# Ensuring Reproducibility in AGD Detection: A Model Comparison Framework

Tomás Pelayo-Benedet[1], Ricardo J. Rodríguez[1] and Carlos H. Gañán[2]

[1] Dpto. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain
[2] Delft University of Technology, the Netherlands
`tpelayo@unizar.es`

**Abstract.** Domain generation algorithms are commonly used by malware to generate command and control domains to contact during execution, avoiding having to use fixed IP addresses or DNS domains, which are easily blockable. During the last years, many solutions have been proposed for the detection of algorithmically generated domains (AGDs) based on artificial intelligence. However, there is no common umbrella that allows experiments to be replicated under the same conditions, making it difficult to ensure how good one solution is compared to the others. To address the current lack of a common environment for model comparison, in this work we present a framework focused on training and comparing artificial intelligence models for AGDs detection. As a use case, we have implemented and evaluated the models proposed in the latest works in this field, showing its applicability.

**Keywords:** Domain Generation Algorithms · Machine Learning · Deep Learning · Malware · Malware detection

## 1 Introduction

Command and Control (C&C) is one of the stages of a cyberattack, according to the Cyber Kill Chain [5] model. During this stage, the attacker sets up a C&C channel to remotely manage the compromised system, allowing them to send commands and control malware that has been installed on the target system.

Domain Generation Algorithms (DGAs) emerged since 2009 [6] as a way to avoid traditional methods of blocking known C&C domains and IP addresses. By constantly generating new domain names (known as *algorithmically generated domains* or AGDs), DGAs make it difficult for security systems to block or blacklist all possible domains that malware might use for communication.

The detection of malicious AGDs has been an important research topic in the last 15 years [6]. Different detection solutions have been proposed, mainly based on artificial intelligence [3, 8–12]. However, a common pitfall in the literature is the difficulty in comparing different proposed models [2,4], mainly due to the fact that they do not freely share their implementations (thus hiding internal details of the model configuration) and because the datasets used in the evaluation are not provided (thus making a fair comparison difficult).

To address these issues, we propose a framework for training and comparing artificial intelligence models for DGA detection. This standardized procedure enables the creation and comparison of new models in a consistent experimentation environment, thereby enhancing detection capabilities.

## 2 Our Framework and Preliminary Results

Our framework consists of two components: the *Core* module, which includes all execution logic, and the *Dataset Manager* module, which manages dataset processing. New models must adhere to the *Classifier* schema, and users need to define the *Data Element* and *Result*, representing the training data and evaluation metrics, respectively.

This framework has been used to train and compare several models found in the literature and reproducible [3,8–12]. We built a dataset composed of 250,000 malicious domains obtained from [7] (from 76 different malware families) and 250,000 non-malicious domains that have been obtained from the Tranco list [1]. This dataset was split 70%/15%/15% into training, validation, and test datasets, respectively. Metric results on the test datasets are shown in Table 1, where the best result for each metric is highlighted.

The results show that simpler models tend to achieve better results when considering a large number of different families. Due to their simplicity, they need to generalize more and are therefore more robust to detect DGAs from different families. For the sake of open science, the framework and the implementation of models are released under GNU/GPLv3 on our GitHub [**ANONYMIZED**].

| **Model (Year)** | **Acc** | **Prec** | **Rec** | **F1** | **FPR** | **TPR** | **MCC** | $\kappa$ |
|---|---|---|---|---|---|---|---|---|
| LSTM [9] (2016) | 95.42 | 97.39 | 95.69 | 96.53 | 5.12 | 95.69 | 89.82 | 0.8045 |
| LSTM [11] (2017) | 95.44 | 97.25 | 95.87 | 96.55 | 5.40 | 95.87 | 89.84 | 0.8059 |
| CNN [11] (2017) | 94.96 | 97.39 | 94.98 | 96.17 | 5.07 | 94.98 | 88.86 | 0.7849 |
| LSTM [10] (2018) | 95.02 | 96.82 | 95.67 | 96.24 | 6.27 | 95.67 | 88.88 | 0.7896 |
| CNN [10] (2018) | 92.94 | 96.29 | 92.99 | 94.61 | 7.16 | 92.99 | 84.49 | 0.7056 |
| CMU [12] (2018) | 94.87 | **97.46** | 94.77 | 96.10 | **4.92** | 94.77 | 88.69 | 0.7810 |
| MIT [12] (2018) | **95.48** | 96.96 | **96.23** | **96.59** | 6.03 | **96.23** | **89.87** | **0.8083** |
| Parallel CNN [12] (2018) | 93.48 | 96.64 | 93.48 | 95.03 | 6.49 | 93.48 | 85.68 | 0.7265 |
| Baseline [12] (2018) | 86.51 | 93.36 | 85.87 | 89.46 | 12.19 | 85.87 | 71.31 | 0.4745 |
| MLP [12] (2018) | 92.59 | 96.41 | 92.32 | 94.32 | 6.86 | 92.32 | 83.84 | 0.6907 |
| CNN [3] (2019) | 95.28 | 97.08 | 95.81 | 96.44 | 5.76 | 95.81 | 89.48 | 0.7998 |
| Max Pooling [3] (2019) | 90.48 | 95.62 | 89.84 | 92.64 | 8.21 | 89.84 | 79.53 | 0.6107 |
| LSTM [3] (2019) | 92.40 | 96.98 | 91.44 | 94.13 | 5.68 | 91.44 | 83.67 | 0.6804 |
| LSTM + CNN [3] (2019) | 83.88 | 94.12 | 80.87 | 86.99 | 10.09 | 80.87 | 67.44 | 0.3796 |
| Bidireccional [3] (2019) | 93.40 | 95.92 | 94.10 | 95.00 | 8 | 94.10 | 85.33 | 0.7261 |
| DBD [8] (2019) | 94.19 | 96.92 | 94.28 | 95.58 | 5.98 | 94.28 | 87.18 | 0.7545 |

**Acc**: accuracy; **Prec**: Precision; **Rec**: Recall; **F1**: F1-score; **FPR**: False Positive Rate; **TPR**: True Positive Rate; **MCC**: Matthews's Correlation Coefficient; $\kappa$: Cohen's Kappa Score

Table 1: Results obtained with our framework from different models.

## References

1. Tranco List. [Online; `https://tranco-list.eu/`], accessed on 1 August, 2023.
2. Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Lessons Learned on Machine Learning for Computer Security. IEEE Security & Privacy **21**(5), 72–77 (2023)
3. Berman, D.S.: DGA CapsNet: 1D Application of Capsule Networks to DGA Detection. Information **10**(5) (2019)
4. Botacin, M., Ceschin, F., Sun, R., Oliveira, D., Grégio, A.: Challenges and pitfalls in malware research. Computers & Security **106**, 102287 (2021)
5. Lockheed Martin: The Cyber Kill Chain. [Online; `https://lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html`], accessed on 22 August, 2023.
6. Porras, P.A., Saïdi, H., Yegneswaran, V.: A Foray into Conficker's Logic and Rendezvous Points. LEET **9**, 7 (2009)
7. Tuan, T.A., Anh, N.V., Luong, T.T., Long, H.V.: UTL_DGA22 A Dataset for DGA Botnet Detection and Classification. Computer Networks **221**, 109508 (2023)
8. Vinayakumar, R., Soman, K.P., Poornachandran, P., Alazab, M., Jolfaei, A.: DBD: Deep Learning DGA-Based Botnet Detection, pp. 127–149. Springer International Publishing (2019)
9. Woodbridge, J., Anderson, H.S., Ahuja, A., Grant, D.: Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. CoRR **abs/1611.00791** (2016)
10. Yang, L., Liu, G., Zhai, J., Dai, Y., Yan, Z., Zou, Y., Huang, W.: A Novel Detection Method for Word-Based DGA. In: Sun, X., Pan, Z., Bertino, E. (eds.) Cloud Computing and Security. pp. 472–483. Springer International Publishing (2018)
11. Yu, B., Gray, D.L., Pan, J., Cock, M.D., Nascimento, A.C.A.: Inline DGA Detection with Deep Networks. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW). pp. 683–692. IEEE (2017)
12. Yu, B., Pan, J., Hu, J., Nascimento, A., De Cock, M.: Character Level based Detection of DGA Domain Names. In: 2018 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2018)