Ensemble of Deep Learning Models for Detecting DGA Malware

Kuan Tung
EPFL
Vaud, Switzerland
kuan.tung@epfl.ch

Kristina Hardi, Iris Safaka, Theus Hossmann
Open Systems AG
Bern, Switzerland
{khardi,isafaka,thossmann}@open-systems.com

ABSTRACT

Domain Generation Algorithms (DGAs) are used in many modern malware families for hiding their Command & Control traffic. By randomly creating thousands of domain names that dynamically change over time, DGA-based malware traffic can not easily be detected and blocked by blocklists. Thus, Deep Learning models have been proposed to detect DGA created domain names accurately. In this work, we implement and compare the performance of eight state-of-the-art models. We find that different models have different strengths and weaknesses, suggesting that combining models may improve detection accuracy. Indeed, we find that by greedily combining three models as an ensemble classifier, we are able to reduce the False Positive Rate by 28% compared to the best performing single model.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**.

KEYWORDS

DGA, malware detection, deep learning

ACM Reference Format:
Kuan Tung and Kristina Hardi, Iris Safaka, Theus Hossmann. 2022. Ensemble of Deep Learning Models for Detecting DGA Malware. In *Proceedings of AI4Cyber/MLHat: AI-enabled Cybersecurity Analytics and Deployable Defense (AI4Cyber/MLHat).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

After infecting a device, a malware client typically connects to command and control (C&C) servers to report back and receive further instructions. This C&C traffic can easily be blocked, if the IP addresses or domain names of the C&C infrastructure are static and hard-coded into the malware. Hence, many modern malware families [17] rely on dynamic rendezvous mechanisms using domain generation algorithms (DGAs) as shown in Figure 1. A DGA-based malware will periodically generate a large number of random domain names based on seeds that are shared between the C&C infrastructure and the malware client (Step 1 in Figure 1). Some of the generated domains are registered by the attacker and bound to the IP addresses of the C&C infrastructure (Step 2). The malware client tries to resolve domains from the list (Step 3), and, once successful, contact the C&C infrastructure (Step 4).

Detecting newly created domain names used as rendezvous points accurately in near real-time is crucial for immediate response and remediation measures to contain the attack. To this end, machine learning models have been proposed to classify DGA-created domain names. Initial approaches have been based mainly on lexical and morphological handcrafted features on domain names for learning, such as length, character occurrence frequency etc. More recently, machine learning approaches using deep learning (DL) models have been proposed [2, 18, 20, 25, 27, 28, 30], typically using a character-level embedding layer [25]. These models remove the need of handcrafted features and have been shown to consistently achieve better performance for a given DGA type [21].

Creating such classifiers is particularly challenging due to the broad variety of DGAs found in different malwares, such as *Conficker* [8], *Tinba* [10], or *Rovnix* [9]. The different variants can be grouped into different classes based on how they work. A first typical grouping dimension distinguishes DGAs based on their method of selecting the random seeds. *Time-dependent* algorithms use dates or time as random seeds for generating new domains, whereas *time-independent* algorithms may use other kind of seeds such as base domain names or hexadecimal numbers. A second grouping dimension distinguishes *random-based* DGAs using pseudo-random characters (e.g., *blcgausdykl.tw*, *intgmxdeadnxuyla.com*) from *dictionary-based* DGAs that use words from embedded dictionaries (e.g., *cloudcolor.net*, *recommendations-machinery.tm*). These different types of algorithms pose different challenges for the classifiers detecting them and different DL models will have their own strengths or weaknesses towards some of these types.

In this work, we first conduct a thorough evaluation of state-of-the-art Deep Learning models for classifying DGA-created domains. To this end, we construct an extensive dataset from real-world traffic, mixed with DGA-created domains from ten different DGA malware families. We implement eight DL models, and evaluate their performance on detecting newly generated domains by these ten families. As a result of this comprehensive performance analysis, we find that each of the models has their own individual strengths and weaknesses. In particular, a specific model is able to correctly classify a subset of domain names, whereas another model has its own strengths in a *different* subset of domain names. Thus, we propose combining a subset of the models into an ensemble classifier [7]. We select the subset of models with a simple greedy algorithm aiming to reduce the False Positive Rate. Achieving a low number of false positives is crucial for a security operation center in order to avoid alert fatigue and keep the investigative workload manageable. We put emphasis on choosing different seeds when generating training and test data in order to effectively evaluate

AI4Cyber/MLHat, August 15, 2022, Washington, DC, DC, United States

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

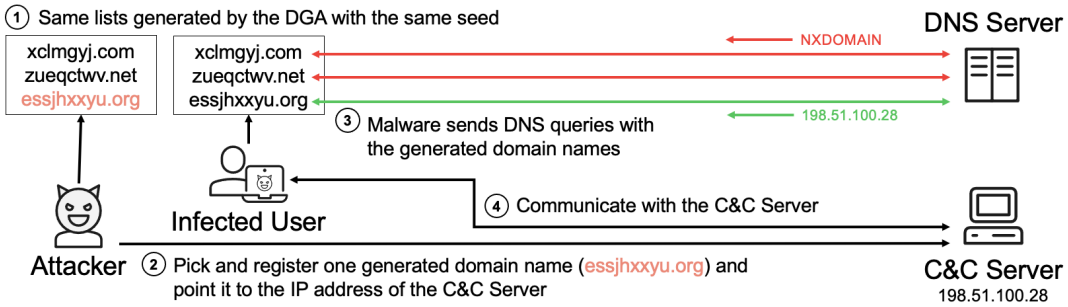

Figure 1: An example of how a DGA is used in an attack.

Table 1: DGA families in our dataset

DGA Families		
Random based	Time dependent	Conficker, Necurs, Flubot
	Time independent	Ramnit, Tinba, Banjori
Dictionary based	Time dependent	Nymaim2, Rovnix, Suppobox, Gozi

Table 2: Data source of each DGA family

Sources	DGA Families
Bambenek and Netlab Feeds [4, 16]	Necurs, Flubot
Open-source reversing scripts [1]	Conficker, Rovnix
Open-source reversing scripts [3]	Banjori, Ramnit, Tinba Nymaim2, Suppobox, Gozi

the generalization ability of the ensemble that is crucial in real-life scenarios. The best performing individual model achieves a 79.34% True Positive Rate (TPR) and a 2.32% False Positive Rate (FPR) across all the DGA families. The ensemble classifier is able to reduce the FPR by 28% while maintaining the same TPR, compared to the lowest FPR single model.

In Section 2 we present related work. In Section 3 we describe the model evaluation process, including the dataset creation, the evaluated models and their implementation details, and finally the performance evaluation results. In Section 4 we propose and evaluate an ensemble classifier, and we conclude in Section 5.

2 RELATED WORK

In [2], the length of the domain names, the distribution statistics of bi-gram and tri-gram, and the character occurrence frequency are used as features and a reputation engine is trained to decide if a domain name is similar to benign or malicious domain names. The entropy, bi-grams, and length of different levels in a domain name are compared with the training data to compute the level characteristic value of the domain name and used as features in [30]. [20] uses a set of linguistic features in the domain name such as the meaningful characters ratio and n-gram normality score, clustering, and fingerprinting to tell DGA and non DGA-generated domain names apart. These approaches have good interpretability but do not achieve as high accuracy as deep learning approaches that learn the features automatically and do not require handcrafted ones.

Deep learning approaches leverage the fact that informative features can be learned within the model itself. Therefore, raw or only slightly processed domain names are used as input to the models instead of human defined features. Since the characters in a domain name are placed in a sequential structure, [25] proposes using a Long-Short-Term-Memory (LSTM) model, a type of recurrent neural network which was designed for time sequence, to process the characters and generate the prediction. More recent approaches include BiLSTM [28], Convolutional Neural Networks (CNNs) [27], LSTM + CNN [12, 28], and attention [18]. These models claim to out-perform the previously proposed models, or tackle real-life problems like the difficulty of getting labeled data [27] and achieving real-time detection [12].

Other methods combine lexical features of domain names with side information, such as WHOIS information [13]. In [5], a recurrent neural network and domain registration side information are used to predict DGA domain names. Focusing on hardening DGA classifiers against adversarial attacks, [22] trains and evaluates deep learning and random forest classifiers with easily obtainable side information such as the length of the RData field, the geo-location of the resolved IP address, and the time-to-live (TTL) value of the DNS query. While leveraging side information can certainly help achieving higher accuracy, in this paper we focus on evaluating and optimizing classification using the domain name only. Extending our approach for leveraging NXDOMAIN responses and WHOIS information is the natural next step.

3 MODEL EVALUATION

We present in this section our evaluation and performance comparison of eight state-of-the-art deep learning models for DGA classification. We first describe the dataset we create, then the implementation of the different individual models, and finally the performance results.

3.1 Dataset

We create a dataset consisting of benign (*negative*) and DGA-created (*positive*) samples. We collect the benign domains from a very large volume of real-world network traffic traces (proxy logs) from a Managed Security Service Provider (MSSP) between May and November 2021. This results in around 500K unique domain names

across different organizations and companies of various industries around the globe. As positive data we consider DGA domain names collected from various online DGA feeds (Bambenek [4] and Netlab [16]) or generated with open-source reversing scripts [1, 3]. We use ten DGA families in our dataset, including random-based (RB) and dictionary-based (DB) ones as listed in Table 1. Table 2 shows the data source of each DGA family. We choose these families based on having a high impact [11], trending and recent activity [26]. For each of the ten families we include 30K DGA domain names in our dataset and sample the same amount of domain names from the benign domains to create a balanced dataset. In total, there are 300K unique benign domain names and 300K unique DGA domain names. The dataset is split into training and test set with a 7:3 ratio. We use different seeds and dictionaries in the training and test set to ensure that we are testing the generalization ability of the models. The only exceptions are for *Nymaim2* and *Rovnix*, one dictionary is used because it is the only one we have access to.

3.2 Evaluated Models

We choose eight state-of-the-art deep learning models to implement and compare. They all have a character-level embedding layer, where each character appearing in a domain is represented by a learned 128-dimension vector. They can be grouped as LSTM-based, CNN-based, LSTM-CNN-based, and Transformer-based, depending on the underlying architecture. We implement all models with Keras [14] and train them using the NVIDIA GeForce GTX 1080 Ti GPU. The details of each model are provided in the following subsections.

3.2.1 *LSTM-based.*

LSTM - Proposed in [25], this model consists of an embedding layer, a LSTM layer with 128 hidden units, and an output layer with sigmoid activation function which provides the probability of the classification. There are 144,641 parameters, and it is the smallest model we use. LSTM was designed for long sequences and is able to keep long-term dependencies.

CMU (BiLSTM) - Initially proposed in [6] for tweet classification and modified in [28] for DGA classification. It consists of an embedding layer, a bidirectional LSTM layer with 64 hidden units, and an output layer. There are 1,802,113 parameters. By having a bidirectional LSTM layer, the characters are processed from both left-to-right and right-to-left.

3.2.2 *CNN-based.*

NYU (Stacked CNN) - Originally proposed in [29] and modified in [27] for DGA classification. The model uses an embedding layer, two stacked convolutional neural network (CNN) layers, two following dense layers with 1024 hidden units, and the final output layer. A CNN layer has a convolution layer and a pooling layer. The convolution layer contains a set of fixed size filters which slide through the input to retrieve information about the character sequences. The pooling layer is applied next to reduce the sensitivity and dimensionality. The two CNN layers have 128 filters of size 3 and 128 filters size 2, respectively. There are 2,581,121 parameters, and it is the largest out of the eight models.

Invincea (Parallel CNN) - Proposed in [19] for detecting malicious entities and used for DGA classification in [27]. It consists of an embedding layer, a CNN layer with 1024 filters, two following dense layers with 1024 hidden units, and the final output layer. The CNN layer in this model is parallel instead of stacked (see Figure 2). There are 256 filters with sizes 2, 3, 4, and 5, to which the input is sent simultaneously. There are 276,353 parameters in total. All the outputs of the different filters are concatenated and sent to the following dense layer.

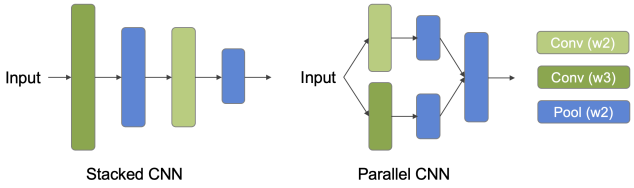

Figure 2: Stacked and parallel CNN model architectures.

3.2.3 *LSTM-CNN-based.*

MIT (CNN + LSTM) - Originally proposed in [24] for learning tweet embeddings and adapted for DGA classification in [28]. The model consists of an embedding layer, a CNN layer with 128 filters of size 3, a LSTM layer with 64 hidden units, and an output layer which provides the probability of the classification. Each layer is stacked sequentially. There are 193,921 parameters.

Bilbo (Parallel CNN + LSTM) - Proposed in [12], this model consists of an embedding layer, a LSTM layer with 256 hidden units, a parallel CNN layer with 300 filters (60 filters with sizes 2, 3, 4, 5 and 6), a dense layer with 100 hidden units, and a output layer. There are 617,469 parameters. It is called the bagging hybrid model. After the input is encoded by the embedding layer, the encoding is sent both to the parallel CNN layer and the LSTM layer. The outputs of these two layers are then concatenated and sent to the dense layer and the final output layer.

Att-CNN-BiLSTM - Proposed in [18], this model consists of an embedding layer, a CNN layer with 128 filters of size 3, a bidirectional LSTM layer with 128 hidden units, an attention layer, a dense layer with 128 hidden units, and an output layer. Each layer is stacked sequentially. There are 358,401 parameters. The attention layer provides the ability to take the hidden states at all positions of the BiLSTM layer. The authors claim that it can assist in detecting dictionary-based DGA and discovering critical parts of DGA domain names.

3.2.4 *Transformer-based.*

Transformer classifier - The Transformer model was initially proposed in [23]. It has been the core of many state-of-the-art natural language processing applications. We implement a transformer classifier that consists of an embedding layer, a transformer block with 3 heads, a global pooling layer, a dense layer with 128 hidden units, and a output layer. The total number of parameters is 266,113. The transformer block was implemented as proposed in the official Keras tutorial [15].

3.3 Results

For comparing the performance of the various models, we train and test all of them using our dataset. During training, we use early stopping to make sure the model has converged and prevent

Table 3: Performance evaluation results

Models		AUC	TPR	FPR
LSTM based	LSTM	0.9450	0.7513	0.0247
	CMU	0.9458	0.7651	0.0247
CNN based	NYU	0.8932	0.7446	0.0541
	Invincea	**0.9623**	**0.7934**	0.0406
LSTM-CNN based	MIT	0.9514	0.7627	0.0244
	Bilbo	0.9539	0.7380	**0.0232**
	Att-CNN-BiLSTM	0.9306	0.7317	0.0273
Transformer based	Transformer classifier	0.9360	0.7090	0.0570
Ensemble (Bilbo + NYU + Transformer)	Majority vote	N.A.	0.7362	0.0181
	Averaging scores	**0.9545**	**0.7355**	**0.0168**

Table 4: True Positive Rate (TPR) per DGA family

DGA Families		Bilbo	NYU	Transformer Classifier	Ensemble (Averaging scores)
Random based	Conficker	**0.9322**	0.8085	0.9108	0.9296
	Necurs	**0.9901**	0.9499	0.9707	0.9893
	Flubot	**0.9989**	0.9931	0.9937	0.9993
	Ramnit	0.9719	**0.9721**	0.9260	0.9733
	Tinba	0.8668	**0.9853**	0.9041	0.9758
	Banjori	0.2438	**0.3103**	0.2767	0.1002
Dictionary based	Nymaim2	**0.9286**	0.7581	0.8256	0.9188
	Rovnix	0.9828	**0.9854**	0.9065	0.9960
	Suppobox	0.0631	0.1107	**0.1761**	0.0289
	Gozi	0.3783	**0.6159**	0.4425	0.4441

overfitting. We use 20% of the training data for computing the validation loss and we set the patience to 5 epochs. We use the Area Under the Receiver Operating Characteristic Curve (AUC), True Positive Rate (TPR) and False Positive Rate (FPR) as evaluation metrics. We set a threshold of 0.5 to decide if the domain names are benign (<0.5) or DGA-created (>0.5).

Table 3 shows the performance evaluation results of the individual models, including all considered DGA families. The *Invincea* model consisting of parallel CNN layers has the highest TPR of 79.34%, and the *Bilbo* model with parallel CNN and LSTM layers has the lowest FPR of 2.32%.

Dissecting the results per DGA family, we compute TPRs to investigate the ability of each model to detect specific DGA families. The family-wise TPRs of a subset of the models (those included in the ensemble classifier, see Section 4) are shown in Table 4. All models have high TPRs, ranging from 80.85% to 99.89%, on most of the random-based families. But they struggle against *Banjori*, with TPRs ranging from 24.38% to 31.03%. For dictionary-based families, they perform well on *Nymaim2* and *Rovnix*, with TPRs between 75.81% and 98.54%. However, for the other two dictionary-based families (*Suppobox* and *Gozi*), they have low TPRs, ranging from 6.31% to 61.59%. The relatively poor performance on *Banjori* and the two DB families explains why we could not get higher overall TPRs. It is important to note that it is difficult even for humans to classify domains generated by DB families. And the dictionaries used to generate DGA domain names are different in the training and the test set.

One important observation is that no single model reaches the highest TPR across all DGA families. Instead different models show strong performance for different families. *Bilbo* has the highest TPR for *Conficker*, *Necurs*, *Flubot* and *Nymaim2*, whereas *NYU* has the highest TPR for *Ramnit*, *Tinba*, *Banjori*, *Rovnix* and *Gozi*. *Transformer* performs best for *Suppobox*.

4 THE ENSEMBLE CLASSIFIER

Observing the performance results for the individual models, it is evident that no single model alone can achieve optimal performance across all DGA families. This is a rather expected result, since different model architectures are optimised to learn different characteristics on the training domains. The question is, can we effectively combine the outputs of these models in order to achieve better aggregated performance across all DGA families?

To this end, we propose to create an ensemble classifier using a subset $\mathcal{M} \subseteq \mathcal{N}$ of the set $\mathcal{N}$ of implemented models (where $|\mathcal{N}| = 8$ in our implementation). The goal is to choose $\mathcal{M}$ such that the training FPR across all DGA-families is minimized. To avoid the prohibitively expensive evaluation of all $\binom{|\mathcal{N}|}{|\mathcal{M}|}$ subsets, we use a simple greedy selection algorithm. First, from $\mathcal{N}$ we choose the model achieving the lowest number of false positive predictions. Then, we compute the union of the false positive predictions of the selected model in the previous step and of the rest $|\mathcal{N}| - 1$ models; we pick the model contributing the smallest number of elements in the union, to place in set $\mathcal{M}$, and we remove it from set $\mathcal{N}$. The previous steps are repeated $|\mathcal{M}| - 1$ times more.

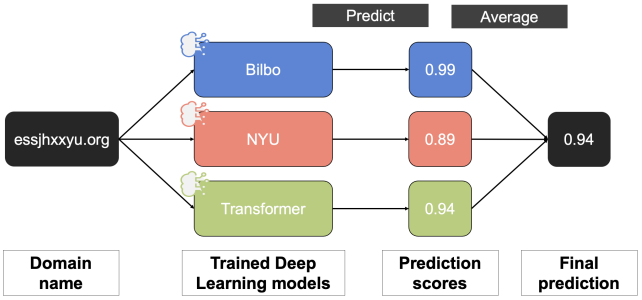

Figure 3: Ensemble with averaging scores.

To combine the verdicts of the $\mathcal{M}$ chosen classifiers we evaluate two methods: (1) majority voting and (2) averaging scores. In Table 3 we see that the results for both methods are very similar, with slightly better performance using the averaging of the scores. Figure 3 shows how the domain classification is performed using the ensemble of models with averaging scores. In the future, we plan to evaluate more sophisticated combination methods, such as using the verdicts as inputs to a new classifier.

We find that choosing $|\mathcal{M}| = 3$ provides a good trade-off between the accuracy of the resulting ensemble classifier and its performance (training time and scoring time)[1]. The chosen three models are *Bilbo* (Parallel CNN + LSTM), *NYU* (Stacked CNN), and *Transformer classifier*. With the ensemble combining these three

[1]Note that with less constraints on training and scoring time we could also select more than 3 models or even use all of them.

models, we achieve an overall FPR of 1.68%, which is a reduction by 28% compared to the lowest FPR individual model *Bilbo*, while keeping the TPR roughly at the level of using *Bilbo* alone. Note that 1.68% FPR may seem too high for a system that generates alerts, which would result in alert fatigue. However, in our system we use the DGA classifier for *decision support*, providing context information to a human security analyst for an ongoing security investigation. For such a decision support system this level of FPR is acceptable.

In Table 4 we see that the relatively low average TPR of 73.55% is largely due to poor performance (of the individual models as well as the ensemble) on the *Banjori* and two dictionary-based families (*Suppobox* and *Gozi*); all other families have TPRs above 90%, even ranging up to 99.93% for *Flubot*. The poor performance with *Banjori*, is due to different seed domains used to generate the training and test domain names. In general, DB DGA families are particularly difficult to classify because the underlying dictionaries that they use to generate domain names can change during inference. In our experiment, we simulated this situation by using different dictionaries in the training and test set for *Suppobox* and *Gozi*. The results of the other two DB families (*Nymaim2* and *Rovnix*) are a lot better because only one dictionary is used to generate the domain names in both the training and test set. Incorporating side information other than the domain names is needed to address this problem.

5 CONCLUSION

In a thorough performance comparison we have implemented and evaluated eight state-of-the-art deep learning models for detecting DGA-created domain names from ten different malware families, both random and dictionary-based. Using our large dataset from real traffic, we show that these models have their own strengths and weaknesses. Different models show good performance on different DGA malware families, and no individual model achieves the best performance across all of them. Therefore, we use a greedy selection method to combine a subset of the models into an ensemble classifier. Our results show that the ensemble classifier can achieve a lower False Positive Rate across all families while keeping the True Positive Rate high.

REFERENCES

[1] Andrey Abakumov. 2022. *DGA*. https://github.com/andrewaeva/DGA
[2] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. 2010. Building a Dynamic Reputation System for DNS. In *19th USENIX Security Symposium (USENIX Security 10)*.
[3] Johannes Bader. 2022. *Domain Generation Algorithms*. https://github.com/baderj/domain_generation_algorithms/
[4] Bambenek Consulting. 2022. *OSINT Feeds*. https://osint.bambenekconsulting.com/feeds/
[5] Ryan R Curtin, Andrew B Gardner, Slawomir Grzonkowski, Alexey Kleymenov, and Alejandro Mosquera. 2019. Detecting DGA domains with recurrent neural networks and side information. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*. 1–10.
[6] Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481* (2016).
[7] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer, 1–15.
[8] Malpedia (Fraunhofer FKIE). 2022. *Conficker*. https://malpedia.caad.fkie.fraunhofer.de/details/win.conficker
[9] Malpedia (Fraunhofer FKIE). 2022. *Rovnix*. https://malpedia.caad.fkie.fraunhofer.de/details/win.rovnix
[10] Malpedia (Fraunhofer FKIE). 2022. *Tinba*. https://malpedia.caad.fkie.fraunhofer.de/details/win.tinba
[11] Federal Office for Information Security of Germany. 2022. *Profile of Botnet*. https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Methoden-der-Cyber-Kriminalitaet/Botnetze/Steckbriefe-aktueller-Botnetze/steckbriefe-aktueller-botnetze_node.html
[12] Kate Highnam, Domenic Puzio, Song Luo, and Nicholas R Jennings. 2021. Real-time detection of dictionary dga network traffic using deep learning. *SN Computer Science* 2, 2 (2021), 1–17.
[13] ICANN. 2022. *Registration data lookup tool*. https://lookup.icann.org/en
[14] Keras. 2022. *The Python deep learning API*. https://keras.io/
[15] Keras. 2022. *Text classification with Transformer*. https://keras.io/examples/nlp/text_classification_with_transformer/
[16] Netlab. 2022. *DGA OpenData Project,*. https://data.netlab.360.com/dga/
[17] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. 2016. A comprehensive measurement study of domain generating malware. In *25th USENIX Security Symposium (USENIX Security 16)*. 263–278.
[18] Fangli Ren, Zhengwei Jiang, Xuren Wang, and Jian Liu. 2020. A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network. *Cybersecurity* 3, 1 (2020), 1–13.
[19] Joshua Saxe and Konstantin Berlin. 2017. eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. *arXiv preprint arXiv:1702.08568* (2017).
[20] Stefano Schiavoni, Federico Maggi, Lorenzo Cavallaro, and Stefano Zanero. 2014. Phoenix: DGA-based botnet tracking and intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 192–211.
[21] Raaghavi Sivaguru, Chhaya Choudhary, Bin Yu, Vadym Tymchenko, Anderson Nascimento, and Martine De Cock. 2018. An evaluation of DGA classifiers. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 5058–5067.
[22] Raaghavi Sivaguru, Jonathan Peck, Femi Olumofin, Anderson Nascimento, and Martine De Cock. 2020. Inline detection of dga domains using side information. *IEEE Access* 8 (2020), 141910–141922.
[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[24] Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. 2016. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 1041–1044.
[25] Jonathan Woodbridge, Hyrum S Anderson, Anjum Ahuja, and Daniel Grant. 2016. Predicting domain generation algorithms with long short-term memory networks. *arXiv preprint arXiv:1611.00791* (2016).
[26] IBM X-Force. 2022. *Botnet Report*. https://exchange.xforce.ibmcloud.com/botnet
[27] Bin Yu, Jie Pan, Daniel Gray, Jiaming Hu, Chhaya Choudhary, Anderson CA Nascimento, and Martine De Cock. 2019. Weakly supervised deep learning for the detection of domain generation algorithms. *IEEE Access* 7 (2019), 51542–51556.
[28] Bin Yu, Jie Pan, Jiaming Hu, Anderson Nascimento, and Martine De Cock. 2018. Character level based detection of DGA domain names. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
[29] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems* 28 (2015).
[30] Ying Zhang, Yongzheng Zhang, and Jun Xiao. 2013. Detecting the DGA-based malicious domain names. In *International Conference on Trustworthy Computing and Services*. Springer, 130–137.