DGAIntel: Targeted Identification of Domain Generation Algorithms with Convolutional-Recurrent Networks

Rushil Mallarapu[1†]
[1]*Fairfield Ludlowe High School, 785 Unquowa Av., Fairfield, CT*
[†]*Email Address: rushil.mallarapu@gmail.com*

Modern malware makes use of domain generation algorithms (DGAs) to establish communications with a command and control (C&C) center, enabling it to execute malicious activities. DGAs generate thousands of domain names of which only a small percentage have been registered, to communicate with the botmaster, presenting an asymmetric challenge to defenders due to the scope of the detection problem. Despite the success of novel deep learning architectures to recognize maliciously generated domains, few implementations of such algorithms exist, hampering the adoption of the technology in security applications. Our research asked whether state-of-the-art DGA detection models could be improved upon for application to threat intelligence pipelines. We report the development of DGAIntel, a deep learning model that can identify whether a domain name is genuine or maliciously generated without auxiliary information. The model uses a convolution-recurrent architecture to quickly extract short and long distance information in domain names. Experiments show better or comparable performance to state-of-the-art DGA detection models. We develop a lightweight package that allows professionals to utilize the model for security applications. This work is a major implementation of deep learning architectures applied to DGA detection, and enables the integrated usage of DGA intelligence in system defense.

Keywords: Domain Generation, Deep Learning, Text Analysis, Cybersecurity

I. INTRODUCTION

Malicious software, or malware, is used for a variety of unauthorized actions, from information theft to targeted denial of computational resources. Malware is estimated to cost the national economy between $57 billion and $109 billion, making it imperative to develop novel technologies to mitigate its ability to achieve its goals [1]. For it to do so, malware needs to communicate with a command and control (C&C) center, to relay instructions, collect information, and receive updates. Botmasters – C&C controllers – used to hardcode IPs to allow communicating with their malware, leading to the connections being shut down upon discovery of the malware. To address this, malware engineers designed domain generation algorithms (DGAs) to allow C&C communication with malware could persist even after detection.

Embedded into many forms of malware, domain generation algorithms work by randomly producing thousands of domain names and attempting to resolve all of them against its DNS server [2]-[3]. The botmaster only needs to have registered a few of these domain names. When the malware-embedded DGA generates a domain that the botmaster has registered, it is able to establish a valid IP address and communicate with the C&C center. As such, detecting DGA domains presents an asymmetric challenge to defenders: with standard blacklisting, defenders would need to keep registering generatable domains, while attackers would only need one successful connection to communicate or execute malware [4]. DGA-based malware includes Conficker, Stuxnet (the malware that attacked Iranian nuclear facilities), and Flame [5]. As such, cybersecurity methodologies require an intelligent way of detecting and responding to the use of DGAs by malware in order to assure network security.

The binary classification problem this research addresses – detect whether a domain is maliciously generated or a genuine address – is well-researched in information security [5]. Knowledge of whether a domain name itself is genuine or malicious is especially useful, as gathering contextual information can present an additional overhead or may be unavailable within certain situations. Some algorithms use WHOIS and NXDomain records to gather rough predictions on whether a domain name is DGA. More recently, Curtin *et al.* proposed an RNN architecture for detecting DGA domains based on domain name and WHOIS data, but they acknowledged that their model was limited by disputes regarding the ability for open access to domain metadata [6].

Standard techniques for DGA detection with machine learning over domain names involve classification based on engineered text features, but this makes the model both vulnerable to adversarial training and impeded in detecting classifiable boundaries when observed DGA features overlap with those of genuine domains [5]. The use of deep learning methodologies enables DGA detection algorithms to avoid the dependence on feature engineering, as such models learn implicit features. As such, active research on DGA detection focuses on developing various deep learning architectures suited to the task, see e.g. [5]-[7]. We report the development of DGAIntel, which implements a convolutional-recurrent architecture for binary DGA

classification in a lightweight package easily deployable for usage by cybersecurity architects.

II. METHODS

1. Data

We combined data from three varied sources; the top million domain names on Alexa, the Bambenek DGA feed, and the Splunk DGA detection dataset. Alexa records the top one million domain names with respect to the number of page views by unique visitors [8]. We implicitly assume the domains in the Alexa dataset are genuine domains.

The Bambenek Consulting DGA domain feed provided 700,000 known DGA domains. This feed collects DGA domains from known DGA families in live network traffic by comparing domain names generated from known DGA algorithms to those encountered in network traffic [9]. It is important to note that DGAIntel is designed to perform binary detection of DGA domains, agnostic of DGA family, which allows the model to be effective against previously unknown malware.

The Splunk DGA dataset contains 100,000 mixed examples of DGA and genuine domains [10]. The dataset was originally curated to demonstrate the difficulty in classifying DGA domains via standard string informatics metrics (n-grams, entropy, etc.).

The dataset was randomly split into training (1,500,000 examples), validation (100,000 examples), and test (10,000 examples) sets. Furthermore, to prevent the model from developing an over-reliance on the TLD as a classification benchmark, 25% of the domain names were stripped of their TLD and subdomain, allowing the model to better recognize malicious characteristics in the base domain name.

2. Architecture

At a high level, DGAIntel uses a stacked convolutional-recurrent architecture inspired by state-of-the-art models for text mining [11], [12]. This allows the model to learn classifiable differences over both short-distance and long-distance patterns in domain names. The model architecture is derived from one presented in Yu *et al.*, with modifications made to improve generalizability and performance [5]. The string passed as input to the model consists of the full domain name. This name is then converted into an integer sequence via a static capitalization-agnostic embedding and padded to a maximum character length of 81. It is then passed through a learnable vector embedding that converts each character into a 128-dimensional vector encoding its information. This results in the domain name being converted as an 81 by 128 matrix, where each column represents one character.

The embedded domain name is first passed through a convolutional neural network (CNN). CNNs are well known for their ability to recognize local informational correlations, especially in the domain of 2D image processing. They have also been used in their 1D variant to

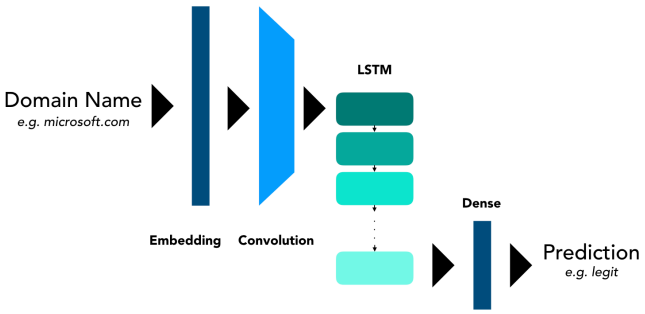

FIG. 1. Diagram of DGAIntel network architecture. The domain name is first embedded into a high-dimensional vector encoding. It is then passed through sequential convolutional and LSTM layers to further encode its lexical information. Finally, a dense layer uses the embedded domain vector to predict whether the domain name is legitimate or malicious.

to perform "temporal" classification of character sequences. In this instance, we use a single CNN layer to further embed the character-level domain name in a context aware manner.

The vector is then passed through a long short-term memory layer (LSTM), a specialized type of recursive neural network (RNN). LSTMs are known for their state of the art performance on sequence-related learning. Therefore, they are a natural choice for classifying patterns in text sequences, such as domain names. We use a single LSTM layer, which embeds the entire domain name into a 64 by 1 vector.

The network finally performs binary classification on this CNN-LSTM embedded domain vector to predict whether the input domain name is or is not DGA. The full network architecture is shown in Figure 1. We use a shallow network architecture to enable both fast predictions and to prevent over utilization of computational resources. The results below show we are able to maintain state-of-the-art performance despite this shallow architecture.

3. Implementation

The network architecture described above was implemented in Python. We utilized Google's Tensorflow framework with the Keras API to build and train the network architecture [13]. As a main goal of this research is to enable widespread utilization of deep learning in malware detection, we make all our code available in the DGAIntel package for Python, which can be accessed on GitHub and PyPI. See appendix for access.

The model was implemented with the intention of being lightweight, intuitive, and uncomplicated. As such, it only supplies three objects. The "get_prob" function returns the predicted likelihood of the input domain names being DGA, where a prediction score of one indicates that the domain name is DGA. The "get_prediction" implicitly uses that function to return full strings describing whether domain names in a list or text file are DGA or not. Finally, the "Intel" class allows for whitelisting support, enabling the user to whitelist certain SLDs and TLDs while

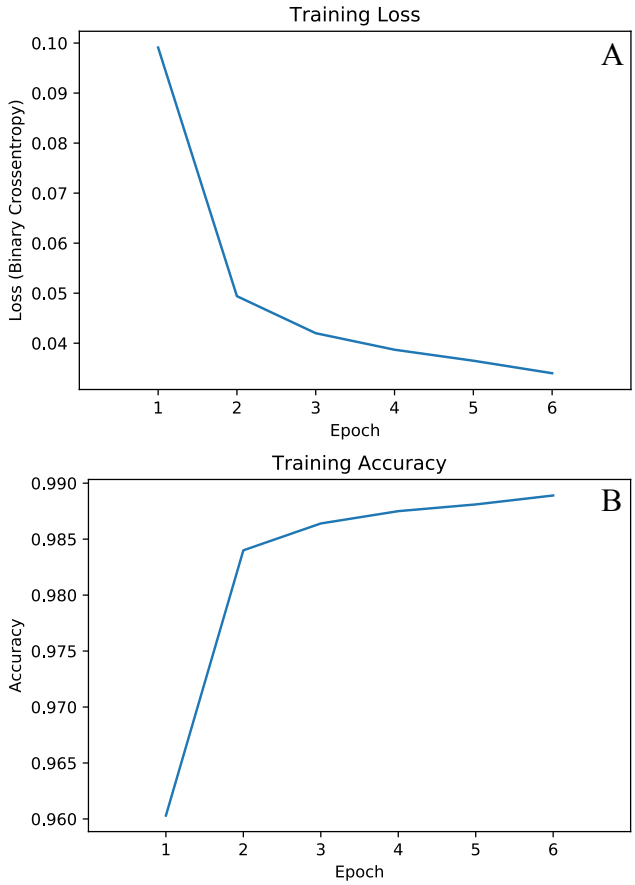

FIG. 2. DGAIntel training performance over six epochs. (A) Training loss for each epoch, determined by binary cross entropy. (B) Training classification accuracy for each epoch.

providing the exact same functions as the base package for DGA prediction.

Additionally, for cybersecurity professionals who want to utilize the model's inference capabilities without having to write code, we have the dgaintel.com website, which runs a Flask application serving the model's predictions. It also allows retrieves WHOIS data for the domain, if it exists. Note that despite the web interface providing WHOIS data, the DGAIntel model does not utilize any side information in generating its predictions.

III. RESULTS AND DISCUSSION

We implemented the DGAIntel model as described in the section above in Python using Google's Tensorflow deep learning framework. Training was conducted for 6 epochs over the aforementioned dataset. The network was trained with the Adam optimizer. Training ran for 45 minutes on a GPU-enabled Google Colaboratory environment. The training loss and accuracy is shown in Figure 2. The final validation accuracy of the model was 98.9%, averaged across the 100,000 validation examples.

The accuracy of the DGAIntel model over the test data, as well as that of other DGA classification models from [5], is shown in Table 1. Our model, thanks to improvements in regularization and architectural depth, is able to outperform other state-of-the-art models.

Model	Architecture	Accuracy
RF	Random Forest	91.51%
MLP	Multilayer Features	73.74%
Endgame	LSTM	98.72%
CMU	LSTM	98.54%
NYU	CNN	98.58%
Invincea	CNN	98.95%
MIT	LSTM-CNN	98.70%
DGAIntel (this work)	LSTM-CNN	98.94%

TABLE 1. Comparison of DGAIntel architecture and accuracy to other state-of-the-art DGA classification models from [5].

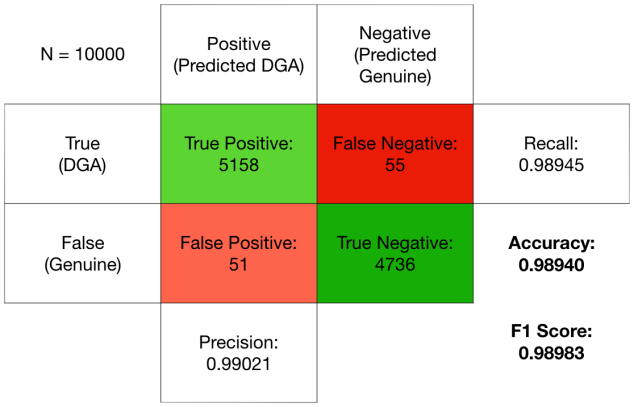

FIG. 3. DGAIntel test data confusion matrix. The horizontal axis shows examples where the model predicted a domain name was genuine or malicious, and the vertical axis shows examples where the domain name had a ground-truth rating of being genuine or malicious.

The confusion matrix of the model over the test dataset is depicted in Figure 3. This shows the high precision (99.0%) and high recall (98.9%) in addition to the accuracy of the model. Recall that precision refers to the ability of the model to reject false positives and that recall is the ability of the model to reject false negatives. Low false positive rates are very important for DGA detection as it prevents production systems from obstructing critical network traffic. Likewise, low false negative rates are also incredibly important, as it means the system will be effective in detecting malware.

Figure 4 shows the receiver-operating characteristic (ROC) curve and the precision-recall (PRC) curve of the model. The test set ROC-AUC score was 0.998, and the PRC-AUC score was 0.999. These demonstrate how DGAIntel is able to balance detecting true DGA domains with rejecting genuine domains.

DGAIntel's shallow implementation allows for incredibly fast predictions. The prediction time on a single domain is

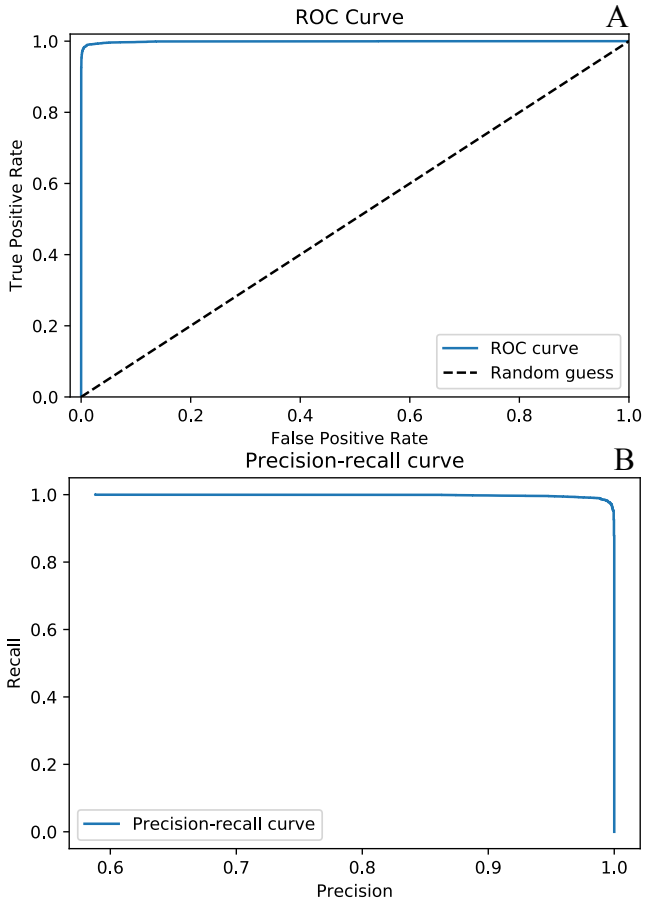

FIG. 4. Binary classification performance curves. (A) Receiver-operating characteristic curve (ROC). (B) Precision-recall curve (PRC).

only 286 milliseconds. Moreover, the Tensorflow model that powers DGAIntel supports input batching, so there is a significant improvement in speed in running predictions on batches of domains. Figure 5 shows the dependence of model prediction time on the prediction batch size, compared to the expected prediction time under an assumption of linear scaling. This shows how the model implementation allows for efficient scaling to large tasks.

IV. CONCLUSION

In summary, we have developed the DGAIntel engine, a lightweight implementation of a CNN-LSTM model to detect DGA domains without the use of contextual information. The model performance demonstrates that deep learning is a powerful tool for intelligent detection of malware. The model implementation also enables cybersecurity experts to integrate DGAIntel into threat intelligence analytics, enabling better network defense. The codebase for this project has been open-sourced to allow for public usage, and an intuitive website has been implemented to allow users to utilize intelligent DGA predictions in their work. Future work will focus on tuning

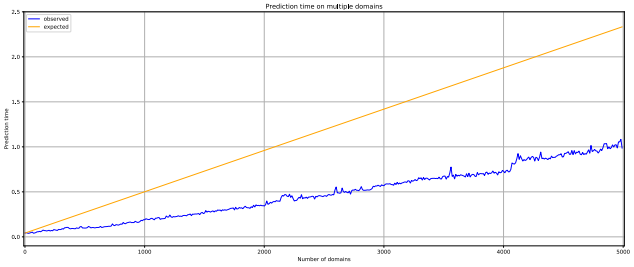

FIG. 5. DGAIntel prediction timing. The horizontal axis shows the number of input domains in a prediction batch. The vertical axis shows the prediction time. The blue line shows observed prediction times, whereas the orange line shows expected times under an assumption of linear runtime.

the model via adversarial training, optimizing the model for performance and speed.

ACKNOWLEDGEMENTS

The first author thanks their father for his domain expertise and inspiration. They thank Bridgewater Associates for the opportunity to present an early version of this research and get feedback on its real world usefulness. They finally thank their friends and family for their timeless support.

REFERENCES

[1] The Council of Economic Advisors, *The Cost of Malicious Cyber Activity to the U.S. Economy*. Washington, DC: Executive Office of the President of the United States, 2018.

[2] D. Plohmdann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A Comprehensive Measurement Study of Domain Generating Malware," in *Proceedings of the 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*

[3] M. Antonakakis, "From Throw Away Traffic to Bots: Detecting the Rise of DGA-Based Malware," *21st USENIX Security Symposium, Bellevue, WA, US, August 8-10, 2012.*

[4] M. Kührer, C. Rossow, and T. Holz, "Paint It Black: Evaluating the Effectiveness of Malware Blacklists," *Research in Attacks, Intrusions, and Defenses*, 8888, pp. 1-21, 2014.

[5] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character level based detection of DGA domain names," in Proc. WCCI, Rio de Janeiro, Brazil, 2018, pp. 4168–4175.

[6] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, "Detecting DGA domains with recurrent neural networks and side information," on *Proceedings of the 14th International Conference on Availability, Reliability, and Security, ARES '19, Canterbury, CA, UK, August 26-29, 2019.*

[7] B. Yu, J. Pan, D. Gray, J. Hu, C. Choudhary, A. Nascimento, and M. De Cock, "Weakly Supervised

Deep Learning for the Detection of Domain Generation Algorithms," in *IEEE Access*, vol. 7, pp. 51542-51556, 2019.

[**8**] "Alexa," 2019. [Online]. Available: https://www.alexa.com. [Accessed 2017-05-28].

[**9**] J. Bambenek, "OSINT DGA Feed", 2019. [Online]. Available: https://osint.bambenekconsulting.com/feeds/dga-feed.txt. [Accessed: November 5, 2019].

[**10**] "DGA Analysis," *Splunk Apps.* [Online]. Available: https://splunkbase.splunk.com/app/3559/. [Accessed: November 5, 2019].

[**11**] S. Vosoughi, P. Vijayaraghavan, and D. Roy, "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, 2016, pp. 1041–1044.

[**12**] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting Domain Generation Algorithms with Long Short-Term Memory Networks." [Online]. Available: arXiv, http://arxiv.org. [Accessed November 22, 2019].

[**13**] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

APPENDIX

1. Code access

The source code for DGAIntel can be found on GitHub (https://github.com/sudo-rushil/dgaintel). The packaged code can be found on PyPI (https://pypi.org/project/dgaintel/). Documentation for the DGAIntel API is available on GitHub (https://github.com/sudo-rushil/dgaintel/master/README.md). All code is released under an MIT license.