Detection of Automatically Generated Domain Names in Real-time with Machine Learning

James Pepper, PhD and Drew Gidwani
ThreatConnect, Inc.
Arlington, VA, USA

Sophisticated adversaries can make malware difficult to detect and mitigate by dynamically generating pseudorandom network configuration parameters, such as the domain name of a command-and-control server. Historically, security practitioners could only keep up with this technique if they were able to reverse engineer the algorithm from a collected sample and monitor a large probability space of possible domains that could number in the hundreds of thousands. This analysis evaluates two popular machine learning algorithms' ability to detect these randomized domain names. A decision tree model and a long short-term memory (LSTM) model were trained using more than 2.5M legitimate domains and over 35M algorithmically generated domains from the aggregated data maintained in ThreatConnect's Collective Analytics Layer™. Although both models had some success, the LSTM performed the best with 99.2% precision and 97.2% recall. The high precision and low latency of the model allows it to be used for real time detection of these types of malware domains without additional WHOIS information or a priori knowledge of the randomization algorithm.

I. INTRODUCTION

As computer security practitioners have evolved their detection and mitigation capabilities, so have malicious actors who wish to establish and maintain a presence on victim networks. A key component of the lifecycle of a cybersecurity attack is to receive command and control (C&C) options from attacker-controlled infrastructure. These C&C options can include something as simple as receiving configuration parameters or specific tasking, as well as more complicated operations such as the exfiltration of victim data or the downloading of additional malware to enable further malicious activity. As with the other various stages of an attack, this inflection point has been a focus for security professionals as much as it has been for the attackers.

One of the developments in avoiding C&C detection and mitigation is the ability for malware to dynamically generate C&C configuration details at a configured interval using a pre-programmed algorithm. This makes the configured C&C server a constantly moving target for security practitioners to identify and target with appropriate countermeasures. Given that most detection and mitigation techniques require a unique element, such as an internet protocol (IP) address or domain name, any malware that dynamically changes its C&C configuration poses a significant challenge to network defenders. Any mitigations against such malware may be rendered moot when the underlying algorithm dynamically changes the mitigated parameters.

One such configuration change leveraged by this technique is to periodically change the domain that the malware uses to communicate with its C&C server via the domain name service (DNS). These underlying algorithms often rely on an initial seed, and are referred to in the industry as Domain Generation Algorithms (DGAs). Likewise, the domains created by these algorithms are known as Automatically Generated Domains (AGDs). Some examples of AGDs, obtained from BambenekLab's open-source list[6], is shown side-by-side with domains from the Alexa's list of the top 1 million most popular domains in Table I. Note that to ensure the attacker can actually register the domain name, the output of the DGA must be sufficiently unique. This, combined with their algorithmic generation, often yields a "pseudo-random" appearance to security practitioners despite being deterministically generated.

Historically, proactive detection of AGD's has required that security practitioners procure a sample of the malware itself, reverse engineer its algorithm, and brute force the known algorithm with all possible input seeds. This yields a verbose but comprehensive list of all possible output AGDs, which can then be used to inform detection or mitigation efforts. While ensuring a high degree of recall, this approach often suffers in precision: for common algorithms the possible output space may have hundreds of thousands of domains, but only a dozen may actually be relevant. Without knowing the seed, the size of this data is unwieldy at scale in modern security solutions and requires a priori knowledge of the algorithm itself.

To overcome these difficulties in detection, there have been many studies[1–5] on the usefulness of machine learning to detect these types of domains. Such models could potentially catch AGDs in near real time, rather than waiting for them to be published by open-source and proprietary lists, and before they have a significant amount of network traffic associated with them. This study will analyze two particular approaches at opposite ends of the machine learning spectrum, and evaluate their effectiveness at detecting AGDs, as well as their rate and type of false positives using a real world dataset obtained from various sources and aggregated by ThreatConnect Inc. We will show that these methods can be employed effectively in the real time detection of AGDs, with high accuracy and precision and minimal false positives.

Bambenek AGDs	Alexa Domains
qyqrrct.pw	networkadvertising.org
rgkbeljryx.com	newyorker.com
rjtjwocnokqjwksflb.com	allaboutcookies.org
roandkowqharndqfmxkpp.pro	soyadmin.com
sqywrirk.pro	zend.com
ssedqckvtbdqoxxyo.pro	marketwatch.com
ttddxiniqcklxkyjfctsr.pw	constantcontact.com
ulgavtusysq.pw	fastcompany.com

TABLE I. Example AGDs from Bambenek side-by-side with Alexa domains for comparison. Although humans can easily distinguish these lists from one another, careful analysis is required to automate the process.

II. METHODOLOGY

A. Data

1. Origin/Content

ThreatConnect's Collective Analytics Layer (CAL) contains hundreds of millions of potential indicators of compromise (IoCs) to support its analytics that provide reputation and contextual insights to security practitioners. This includes a series of data feeds providing malicious indicators, as well as a series of data feeds that provides relevant metadata to inform CAL analytics. These datasets were used as described below to generate a labeled dataset of high-confidence AGDs and high-confidence legitimate domains.

To generate the list of high-confidence AGDs, the entirety of the available Bambenek Consulting dataset was used. As discussed in the Introduction, this dataset is provided via the brute forcing of known seeds for already reverse-engineered algorithms.

To generate the list of high-confidence legitimate domains, a plurality of Top 1 Million domain lists in CAL, as determined by various data aggregators, were used. These data providers each leverage proprietary data and algorithms to determine the most popular websites on the internet. The size and source of the datasets used for each respective labeling are detailed in the following table.

Data Source	Label	Approx. Size
Bambenek DGA domains	AGD	35M
Total "legitimate" domains	Legitimate	2.5M
Alexa[7]	Legitimate	1M
Cisco Umbrella[8]	Legitimate	1M
Majestic[9]	Legitimate	1M
Quantcast[10]	Legitimate	1M

The combination of all top million lists amounts to around 2.5M unique domains due to overlap between the lists. Top-level domains (.com, .net, .cn, .ru, etc) were not considered in the analysis that follows due to the fact they are not random, and have the potential to introduce bias.

2. Statistical Analysis

The first step towards distinguishing between good domains and AGDs involves finding statistical differences between the two. Defining features, that provide a clear separation between the two types of domains, can be leveraged by humans or machine learning algorithms to assign a probability of a given domain belonging to either group.

Analysis of the frequency at which characters appear in legitimate domains compared to AGDs, immediately elucidates some differences that can be leveraged to distinguish between the two domain types. As expected, legitimate domains follow a similar distribution to the English language. This is due to the fact that the most desirable domains are easy to remember, and therefore generally contain complete English words.

For legitimate domains, vowels are by far the most frequently occurring, followed by the most common consonants (R, S, T, L, and N) in the English language. AGDs on the other hand, exhibit a mostly featureless distribution of characters, with ∼3-4% probability for any given alphabetic character, and significantly less (∼ 0.1%) for numerical characters as seen in Figure 1.

One obvious consequence of the flat character distribution for AGDs is the overabundance of consonants. Intuitively, one would not expect long strings of consecutive consonants in human created domains, since these would likely not be composed of English words, and thus be more difficult to remember. Plots of the number of consecutive consonants, as well as the longest chain of consecutive consonants in each domain are shown in Figure 2. The longest consecutive consonant chain in legitimate domains was 1.8 on average, while the AGDs averaged 4.8. Similary, legitimate domains averaged 2.7 instances of consecutive consonants, while AGDs averaged 12.1.

Another important difference between legitimate domains and AGDs is domain length. It serves to reason that domains in the top 1 million list would have more desirable domain names of shorter length. AGDs, on the other hand, don't need to be remembered by end users, and thus can be any arbitrary length. Generating longer domains also reduces the possibility of a "collision" with a pre-existing domain. On average, the legitimate domains considered in this analysis averaged 10.5 characters, whereas the AGDs averaged 18.5.

Kullbeck-Liebler divergence, or relative entropy, is a useful metric for comparing the distribution of characters in a domain name to that of legitimate domains (as detailed in [11]). If a domain diverges significantly from the legitimate domain character frequency distribution, it will have a higher entropy. The equation for Kullbeck-Liebler divergence is,

$$D_{KL}(P \parallel Q) = \sum_{i=1} p_i \log\left(\frac{p_i}{q_i}\right) \tag{1}$$

where p_i and q_i refer to the frequency of a given char-

Character Frequencies

	Alexa	Bambenek
a	0.0976	0.0323
e	0.0948	0.0340
i	0.0736	0.0324
o	0.0713	0.0355
r	0.0658	0.0347
s	0.0642	0.0340
t	0.0628	0.0336
n	0.0609	0.0348
l	0.0471	0.0336
c	0.0369	0.0320
m	0.0344	0.0337
d	0.0326	0.0323
u	0.0300	0.0327
p	0.0292	0.0350
h	0.0272	0.0306
g	0.0239	0.0301
b	0.0230	0.0312
k	0.0209	0.0325
y	0.0172	0.0304
f	0.0157	0.0332
v	0.0146	0.0316
-	0.0123	9.13e-6
w	0.0110	0.0310
z	0.0067	0.0199
x	0.0061	0.0304
j	0.0046	0.0312
2	0.0023	0.0140
1	0.0022	0.0412
q	0.0018	0.0344
0	0.0017	0.0126
4	0.0016	0.0137
3	0.0014	0.0138
8	0.0010	0.0134
7	0.0009	0.0134
5	0.0009	0.0138
6	0.0008	0.0137
9	0.0008	0.0134

FIG. 1. The frequencies at which characters appear in Bambenek and Alexa domains, ordered from most frequent to least with respect to Alexa. As expected, the probability distributions are very different. For Bambenek domains, all letters have similar probabilities, as do numbers. Alexa domains more closely follow the English language with vowels being the most frequent, and characters like ‘j’ and ‘q’ being the least frequently used.

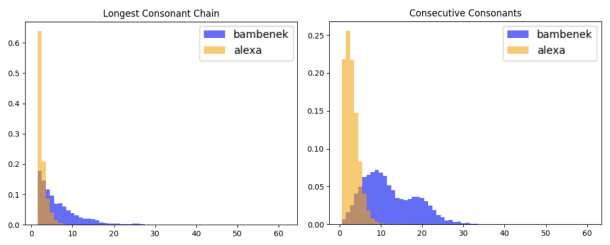

FIG. 2. AGDs are more likely to have consecutive consonants compared to legitimate domains, including long “chains” of consonants. The bimodal distribution in consecutive consonants is likely due to the naming conventions of prevalant malware families in the Bambenek data.

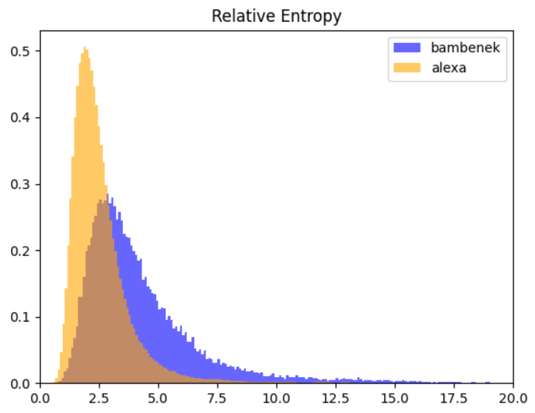

FIG. 3. The relative entropy of Bambenek AGDs with respect to Alexa domains from Equation 1. The majority of Alexa domains are below the peak entropy value for the Bambenek AGDs, meaning the distribution of characters in Bambenek domains differs significantly from those in Alexa.

acter in the domain name and legitimate domains respectively. The entropy for the Bambenek and Alexa domains included in this analysis are shown in Figure 3.

It is important to note, the features outlined in this section rely only on the domain name itself, and display separation power between legitimate domains and AGDs. While WHOIS data and other information about the domain might aid in classification of domains, our goal is to perform this classification on many domains in real time, and these methods would add complexity, thereby slowing down the classification process considerably. In the following section, we will use machine learning algorithms to exploit the separation power of these features to classify domains between AGD or legitimate.

B. Experimental Setup

Having found various metrics in which legitimate domains behave differently from AGDs, it makes sense to use these metrics in conjunction with supervised machine learning algorithms to train on AGD classification. For this experiment, two different classification algorithms were chosen at opposite ends of the model complexity spectrum, to compare their accuracy, precision, recall, and training time. Decision trees (DTs) train quickly, and their internal logic is straightforward to examine. However, they are highly dependent on feature engineering. Conversely, Long short-term memory (LSTM) models are “featureless,” but require a considerable amount of time to train, and their internal logic is not as easily interpreted.

	DT	LSTM
Accuracy	93.4%	98.2%
Precision	95.3%	99.2%
Recall	91.3%	97.2%
AGDs	gnfvghvjnbvhku	1i3u3c2139sbhz8hq4007uumhv
	8cwfr-exnr22ii1-r3	xcpjm1ueq1u3nh6z3q9z0vas
	xnd60q69c35it8d5x6broaj75m	fd5gsffjvi3599bkr3l546jv
	6ns-8k-5wkr8gm8a6	di5b5grycayh80cxdskov80k
	xndihq2fy6rpthcxobx0bsu7a	dyh7fhps5h0prsyd839oxov2
	t4dk57gpfbqibfk	sdhfshdfhsdfghsdfghdfghff
Example FPs	zhongeryen	rubbinbutzbbq
	thebswedding2020	lgbtqrunning

TABLE II. AGDs as determined by each model, along with examples of false positives, which highlight the limits of each approach. The decision tree is biased against domains with a low ratio of vowels, as well as non-English domains. The LSTM struggles with acronyms, in particular those involving the letter Q, as well as intentional misspellings, due to these types of domains being especially uncommon in the top ranking lists.

1. Decision Tree model

Decision trees choose a specific feature and value at which to divide the data into two sets. Each of the sets and corresponding features are then re-evaluated and split into another two nodes, and so on recursively, creating a binary tree. These splitting decisions occur at a specific value of a given feature so as to maximize information gain. This action continues until some threshold is met, such as on the degree of purity in the leaf nodes.

Decision trees have the advantage of being computationally easy to train. In addition, the decisions at each node are transparent, thus the model logic is easy to understand. However, DTs can only consider one variable at a time for a given decision. This is in stark contrast to more sophisticated neural networks whose logic can be difficult to decipher, and require more computational power to train. Features similar to those mentioned in Section II A 2, were calculated for every domain in the data, and these features were given to an sklearn-based DT for training.

2. LSTM model

LSTM models are a type of recurrent neural network (RNN) where each node's output is passed as input to the following node. These models therefore generally perform well at learning sequences. In contrast to a decision tree, which relies heavily on well researched input features, LSTMs are "featureless" in the sense that they learn features of the dataset on their own.

To input a domain, each character in the domain is mapped to a set of integers, and these integers are passed to the LSTM. In this way, the LSTM learns which sequences are top one million-like vs DGA-like. The downside to these models is, without the bonus of user-defined features, they take considerably longer to train. The LSTM in this experiment was created using Keras with a TensorFlow backend.

III. RESULTS

The DT and LSTM models were both trained on a g4.2xlarge EC2 instance in Amazon Web Services. The DT training completed in half an hour, while the LSTM required eleven hours to achieve the results discussed in this analysis. The accuracy, precision, and recall of each model are shown in Table II. The LSTM, as expected, significantly outperformed the DT in all areas. Examples of domains classified as AGDs, as well as false positives are also shown in Table II.

The DT leans towards AGD for domains that are very long, have a significant proportion of numbers, or use a language other than English. The latter two cause a high entropy relative to the top one million lists, and as a result, a fair amount of these false positives get a high confidence score for being an AGD.

The LSTM model, however, is not constrained to such rigid features, and considers the "flow" of the domain characters. For these reasons, it is able to make a nuanced decision with regards to domains with numbers at the end, or longer domain names, so long as they appear to have a legitimate pattern of consonants, vowels, and numbers. However, due to the popularity of English words in the top one million rankings, there is still a bias against non-English domains. The model can also be fooled by acronyms that don't commonly occur in top ranking websites, specifically those that use less common letters (BBQ, LGBTQ, etc). In spite of these limitations, false positives were rare and the LSTM model accurately classifies the vast majority of domains with very few false positives at the highest confidence levels.

LSTM DGA Probability	Percentage of Domains
>50%	99.9%
>75%	99.81%
>90%	99.66%
>99%	98.72%

TABLE III. Results of scoring 128,000 domains generated using the Zloader DGA. The left column shows the model probability that a given domain was generated by a DGA, and the right column shows what percentage of Zloader domains scored above this probability. Roughly 99% of the Zloader domains were labeled as having been generated by a DGA with 99% probability.

IV. ZLOADER VALIDATION

During the validation phase of this research project, malware from 2016, known as Zloader (aka Zeus Sphinx), experienced a resurgence in popularity. The InfoSec community was able to reverse engineer the DGA, and Johannes Bader published a list[12] of precalculated domain names for three different seeds.

A sample of 128,000 domains generated by the Zloader DGA were scored by the LSTM model, the results of which are shown in Table III. The results of this test demonstrate high accuracy and precision, as roughly 99% of the domains were classified as a DGA with a probability of 99% or higher.

V. CONCLUSION

As malware continues to evolve, so to must the methods for detecting and mitigating it. Currently, malware can avoid firewall blacklists by constantly generating new random domain names as a function of time or other inputs that yield pseudorandom C&C configuration information. In this experiment, we looked at two possible machine learning options for automatic detection of AGDs. Both methods were able to detect these domains with an accuracy above 90%.

Unfortunately, due to the popularity of english language websites, the models also contain such a bias. Features of the dataset used for prediction may be skewed by the input data, thus the models may incorrectly label legitimate domains. This is especially noticeable in character sequences that while legitimate, do not appear in the training set. Sequences such as “llc”, “bbq”, and “lgbt” have legitimate uses but were not common enough to earn a spot in the Top 1 Million sites used to train the model. This bias unfairly convicts any such pattern that doesn't exist in the upper echelon of popular websites. Improvements could be made with a more comprehensive training set of legitimate domains in various languages and of varying popularities.

There is also room for improvement with this approach to look at additional families of DGAs. Some DGAs may use a character frequency distribution that is more in line with legitimate English domain names. Additionally, some AGDs are created by chaining together legitimate English words. Based on the source data, the models discussed in this paper would struggle to correctly label those families of AGDs. While each model has its own strengths and weaknesses, they are both inherently limited by the variety and scale of source data used to train them.

Of the two models discussed in this paper, decision trees train quickly, and their internal logic is completely transparent. However, they require insight from researchers to generate features of the training data to help distinguish between legitimate and algorithmically generated domains. These features take time to discover and computational power to calculate. Even with a high degree of subject matter expertise, these features can lead to issues in predictive performance as they may be tainted by researcher bias or limitations in feature engineering. This will generally result in an unacceptable number of false positives.

The LSTM model, on the other hand, requires no feature engineering, minimal preprocessing on the original domain name, and is able to learn features of the sequence of characters. This is more precise than the user-defined features in a decision tree. Training time takes significantly longer for the LSTM compared to the DT, but this is a one time cost. One drawback to these neural networks is the internal logic is fairly opaque, and this can make understanding false positives more difficult. However, the overall encoding and prediction time is faster, more accurate, and more precise for the LSTM making it extremely useful in real-time detection of AGDs.

In conclusion, when provided with appropriately labeled data of sufficient scale, the LSTM model can serve as a valuable tool in aiding security practitioners in detecting AGDs. The nature of these models will always leave room for improvement, and future efforts in retraining can yield positive gains as more data becomes available across a variety of DGA families.

In accordance with in-depth defense practices, detection of pseudorandom domains can be enhanced with the appropriate addition of other heuristics and metadata. Even without these additions, this research shows that a properly-trained LSTM enables a new application of real-time pseudorandom domain detections that can outperform existing mitigation techniques with substantially less resources and analysis.

[1] S. Akarsh, S. Sriram, P. Poornachandran, V. K. Menon and K. P. Soman, "Deep Learning Framework for Domain Generation Algorithms Prediction Using Long Short-term Memory," *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, Coimbatore, India, 2019, pp. 666-671, doi: 10.1109/ICACCS.2019.8728544.

[2] Chin T., Xiong K., Hu C., Li Y. (2018) A Machine Learning Framework for Studying Domain Generation Algorithm (DGA)-Based Malware. In: Beyah R., Chang B., Li Y., Zhu S. (eds) Security and Privacy in Communication Networks. SecureComm 2018. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 254. Springer, Cham.
`https://doi.org/10.1007/978-3-030-01701-9_24`

[3] M. Mowbray and J. Hagen, "Finding Domain-Generation Algorithms by Looking at Length Distribution," *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, Naples, 2014, pp. 395-400, doi: 10.1109/ISSREW.2014.20.

[4] B. Yu et al., "Weakly Supervised Deep Learning for the Detection of Domain Generation Algorithms," in *IEEE Access*, vol. 7, pp. 51542-51556, 2019, doi: 10.1109/ACCESS.2019.2911522.

[5] Shuaiji Li, Tao Huang, Zhiwei Qin, Fanfang Zhang, and Yinhong Chang. 2019. Domain Generation Algorithms detection through deep neural network and ensemble. In Companion Proceedings of The 2019 World Wide Web Conference (WWW '19). Association for Computing Machinery, New York, NY, USA, 189–196. DOI:https://doi.org/10.1145/3308558.3316498

[6] Bambenek Consulting | Threat Intelligence and Digital Investigation Services,
`http://osint.bambenekconsulting.com/feeds/`

[7] Alexa - Top Sites
`https://www.alexa.com/topsites`

[8] Cisco Popularity List
`http://s3-us-west-1.amazonaws.com/umbrella-static/index.htm`

[9] Majestic Million - Majestic
`https://majestic.com/reports/majestic-million`

[10] Quantcast: AI-driven Audience Insights, Targeting & Measurement
`https://www.quantcast.com`

[11] Downing, Ben. Using Entropy in Threat Hunting: a Mathematical Search for the Unknown.
`https://redcanary.com/blog/threat-hunting-entropy/`

[12] The DGA of Zloader
`https://johannesbader.ch/blog/the-dga-of-zloader/`