Journal of Theoretical and Applied Information Technology

30th September 2023. Vol.101. No 18

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ALGORITHMICALLY GENERATED MALICIOUS DOMAIN DETECTION USING N-GRAMS EMBEDDING AND ATTENTION-BASED BIDIRECTIONAL GATED RECURRENT UNIT

ALFONSUS SUCAHYO HARIAJI[1] , ABBA SUGANDA GIRSANG[2]

[1]Computer Science Department, BINUS Graduate Program - Master of Computer Science,

Bina Nusantara University, Jakarta, Indonesia 11480

[2]Computer Science Department, BINUS Graduate Program - Master of Computer Science,

Bina Nusantara University, Jakarta, Indonesia 11480

E-mail: [1]alfonsus.hariaji@binus.ac.id, [2]agirsang@binus.edu

ABSTRACT

Botnets are one of the recent main cyber security threats. In order to avoid detection, botnets use Domain Generation Algorithm (DGA) to generate malicious domain names and maintain communication between infected bots and command and control server (C&C). Botnet malwares use various algorithm to generate domain names such as arithmetic, hashing, and wordlist/dictionary techniques. Recent traditional machine learning and deep learnin based detection methods need handcrafted domain name features which require more effort and advanced expertise and knowledge. This study aims to detect and classify DGA malicious domain without manually define and handcraft domain name features by only using the domain name. N-grams method was used to create sequences of domain names and then vectorize the sequences using word embedding technique to create n-grams embedding model. After vectorization, Bidirectional Gated Recurrent Unit (BiGRU) was used for domain name classification and attention mechanism was used to improve classification performance by applying attention weight. The experiment results demonstrate the N-Grams Embedding and Attention-based BiGRU model proposed in this paper can detect and classify various type of DGA domains generated by arithmetic, hashing, and wordlist algorithm more effective compared to older algorithm such as CNN and LSTM for both DGA malicious domain detection and classification task. The use of attention mechanism can also improve the accuracy and performance of the DGA malicious domain detection model compared to models that do not use attention mechanism.

Keywords: *Attention Mechanism, Domain Generating Algorithm, Gated Recurrent Unit, Malicious Domain, N-grams Embedding,*

1. INTRODUCTION

Currently, the internet has become a vital part of both personal and professional life for individuals. With the help of the internet, anyone can communicate with others with the intention and purpose of seeking information, and accessing various services such as email, e-banking, e-learning, social media, etc. Internet access media is also increasingly diverse. Not only computers, but everyone can access the internet through laptops, smartphones, and other devices that can connect to the internet.

Since the development of software development techniques, programs containing malicious code have emerged, but initially only had an impact on local devices. With the development of the internet, these programs can easily spread by taking advantage of the internet network. A hacker can spread malware through the internet to gain control and access to the infected computer. The infected computer can spread the same malware to other computers without the user's knowledge, forming a network of infected computers called a "Robot Network" / Botnet [1].

This network which contains bots (devices that have infected with botnet) is controlled by

attacker/hacker which called botmaster to control and take advantages of the network to do malicious act such as exploit, fraud, data stealing, spam spreading, etc. The characteristic of this botnet network is the existence of a dedicated server prepared by the botmaster as a Command and Control (C&C/C2) to distribute commands from the botmaster to the bots on the botnet network[2].

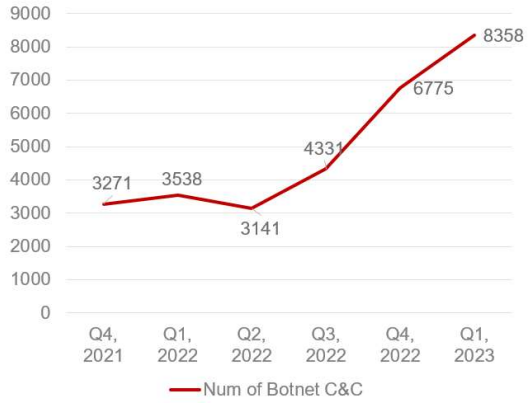

Figure 1: Number Of Botnet C&C

As shown on Figure 1, based on the Q1 2023 Botnet Threat Update report released by Spamhaus Project[3], a total of 8,358 Botnet C&Cs (Command and Control) were discovered in Q1 2023, compared to 6,775 in Q4 2022. This indicates a significant increase of 25% in the number of botnet servers that have been created. On a quarterly scale, the number of Botnets C&Cs found increased from 4,331 in Q3 2022 to 6,775 in Q4 2022, and to 8,358 in Q1 2023. With this significant increase in botnet C&C servers, researches is needed to detect domain names generated by DGAs to reduce the risk of botnet infection on computer devices.

Initially, remote access from the botmaster to the botnet network depended on pre-established IP addresses or domains. This method is easy for security applications to block or disrupt communication between the bots and the C&C server[4]. To overcome such blockades, hackers developed the Domain Generating Algorithm (DGA) method, which periodically generates pseudo domain names that can be used to maintain connectivity between the botnet, botmaster, and C&C server. By using the DGA method, botnets, and C&C server locations become difficult to detect and eliminate.

Research on DGA domain detection has been conducted, mainly using traditional machine learning techniques such as Random Forest[5], Decision Tree, and Naïve Bayes[6], as well as feature-based approaches such as the FANCI method[7]. However, traditional machine learning models like these require manual handcrafted features which need advanced expertise and knowledge[8]. If the method or algorithm used by the malware botnet changes, the string composition of the domain name will also change, so models that use handcrafted features will no longer have high accuracy.

To overcome the problem of the need for manual features engineering, research related to DGA malicious domain detection was conducted using deep learning methods such as using CNN [9] and LSTM [10], but these existing research only used one character in the domain name as a sequence, it was not yet known whether using two or more characters can improve accuracy of the model or not. In addition, newer deep learning algorithms such as Gated Recurrent Unit (GRU) had not been widely implemented in the case of DGA malicious domain detection. In this paper, N-grams embeddings was used to extract sequences from domain names not only one but two or more characters and Bidirectional Gated Recurrent Unit (BiGRU) with attention mechanism was proposed to detect and classify malicious domain which generated by DGA. Using n-grams embedding and BiGRU with attention mechanism, DGA malicious domains can be detected with no need to specify and handcraft domain name features manually.

2. RELATED WORK

This section summarizes earlier research into DGA-based domain detection. It also examines various methods for detect and classify non-malicious and malicious domain which generated by DGA. Such a review was carried out to improve the theoretical foundation of this research. Also, this section offers a discussion of the used techniques for DGA detection.

2.1 Domain Generating Algorithm

Domain names make it possible for browsers, applications, and servers to use internet resources by identifying domain names with IP addresses. Internet users experience the benefits of using a domain by simply entering the domain name into a browser or other online application without having to remember a complicated IP address.

The botmasters or attackers use Domain Generating Algorithm (DGA) techniques with

certain algorithms which are inserted in the malware spread on the botnet to dynamically generate domains that can avoid detection by the botnet detector system and choose one of these domains to be used as the address of the C&C server used.

Based on the DGA domain name that has been detected, the types of algorithms commonly used to generate domains [11] are as follows :

(1) Arithmetic-based DGA

Domain names are generated by calculating values with defined formula then converted into letters, numbers, or characters based on ASCII or random encoding. This algorithm is the most widely used for DGA. Example : "fgavropgu.com" (generated by conficker DGA)

(2) Hash-based DGA

Domain-generating algorithm that use hashing methods to generate domains. The hashing techniques used in this type of DGA are such as MD5 and SHA256. Example : "47faeb4f1b75a48499ba14e9b1cd895a.org" (generated by bamital DGA)

(3) Wordlist/Dictionary-based DGA

Domain names that are resulted from combining two or more words taken from English or other language word lists or dictionary to form a domain name. DGA malicious domains generated from this algorithm are difficult to detect because at first glance they resemble ordinary, legitimate domains that do not point to a C&C server. Example : "catpeakfearinterview.com" (generated by matsnu DGA)

Previous works related to DGA domain detection have been conducted using various methods and approaches. One of them used the DNS rule-based method[12]. In this study, researchers used DNS traffic to detect botnet. DNS queries extracted from DNS traffic and then compared with a predetermined database that consists of several legitimate domains. The domain shall be considered legitimate if there is a match. If not, it will be regarded as suspicious domain.

Traditional machine learning such as Random Forest is used in DGA detection research [5]. This study used Random Forest to detect DGA domains. The dataset used in this study were from Alexa for legitimate domain names as much as 100,000 domains and from NetLab360 as much as 153,200 for DGA domain. The Random Forest model used in this study obtained an accuracy of 83,82% but failed to detect "banjori", "matsu", "bigviktor" DGA families.

Big data technology is used in study related to DGA malicious domain detection [6]. Technology which used in this study is Apache Spark. The main purpose of using Spark is to effectively manage massive amounts of data [13]. This study used two types of datasets, one of which consists of public data from Alexa and OpenDNS for normal domain names and OSINT Feed for malicious domain names. The other dataset comes from realtime DNS traffic captured using big data. For detection and classification, this study used traditional machine learning and deep learning model. Random Forest, Decision Tree, and Naive Bayes for traditional machine learning and RNN and LSTM for deep learning. The LSTM method in this study was able to obtain an F1-score of 0.932, the highest score compared to RNN (0.909), Random Forest (0.828), Decision Tree (0.648), and Naive Bayes (0.693). LSTM was also used to detect DGA domain.

Study [14] used RNN and WHOIS Lookup as side information to domain name. Main purpose of this study is to detect DGA domains which have similarities with english words. This study used Alexa and OpenDNS dataset as normal domain names and DGArchive, Andrey Abakumov's DGA, and J. Bader's DGA as DGA domains.

Deep learning model such as CNN and LSTM was used to detect dictionary DGA [9]. Dictionary DGA is harder to detect because they use common english words and have similarities with normal domains. The dataset used in this study comes from Alexa for normal domain names and DGArchive for DGA domains. This study used model named Bilbo which contained CNN and LSTM but only obtain an F1-score of 0.5660.

2.2 N-Grams Embedding

N-grams are sequences of n units, these sequences can be characters, words, or parts of words consisting of n units. N-grams are one of the most widely used processes in text mining and natural language processing. N-gram counting is usually done by moving one word forward. For example, in the sentence "This is a sentence", if the n-gram extraction is done with n=1, it would become the following sequences: ["this", "is", "a", "sentence"], with n=2 it would become the

following sequences: [this is, is a, a sentence], and with n=3 it would become the following sequences : [this is a, is a sentence]. The number n=1 is often refered to as "unigram", n=2 as "bigram", n=3 as "trigram". There is no limit to the number of n used for n-gram extraction. In addition 1, 2, or 3, 4, 5, etc. can also be used.

In the field of text classification, text classification tasks can not only be performed at the word level, but also at the character level . A domain name as short text consists of a set of characters. Therefore, domain names can be processed into n-gram sequences [12]. For example, the domain name "rgjvjced.com" (generated by Torpig DGA) would be transformed into the following sequences of 2-grams: [rg,gj,jv,vj,jc,ce,ed,d.,.c,co,om] and sequences of 3-grams: [rgj, gjv, jvj, vjc, jce, ced, ed., d.c, .co, com].

Deep learning models cannot accept data in the form of strings or text. Therefore, the input data must be vectorized. The extraction results in the form of n-gram sequences from the domain can be vectorized into word vectors using word embedding[15]. Word embedding methods have been developed to assist in word embedding tasks with libraries such as Word2Vec [15], FastText [16], or GloVe[17].

2.3 Gated Recurrent Unit

Gated Recurrent Unit (GRU) was introduced to overcome the vanishing gradient descent problem in Recurrent Neural Network (RNN)[18]. Vanishing gradient descent is a condition where the gradient update from the initial time sequence to the final time sequence is getting smaller, so the RNN architecture or model does not perform well [19]. GRU is a variation of the RNN architecture. As in LSTM, GRU also uses a gates system but unlike LSTM which uses 3 (three) gates units (forget gate, input gate, output gate), GRU uses 2 (two) gates, namely reset gate and update gate z_t so that the computational process in GRU is simpler than LSTM [20] as shown on Figure 2.

For learning the current input vector X_t, a GRU unit update its hidden state by calculating reset gate r_t by summing the input vector X_t and hidden state from the previous time step $h_{(t-1)}$with bias b_r followed by sigmoid activation (σ) funcion. When r_t close to 0, the reset gate makes the unit act as if it is reading the first character of an input sequence, allowing it to *forget* the previously computed state [20]. In the form of equation, reset gate is computed using Eq. (1) where W_r and U_r are

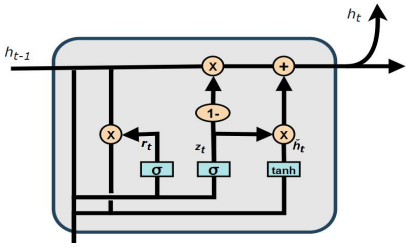

Figure 2: GRU Architecture

weight matrices for input vector X_t and hidden state $h_{(t-1)}$.

$$r_t = \sigma(W_r \,.\, X_t + U_r \,.\, h_{(t-1)} + b_r) \quad (1)$$

Similar to reset gate r_t , update gate z_t decides how much a GRU unit updates its content. Update gate is computed using Eq. (2) where W_z and U_z are weight matrices for input vector X_t and hidden state $h_{(t-1)}$.

$$z_t = \sigma(W_z . X_t + U_z . h_{(t-1)} + b_z) \quad (2)$$

The candidate hidden state $\tilde{h}_t$ is computed using Eq. (3) where W_h and U_h are weight matrices for input vector X_t and hidden state $h_{(t-1)}$.

$$\tilde{h}_t = tanh(W_h . X_t + r_t \odot U_h . h_{(t-1)} + b_z) \quad (3)$$

r_t in Eq. (3) is reset gate and $\odot$ is an element-wise multiplication.

After computing reset gate r_t, update gate z_t, and candidate hidden state $\tilde{h}_t$, a GRU unit computes its hidden state h_t for time step t by using Eq. (4).

$$h_t = (1 - z_t) \odot h_{(t-1)} + z_t \odot \tilde{h}_t \quad (4)$$

The hidden state calculation in GRU unit is carried forward to the next time step until the entire input sequence is calculated.

2.4 Attention Mechanism

The attention mechanism simulates the human brain's attention characteristics, which can be understood as always paying attention to more important information. In the field of NLP, the attention mechanism is introduced in the neural translation model using the encoder-decoder approach [21]. A novel types of attention-based model for machine translation is also proposed [22].

One of the most popular task in NLP, sentiment analysis, is also get advantages from attention mechanism[23]. Since then, more research has conducted to implement attention mechanism to many tasks such as text classification, abstract extraction, and text summarization. The most popular research on attention mechanism is the implementation of the transformer model[24] that started popular language models development such as GPT. The effective use of attention mechanism in natural language ignites interest in DGA domain name detection.

3. PROPOSED METHOD

3.1 System Architecture

An overview of system architecture proposed in this study to detect DGA-based malicious domain names using n-gram embedding and attention-based BiGRU is shown in Figure 3.

The model architecture in this paper consists of four components: N-grams embedding, BiGRU layer, attention layer, and output layer. Before entering the output layer, the domain name sequence is trained and uses dropout to prevent overfitting[25].

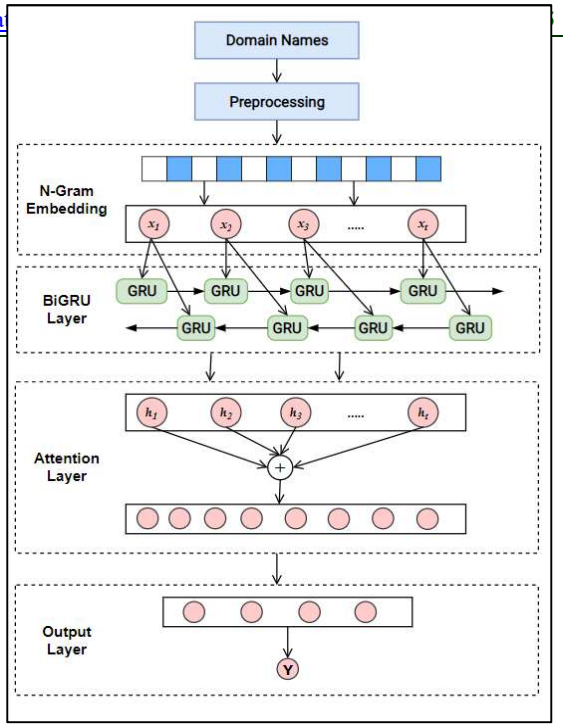

Figure 3: Proposed System Architecture

3.2 Preprocessing

Domain names are preprocessed before embedding. The preprocessing process in this paper includes :

(1) Case folding

The characters in domain names are case insensitive which means there is no difference between lowercase and uppercase letters. The characters in the domain name was folded to lowercase.

(2) Subdomain extraction

Subdomain parts of the domain name were removed by using the TLDExtract library in Python. For domain names that use free dynamic dns services such as "ghqdorqluleja21.ddns.net" (generated by DGA Corebot) or "pmfgfctidcffeago.mynumber.org" (generated by DGA Sutra), the subdomain is not removed because the generated part is already in the subdomain part of the domain name.

3.3 N-Grams Embedding

Pretrained n-grams embedding was developed using legitimate and DGA domain names from datasets. For more detail, pretrained n-grams embedding model build process is shown in Figure 4.

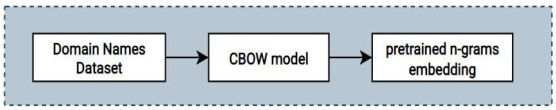

Figure 4: Pretrained N-Grams Embedding

In this research, domain names can be considered as sentences consisting of words. The term "words" in this case are streams of character level n-grams. FastText Continuous-Bag-of-Words (CBOW) is was implemented as word embedding technique which has good performance for text classification [16][26]. This study used 1-gram, 2-grams, 3-grams, and 4-grams to determine the best number of n for DGA domain detection. The length of domain names input sequences was set to 70 and embedding size to 100. The output of n-grams embedding will be input to the BiGRU layer.

3.4 BiGRU Layer

BiGRU is used to capture temporal features from every n-grams sequences in domain names. On the bidirectional architecture, there are two hidden layers from two separate GRUs. The two GRUs capture the dependencies in different direction. The first hidden layer of GRU captures dependencies of input vector sequences from x_1 to x_t, and the next layer captures dependencies of input vector sequences from x_t to x_1. The hidden layer state of BiGRU at time t is computed by weighted summation of forward hidden layer state h_{t-1} and reverse hidden layer state as shown as Eq. Eq. (7)

$$\overrightarrow{h_t} = GRU\left(x_t, \overrightarrow{h_t}\right) \quad (5)$$

$$\overleftarrow{h_t} = GRU\left(x_t, \overleftarrow{h_t}\right) \quad (6)$$

$$h_t = w_t \overrightarrow{h_t} + v_t \overleftarrow{h_t} + b_t \quad (7)$$

where w_t and v_t respectively represent the weights corresponding to the forward hidden state $\overrightarrow{h_t}$ and the reverse hidden state $\overleftarrow{h_t}$ of the BiGRU at time t, and b_t represents the bias of the hidden layer state. The GRU function in Eq. (5) and Eq. (6) to both forward and backward hidden state refers to the computation explained on the Section 2.3.

3.5 Attention Layer

The purpose of attention layer is to neglect unimportant information from sequences of domain names, selectively screening out of a small part of important information and focusing on it. For DGA domain detection, focusing on some certain parts of sequences will be effective to filter out the DGA irrelevant noise. Each domain names consist of several sequences which have the same weight and carry noise or useless information.

Attention layer is applied after BiGRU layer to capture relevant features from the output of the BiGRU layer and the relationships between current hidden state and all the previous hidden states. This layer used encoder-decoder approach[21] which BiGRU layer acts as the encoder. First, attention layer receives hidden states [h_1, h_2, … h_T] as the input which is computed by the BiGRU layer, then the attention hidden state s_i for each target sequence y for time i can be computed using Eq. (8).

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \quad (8)$$

The context vector c_i can be calculated based on attention weight vector and the hidden state as shown in Eq. (9).

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij}\, h_j \quad (9)$$

Attention weights for each sequence can be calculated to focus on the relevant features. Attention weight vector α_{ij} can be calculated using Eq. (10)

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (10)$$

In Eq. (9), T_x is the length of the input sequences. The attention score e_{ij} can be computed using Eq. (10)

$$e_{ij} = v_a^T \tanh(\mathrm{W}_a[s_{i-1}, h_j]) \quad (11)$$

Eq. (11) calculates the similarity between the previous hidden state s_{i-1} and vector h_j where v_a and W_a are the learning parameters for applying the attention.

3.6 Output Layer

Output layer consists two dense layer, which the first dense layer is fed into second dense layer with n hidden neurons, where n is the class number of domain names. Based on which task, sigmoid activation function is applied to detection task (binary classification) and softmax activation is

applied to DGA families classification task (multiclass classification).

3.7 Hyperparameters Tuning

In our experiments, GPU-enabled TensorFlow is used with one NVIDIA Geforce RTX 3060Ti and Keras as software framework. By adjusting and optimizing the model parameters, the most effective hyperparameters are as follows:

- The length of input vector was set to 70.
- The dimension of the embedding vector was set to 100.
- The deep learning models were trained using a batch size of 100 on the training set.
- The number of hidden nodes in BiGRU layer was set to 128.
- Adam[27] was implemented as an optimization algorithm.
- Training epochs was set to 20.
- Learning rate was set to 0.0001.
- The dropout rate was set to 0.2 to prevent overfitting.

3.8 Evaluation Metrics

Standard accuracy, precision, recall, and F1-score formula is used as the classification evaluation metrics to evaluate performance of DGA malicious domain detection. The evaluation metrics can be defined as Equation (12)-(15).

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (12)$$

$$precision = \frac{TP}{TP + FP} \quad (13)$$

$$recall = \frac{TP}{TP + FN} \quad (14)$$

$$F1 = \frac{2 * precision * recall}{precision + recall} \quad (15)$$

where TP is true positives, TN is True Negatives, FP is False Positives, and FN is False Negatives respectively. In this study, DGA malicious domain names are defined as positive and the benign ones as negatives.

4. ANALYSIS RESULTS

The benign domain names are from the top 1 million domain names dataset collected by Tranco[28]. From Tranco, 950,000 domain names were selected to make the benign domain dataset. The DGA domain name samples were from the dataset collected by DGArchive[29]. Total of 30 types of DGAs were selected and 30,000 samples were chosen for each DGA types.

Table 1: Dataset Description Of Benign And DGA Domain Names

Classes	Description	Quantity
Benign	Benign names from Tranco	950,000
DGA	Domain names generated by 30 DGA families including : Bamital, Banjori, Chinad, Conficker, Corebot, Cryptolocker, DNSChanger, Dyre, Emotet, Gameover, Gozi, Locky, Matsnu, Moneromine, Murofet, Mydoom, Padcrypt, Pandabanker, Pushdo, Ranbyus, Rovnix, Sisron, Sphinx, Suppobox, Sutra, Symmi, Tinynuke, Torpig, Urlzone, and Virut .	900,000

Table 2: Dataset Distribution Of DGA Domain Names

DGA	Example	Quantity
bamital	47faeb4f1b75a48499ba14e9b1cd895a.org 9b86bb2ef4bad69cca0110076215e1f4.info	30,000
banjori	andersensinaix.com hlrfrsensinaix.com	30,000
chinad	qjsqfqluegtztu73.com aqywbjjjapgvivkaa2.com	30,000
conficker	vvyaxelso.info	30,000
corebot	ghqdorqluleja21.ddns.net	30,000
cryptolocker	dqefltweoykcxox.biz vuoykihcnuipoae.co.uk	30,000
dnschanger	pnftvoksjj.com jxtvxvfxao.com	30,000
dyre	b1ca5eebd8e0eb8ea6b61eaccbde527c26.ws ecca14cc3f5be1d665cbe8992beddc6c2d.hk	30,000
emotet	dcywmiroutolkvwu.eu djrsxvegsidbqoru.eu	30,000
gameover	gyinvzrpleiprwgdekjrgll.ru tlbjzsglldxwtlpgifikfpjij.biz	30,000
gozi	recordsdestlawsappealed.ru forbandabuseslostation.ru	30,000
locky	tugokclkeknypjb.in	30,000

	tbiilacldakrv.be	
matsnu	fatlistingsguarantee.com changingmaiden.com	30,000
moneromine r	31b4bd31fg1x2.hosting 623bf20d36fd8.tickets	30,000
murofet	mzbtfza57bygzbrl28d60 a57o41d20drowc19h14. net m59a67huhta27cziskybu ixlsb38h64iuaql28.ru	30,000
mydoom	spsmhwwrrn.biz whqeehhenr.in	30,000
padcrypt	bacfdfadbfolaffk.website lcbbblmcbflbdmdn.ga	30,000
pandabanker	a24c1dd7ea98.net 628baccb2a98.net	30,000
pushdo	qatweduqr.kz ruvuryzojezx.kz	30,000
ranbyus	sftbgrhotucscmgdr.cc yfwpmuivpmmfykven.su	30,000
rovnix	yn1cx4abl8yfp8xakt.biz r2xy266i6tas7pclt3.com	30,000
sisron	mzexmjiwmtka.net mjkxmjiwmtka.org	30,000
sphinx	bbbkmlhdjkchrxlw.com qpchwabyoylfrmha.com	30,000
suppobox	milkhello.net alexandreamadelina.ru	30,000
sutra	qmedwjlwfareaitw.mynu mber.org	30,000
symmi	pueqtepiosg.ddns.net mowaeminipedwio.ddns. net	30,000
tinynuke	cca658631a0ae5c2592c7 a694f9d120a.top 9a53d55a6d12104aa900 8947f77f3b3d.top	30,000
torpig	xccmedc.biz xckscmced.com	30,000
urlzone	lupwzhfj1a.net co1dlggj4te1su.com	30,000
virut	ipcfcl.com pbrxdw.com	30,000

In total, 1,850,000 domain names are collected from the data sources. The dataset description and distribution are as shown as Table 1 and Table 2. From the dataset distribution as shown in Table 2, some domain names such as "fatlistingsguarantee.com", "changingmaiden.com" (generated by DGA Matsnu), "milkhello.net", "alexandreamadelina.ru" (generated by DGA suppobox) consist of common and readable English words (wordlist-based DGA) so they are more difficult to detect as DGA domains because of the similarity with benign domains. Dataset is split into 80% for training set and 20% for testing.

Table 3 Type of DGA Distribution

Algorithm	DGA Malware	Count
Arithmetic-based	Banjori, Chinad, Conficker, Corebot, Cryptolocker, DNSChanger, Emotet, Gameover, Locky, Moneromine, Mydoom, Padcrypt, Pandabanker, Pushdo, Ranbyus, Rovnix, Sisron, Sphinx, Sutra, Symmi, Torpig, Urlzone, Virut	23
Hash-based	Bamital, Dyre, Murofet, Tinynuke	4
Wordlist-based	Gozi, Matsnu, Suppobox	3

Out of all the DGA classes in the dataset as shown in Table 3, there are 23 classes that use Arithmetic-based algorithms, 4 classes use Hash-based algorithms, and 3 classes use Wordlist-based algorithms. This shows that arithmetic-based DGA is the more widely used method in botnet malware that uses domain generating techniques.

In this study, number of n-grams (1-gram, 2-gram, 3-gram, and 4-gram) is experimented to get the best performance and then tested with BiGRU-Attention model. Table 4 shows the performance of each n-grams embedding.

Table 4: N-Grams Embedding Evaluation Results

N-gram	Precision	Recall	F1-score
1-gram	0.9858	0.9859	0.9859
2-grams	0.9851	0.9853	0.9852
3-grams	**0.9865**	**0.9866**	**0.9865**
4-grams	0.9845	0.9846	0.9846

From Table 4, it can be seen that the results from either 1-gram, 2-grams, 3-grams, or 4-grams don't have much difference on precision, recall, and F1-score. The embedding model with 3-grams (trigram) has the best performance between them with F1-score of 98.65%.

Three deep learning models are set to be compared with proposed BiGRU-Attention model. The models in the experiment were CNN scheme, LSTM scheme, and unidirectional GRU. Table 5 shows the evaluation results of the proposed and compared models on the DGA detection task (binary classification).

Table 5. Evaluation Results On DGA Domain Names Detection

Model	Precision	Recall	F1-score
CNN	0.9774	0.9774	0.9774
LSTM	0.9740	0.9740	0.9740
GRU	0.9781	0.9779	0.9780
BiGRU-Att	**0.9865**	**0.9866**	**0.9865**

The proposed BiGRU-Attention model could achieve a high performance with F1-score of 98.65% as shown on Table 5, higher than the three compared models. From this evaluation result, the implementation of attention mechanism could improve F1-score compared to other deep learning models which didn't use attention mechanism.

Compared and proposed models are also experimented for DGA domain classification task (multiclass classification). Table 6 shows the evaluation results of the models using precision, recall and F1-score values for benign domains and 30 types of DGA malicious domains.

Table 6. Evaluation Results On DGA Domain Names Classification

DGA	CNN			LSTM			GRU			BiGRU-ATT(Proposed)			Support
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	
benign	0.9587	0.9832	0.9708	0.9776	0.9739	0.9758	0.9863	0.9751	0.9807	0.9890	0.9813	**0.9851**	190000
bamital	0.9972	0.9985	0.9978	0.9998	1.0	0.9999	0.9997	1.0	0.9998	1.0	1.0	**1.0**	6000
banjori	0.9947	0.9980	0.9963	0.9953	0.9975	0.9964	0.9929	1.0	0.9964	0.9972	1.0	**0.9986**	6000
chinad	0.8626	0.6332	0.7303	0.9853	0.9852	0.9852	0.9676	0.9768	0.9722	0.9976	0.99	**0.9938**	6000
conficker	0.8035	0.6290	0.7056	0.7768	0.7400	0.7579	0.7692	0.7683	0.7688	0.7448	0.855	**0.7961**	6000
corebot	0.8942	0.9448	0.9188	0.9724	0.9855	0.9789	0.9772	0.9938	0.9855	0.9967	0.9943	**0.9955**	6000
cryptolocker	0.6558	0.6402	0.6479	0.7895	0.8150	0.8020	0.7806	0.7398	0.7596	0.7911	0.9158	**0.8489**	6000
dnschanger	0.8154	0.8753	0.8443	0.9271	0.9305	0.9288	0.9186	0.9575	0.9377	0.9510	0.9767	**0.9637**	6000
dyre	0.9997	0.9810	0.9902	0.9997	1.0	0.9998	0.9998	1.0	0.9999	1.0	1.0	**1.0**	6000
emotet	0.9987	0.9958	0.9972	0.9992	0.9970	0.9981	0.9975	0.9987	0.9981	0.9997	1.0	**0.9998**	6000
gameover	0.9356	0.9083	0.9218	0.9944	0.9988	0.9966	0.9965	0.9997	0.9981	0.9992	0.9998	**0.9995**	6000
gozi	0.9716	0.8937	0.9310	0.9512	0.9158	0.9332	0.9517	0.9682	0.9598	0.9686	0.9875	**0.9780**	6000
locky	0.7800	0.6235	0.6930	0.9244	0.6355	0.7532	0.852	0.6697	0.7499	0.9321	0.6888	**0.7922**	6000
matsnu	0.8332	0.5878	0.6893	0.6991	0.8427	0.7642	0.855	0.917	0.8849	0.8675	0.96	**0.9114**	6000
moneromine r	0.9974	0.9587	0.9776	0.9992	1.0	0.9996	0.9993	1.0	0.9997	0.9998	1.0	**0.9999**	6000
murofet	0.7775	0.7687	0.7730	0.7979	0.8627	0.8290	0.7852	0.8573	0.8197	0.8634	0.869	**0.8662**	6000
mydoom	0.9937	0.9938	0.9938	0.9927	1.0	0.9963	0.9947	0.9997	0.9972	0.9959	1.0	**0.9979**	6000
padcrypt	0.9930	0.9892	0.9911	0.9978	0.9963	0.9971	0.9965	0.9895	0.993	0.9995	0.9998	**0.9997**	6000
pandabanker	0.997	0.9967	0.9968	0.9985	1.0	0.9993	0.998	0.9995	0.9988	0.9985	1.0	**0.9993**	6000
pushdo	0.9876	0.9982	0.9929	0.9881	0.9992	0.9936	0.9943	0.9972	0.9958	0.9942	0.9998	**0.9970**	6000
ranbyus	0.7150	0.8303	0.7684	0.9963	0.9835	0.9899	0.978	0.9832	0.9806	0.9983	0.9948	**0.9966**	6000
rovnix	0.9997	1.0	0.9998	0.9998	1.0	0.9999	1.0	1.0	1.0	1.0	1.0	**1.0**	6000
sisron	0.7338	0.9298	0.8203	0.9149	0.9627	0.9382	0.8695	0.7398	0.7995	0.9526	0.9805	**0.9663**	6000
sphinx	0.9836	0.9892	0.9864	0.9668	0.9995	0.9829	0.9825	0.9995	0.9909	0.9883	0.9998	**0.9940**	6000
suppobox	0.9814	0.9942	0.9877	0.9789	0.9998	0.9893	0.995	0.9988	0.9969	0.9983	1.0	**0.9992**	6000
sutra	0.9977	0.9998	0.9988	0.9977	0.9998	0.9988	0.9997	0.9997	**0.9997**	0.9983	1.0	0.9992	6000
symmi	0.8908	0.7997	0.8428	0.9242	0.9830	0.9527	0.7672	0.9893	0.8642	0.9737	0.9923	**0.9829**	6000
tinynuke	0.9983	0.9978	0.9981	0.9990	0.9997	0.9993	0.9987	1.0	0.9993	0.9998	1.0	**0.9999**	6000
torpig	0.9904	0.9843	0.9874	0.9657	0.9937	0.9795	0.9893	0.9977	0.9934	0.9922	0.999	**0.9956**	6000
urlzone	0.8836	0.8682	0.8758	0.9307	0.9082	0.9193	0.9496	0.8973	0.9227	0.9597	0.9255	**0.9423**	6000
virut	0.7483	0.6808	0.7130	0.7999	0.7920	0.7959	0.7099	0.8783	0.7852	0.8374	0.8423	**0.8399**	6000
Accuracy			0.9344			0.9594			0.9599			**0.9737**	370000
Macro-Avg	0.9087	0.8862	0.8948	0.9432	0.9451	0.9429	0.9372	0.9449	0.9396	**0.9608**	**0.9662**	**0.9625**	370000

*P:Precision, R:Recall, F1:F1-score

For the DGA domain classification task (multiclass classification), BiGRU-Attention has a higher F1-score than other models in almost all DGA types. BiGRU-Attention also has better results for detecting wordlist-DGA types such as gozi, matsnu, and suppobox. From the whole evaluation results, BiGRU-Attention managed to get the highest macro-average F1-score compared to other models, which is 96.25%.

5. CONCLUSION

In this paper, n-grams embedding and attention-based BiGRU model is proposed to detect

DGA malicious domain names, not only to detect a domain is legitimate domain or DGA domain, but also to classify which DGA type of the domain. The 3-grams embedding achieved the highest F1-score result among 1-gram, 2-grams, 3-grams, and 4-grams. It can be concluded that extracting three characters sequences from domain names is the best method to prepare word vectors before classifying DGA domains using BiGRU-Attention layer. BiGRU-Attention could improve the accuracy of DGA domain detection task without manually define and handcraft domain features and achieved better performance not only on pseudo-random string DGA types generated by arithmetic-based DGA and hash-based DGA but also the wordlist-based DGA such as gozi, matsnu, and suppobox. Results from experiments demonstrate the model effectiveness that obtained an average F1-score of 98.65% for the detection and an macro average F1-score of 96.25% for the classification of DGA domain names.

The focus of future research will be on integrating this deep learning model into a larger detection system for the identification of cyber-threats in actual and real traffic data. We also intend to study newer DGA types to improve accuracy and performance of the proposed model to detect new types of DGA malicious domains.

REFERENCES:

[1] A. Karim, R. Bin Salleh, M. Shiraz, S. A. A. Shah, I. Awan, and N. B. Anuar, "Botnet detection techniques: review, future trends, and issues," *Journal of Zhejiang University: Science C*, vol. 15, no. 11, pp. 943–983, 2014, doi: 10.1631/jzus.C1300242.

[2] I. Ullah, N. Khan, and H. A. Aboalsamh, "Survey on Botnet: Its Architecture, Detection, Prevention and Mitigation," *IEEE Electrical Insulation Society Staff*, pp. 660–665, 2013.

[3] Spamhaus, "Spamhaus Botnet Threat Update Q1 2023," 2023. Accessed: Apr. 17, 2023. [Online]. Available: https://info.spamhaus.com/hubfs/Botnet%20Reports/2023%20Q1%20Botnet%20Threat%20Update.pdf

[4] M. Antonakakis *et al.*, "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware," in *Proceedingsof the 21st USENIX conference on security symposium (Security'12)*, 2012.

[5] X. D. Hoang and X. H. Vu, "An improved model for detecting DGA botnets using random forest algorithm," *Information Security Journal*, vol. 31, no. 4, pp. 441–450, 2022, doi: 10.1080/19393555.2021.1934198.

[6] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Detecting malicious domain names using deep learning approaches at scale," *Journal of Intelligent and Fuzzy Systems*, vol. 34, no. 3, pp. 1355–1367, 2018, doi: 10.3233/JIFS-169431.

[7] S. Schüppen, P. Herrmann, U. Meyer, and D. Teubert, "FANCI : Feature-based Automated NXDomain Classification and Intelligence," *Proceedings of the 27th USENIX Security Symposium*, 2018, [Online]. Available: www.usenix.org/conference/usenixsecurity18/presentation/schuppen

[8] X. Pei, S. Tian, L. Yu, H. Wang, and Y. Peng, "A Two-Stream Network Based on Capsule Networks and Sliced Recurrent Neural Networks for DGA Botnet Detection," *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 1694–1721, Oct. 2020, doi: 10.1007/s10922-020-09554-9.

[9] K. Highnam, D. Puzio, S. Luo, and N. R. Jennings, "Real-Time Detection of Dictionary DGA Network Traffic Using Deep Learning," *SN Comput Sci*, vol. 2, no. 2, pp. 1–17, 2021, doi: 10.1007/s42979-021-00507-w.

[10] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting Domain Generation Algorithms with Long Short-Term Memory Networks," *ArXiv*, Nov. 2016, [Online]. Available: http://arxiv.org/abs/1611.00791

[11] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gelmar-Padilla, "A Comprehensive Measurement Study of Domain Generating Malware," in *Proceedings of the 25th USENIX Security Simposium*, USENIX Association, 2016, pp. 263–278.

[12] K. Alieyan, A. Almomani, M. Anbar, M. Alauthman, R. Abdullah, and B. B. Gupta, "DNS rule-based schema to botnet detection," *Enterp Inf Syst*, vol. 15, no. 4, pp. 545–564, 2021, doi: 10.1080/17517575.2019.1644673.

[13] A. Gupta, H. Thakur, R. Shrivastava, P. Kumar, and S. Nag, "A Big Data Analysis Framework Using Apache Spark and Deep

Learning," *ArXiv*, Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.09279

[14] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, "Detecting DGA domains with recurrent neural networks and side information," *ACM International Conference Proceeding Series*, 2019, doi: 10.1145/3339252.3339258.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv Prepr. arXiv*, Jan. 2013, [Online]. Available: http://arxiv.org/abs/1301.3781

[16] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Trans Assoc Comput Linguist*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051/1567442/tacl_a_00051.pdf.

[17] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[18] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *ArXiv*, Jun. 2014, [Online]. Available: http://arxiv.org/abs/1406.1078

[19] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[20] K. Cho, C. Gulcehre, J. Chung, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *ArXiv*, Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.3555

[21] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *International Conference on Learning Representations*, Sep. 2015.

[22] M.-T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," Aug. 2015, [Online]. Available: http://arxiv.org/abs/1508.04025

[23] L. Zhou and X. Bian, "Improved text sentiment classification method based on BiGRU-Attention," in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Nov. 2019. doi: 10.1088/1742-6596/1345/3/032097.

[24] A. Vaswani *et al.*, "Attention Is All You Need," *31st Conference on Neural Information Processing Systems*, no. NIPS, pp. 47–82, 2017.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[26] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," *ArXiv*, Jul. 2016, [Online]. Available: http://arxiv.org/abs/1607.01759

[27] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*, Dec. 2015.

[28] V. Le Pochat, T. Van Goethem, Tajalizadehkhoob. Samaneh, M. Korczyński, and W. Joosen, "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation," *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS 2019)*, 2019, doi: 10.14722/ndss.2019.23386.

[29] D. Plohmann, "DGArchive Dataset," *Fraunhofer FKIE.[Online]. Available: https://dgarchive.caad.fkie.fraunhofer.de/.[Accessed: 12-Dec-2022]*, 2018.