277.
8. Schiavoni, S.; Maggi, F.; Cavallaro, L.; Zanero, S. Phoenix: DGA-based botnet tracking and intelligence. In Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Egham, UK, 10–11 July 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 192–211.
9. Satoh, A.; Fukuda, Y.; Kitagata, G.; Nakamura, Y. A Word-Level Analytical Approach for Identifying Malicious Domain Names Caused by Dictionary-Based DGA Malware. *Electronics* **2021**, *10*, 1039. [CrossRef]
10. Casino, F.; Lykousas, N.; Homoliak, I.; Patsakis, C.; Hernandez-Castro, J. Intercepting hail hydra: Real-time detection of algorithmically generated domains. *J. Netw. Comput. Appl.* **2021**, *190*, 103135. [CrossRef]
11. Catania, C.; García, S.; Torres, P. Deep convolutional neural networks for DGA detection. In Proceedings of the Argentine Congress of Computer Science, Tandil, Argentina, 8–12 October 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 327–340.
12. 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. (TISSEC)* **2014**, *16*, 1–28. [CrossRef]
13. Fang, X.; Sun, X.; Yang, J.; Liu, X. Domain-embeddings based DGA detection with incremental training method. In Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 7–10 July 2020; pp. 1–6.
14. Wang, T.S.; Lin, C.S.; Lin, H.T. DGA botnet detection utilizing social network analysis. In Proceedings of the 2016 International Symposium on Computer, Consumer and Control (IS3C), Xi'an, China, 4–6 July 2016; pp. 333–336.
15. Abbink, J.; Doerr, C. Popularity-based detection of domain generation algorithms. In Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 29 August 29–1 September 2017; pp. 1–8.
16. Menon, A. Thwarting C2 Communication of DGA-Based Malware using Process-level DNS Traffic Tracking. In Proceedings of the 2019 7th International Symposium on Digital Forensics and Security (ISDFS), Barcelos, Portugal, 10–12 June 2019; pp. 1–5.
17. Pereira, M.; Coleman, S.; Yu, B.; DeCock, M.; Nascimento, A. Dictionary extraction and detection of algorithmically generated domain names in passive DNS traffic. In Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses, Heraklion, Greece, 10–12 September 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 295–314.
18. Shen, Q.; Zou, F. Detecting Dictionary Based AGDs Based on Community Detection. In Proceedings of the International Conference on Security and Privacy in Communication Systems, Washington, DC, USA, 21–23 October 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 42–48.
19. Zheng, C.; Qiang, Q.; Zang, T.; Chao, W.; Zhou, Y. Themis: A Novel Detection Approach for Detecting Mixed Algorithmically Generated Domains. In Proceedings of the 2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), Shenzhen, China, 11–13 December 2019; pp. 259–264.
20. Koh, J.J.; Rhodes, B. Inline detection of domain generation algorithms with context-sensitive word embeddings. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 2966–2971.
21. Zhou, S.; Lin, L.; Yuan, J.; Wang, F.; Ling, Z.; Cui, J. CNN-based DGA detection with high coverage. In Proceedings of the 2019 IEEE International Conference on Intelligence and Security Informatics (ISI), Shenzhen, China, 1–3 July 2019; pp. 62–67.
22. Lin, S.; Zhong, S.; Cheng, K. A Method with Pre-trained Word Vectors for Detecting Wordlist-based Malicious Domain Names. In Proceedings of the Journal of Physics: Conference Series, Changsha, China, 24–25 October 2020; IOP Publishing: Bristol, UK, 2021; Volume 1757, p. 012171.
23. Yang, L.; Liu, G.; Liu, W.; Bai, H.; Zhai, J.; Dai, Y. Detecting Multielement Algorithmically Generated Domain Names Based on Adaptive Embedding Model. *Secur. Commun. Netw.* **2021**, *2021*, 5567635. [CrossRef]
24. Patsakis, C.; Casino, F. Exploiting statistical and structural features for the detection of Domain Generation Algorithms. *J. Inf. Secur. Appl.* **2021**, *58*, 102725. [CrossRef]
25. Yang, L.; Liu, G.; Zhai, J.; Dai, Y.; Yan, Z.; Zou, Y.; Huang, W. A novel detection method for word-based DGA. In Proceedings of the International Conference on Cloud Computing and Security, Haikou, China, 8–10 June 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 472–483.

26. Xu, C.; Shen, J.; Du, X. Detection method of domain names generated by DGAs based on semantic representation and deep neural network. *Comput. Secur.* **2019**, *85*, 77–88. [CrossRef]
27. Morbidoni, C.; Spalazzi, L.; Teti, A.; Cucchiarelli, A. Leveraging n-gram neural embeddings to improve deep learning DGA detection. In Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, Virtual, 25–29 April 2022; pp. 995–1004.
28. Selvi, J.; Rodríguez, R.J.; Soria-Olivas, E. Detection of algorithmically generated malicious domain names using masked N-grams. *Expert Syst. Appl.* **2019**, *124*, 156–163. [CrossRef]
29. Cucchiarelli, A.; Morbidoni, C.; Spalazzi, L.; Baldi, M. Algorithmically generated malicious domain names detection based on n-grams features. *Expert Syst. Appl.* **2021**, *170*, 114551. [CrossRef]
30. Alaeiyan, M.; Parsa, S.; Vinod, P.; Conti, M. Detection of algorithmically-generated domains: An adversarial machine learning approach. *Comput. Commun.* **2020**, *160*, 661–673. [CrossRef]
31. Vranken, H.; Alizadeh, H. Detection of DGA-Generated Domain Names with TF-IDF. *Electronics* **2022**, *11*, 414. [CrossRef]
32. Liang, J.; Chen, S.; Wei, Z.; Zhao, S.; Zhao, W. HAGDetector: Heterogeneous DGA Domain Name Detection Model. *Comput. Secur.* **2022**, *120*, 102803. [CrossRef]
33. Yang, L.; Liu, G.; Dai, Y.; Wang, J.; Zhai, J. Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework. *IEEE Access* **2020**, *8*, 82876–82889. [CrossRef]
34. Ren, F.; Jiang, Z.; Wang, X.; Liu, J. A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network. *Cybersecurity* **2020**, *3*, 1–13. [CrossRef]
35. Ren, F.; Jiang, Z.; Liu, J. Integrating an Attention Mechanism and Deep Neural Network for Detection of DGA Domain Names. In Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), Portland, OR, USA, 4–6 November 2019; pp. 848–855.
36. Highnam, K.; Puzio, D.; Luo, S.; Jennings, N.R. Real-time detection of dictionary dga network traffic using deep learning. *SN Comput. Sci.* **2021**, *2*, 1–17. [CrossRef]
37. Wang, Z. Detecting Algorithmically Generated Domains Using a GCNN-LSTM Hybrid Neural Network. *arXiv* **2022**, arXiv:2208.03445.
38. Curtin, R.R.; Gardner, A.B.; Grzonkowski, S.; Kleymenov, A.; Mosquera, A. Detecting DGA domains with recurrent neural networks and side information. In Proceedings of the 14th International Conference on Availability, Reliability and Security, Canterbury, UK, 26–29 August 2019; pp. 1–10.
39. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*; Association for Computational Linguistics: New Orleans, LA, USA, 2018; pp. 2227–2237. [CrossRef]
40. Anderson, H.S.; Woodbridge, J.; Filar, B. DeepDGA: Adversarially-tuned domain generation and detection. In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, Vienna, Austria, 28 October 2016; pp. 13–21.
41. Peck, J.; Nie, C.; Sivaguru, R.; Grumer, C.; Olumofin, F.; Yu, B.; Nascimento, A.; De Cock, M. CharBot: A simple and effective method for evading DGA classifiers. *IEEE Access* **2019**, *7*, 91759–91771. [CrossRef]
42. Spooren, J.; Preuveneers, D.; Desmet, L.; Janssen, P.; Joosen, W. Detection of algorithmically generated domain names used by botnets: A dual arms race. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 8–12 April 2019; pp. 1916–1923.
43. Yun, X.; Huang, J.; Wang, Y.; Zang, T.; Zhou, Y.; Zhang, Y. Khaos: An adversarial neural network DGA with high anti-detection ability. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 2225–2240. [CrossRef]
44. Zheng, Y.; Yang, C.; Yang, Y.; Ren, Q.; Li, Y.; Ma, J. ShadowDGA: Toward Evading DGA Detectors with GANs. In Proceedings of the 2021 International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 19–22 July 2021; pp. 1–8.
45. Liu, W.; Zhang, Z.; Huang, C.; Fang, Y. CLETer: A Character-level Evasion Technique against Deep Learning DGA Classifiers. *EAI Endorsed Trans. Secur. Saf.* **2021**, *7*, e5. [CrossRef]
46. Anderson, D. Word Ninja. Available online: https://github.com/keredson/wordninja (accessed on 10 June 2022).
47. Jenks, G. Python Word Segmentation. Available online: https://github.com/grantjenks/python-wordsegment (accessed on 10 June 2022).
48. Wikipedia. Top-Level_Domain. Available online: https://en.wikipedia.org/wiki/Top-level_domain (accessed on 1 June 2021).
49. Gavin, M. Second-Level-Domains. Available online: https://github.com/gavingmiller/second-level-domains/blob/master/SLDs.csv (accessed on 1 June 2021).
50. Mockapetris, P.V. RFC1034: Domain Names-Concepts and Facilities. 1987. Available online: https://dl.acm.org/doi/pdf/10.17487/RFC1034 (accessed on 20 September 2019)
51. Alexa Web Information Company. Topsites. Available online: https://www.alexa.com/topsites (accessed on 10 January 2022).
52. Zago, M.; Pérez, M.G.; Pérez, G.M. UMUDGA: A dataset for profiling DGA-based botnet. *Comput. Secur.* **2020**, *92*, 101719. [CrossRef]
53. Plohmann, D. DGArchive. Available online: https://dgarchive.caad.fkie.fraunhofer.de/ (accessed on 10 June 2022).
54. Network Security Research Lab at 360. Netlab DGA Project. Available online: https://data.netlab.360.com/dga/ (accessed on 11 March 2022).

55. Virustotal. Virustotal-Free Online Virus, Malware and Url Scanner. Available online: https://www.virustotal.com (accessed on 11 March 2023).
56. Google. TensorFlow Hub: ELMo. Available online: https://tfhub.dev/google/elmo/2 (accessed on 10 June 2022).