Detection of Algorithmically Generated Domain Names used by Botnets: A Dual Arms Race.

Jan Spooren
imec - Distrinet - KU Leuven
Heverlee, Belgium
jan.spooren@cs.kuleuven.be

Davy Preuveneers
imec - Distrinet - KU Leuven
Heverlee, Belgium
davy.preuveneers@cs.kuleuven.be

Lieven Desmet
imec - Distrinet - KU Leuven
Heverlee, Belgium
lieven.desmet@cs.kuleuven.be

Peter Janssen
EURid VZW, Belgium
Diegem, Belgium
Peter.Janssen@eurid.eu

Wouter Joosen
imec - Distrinet - KU Leuven
Heverlee, Belgium
wouter.joosen@cs.kuleuven.be

ABSTRACT

Malware typically uses *Domain Generation Algorithms* (DGAs) as a mechanism to contact their *Command and Control* server. In recent years, different approaches to automatically detect generated domain names have been proposed, based on machine learning. The first problem that we address is the difficulty to systematically compare these DGA detection algorithms due to the lack of an independent benchmark. The second problem that we investigate is the difficulty for an adversary to circumvent these classifiers when the machine learning models backing these DGA-detectors are known. In this paper we compare two different approaches on the same set of DGAs: classical machine learning using manually engineered features and a 'deep learning' recurrent neural network. We show that the deep learning approach performs consistently better on all of the tested DGAs, with an average classification accuracy of 98.7% versus 93.8% for the manually engineered features. We also show that one of the dangers of manual feature engineering is that DGAs can adapt their strategy, based on knowledge of the features used to detect them. To demonstrate this, we use the knowledge of the used feature set to design a new DGA which makes the random forest classifier powerless with a classification accuracy of 59.9%. The deep learning classifier is also (albeit less) affected, reducing its accuracy to 85.5%.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; • **Computing methodologies → Neural networks**; **Classification and regression trees**;

KEYWORDS

Malware Detection, Domain Generation Algorithms, Machine Learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SAC '19, April 8–12, 2019, Limassol, Cyprus
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-5933-7/19/04…$15.00
https://doi.org/10.1145/3297280.3297467

ACM Reference Format:
Jan Spooren, Davy Preuveneers, Lieven Desmet, Peter Janssen, and Wouter Joosen. 2019. Detection of Algorithmically Generated Domain Names used by Botnets: A Dual Arms Race.. In *The 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), April 8–12, 2019, Limassol, Cyprus.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3297280.3297467

1 INTRODUCTION

The Internet connects billions of devices, ranging from servers and personal computers to tablets, mobile phones, household appliances, and many more. Malicious actors are constantly scanning the internet for vulnerable devices which could be compromized, or are tricking users into unknowingly installing malware on their devices. Once this malware is present on a machine, it can be used to attack other machines, send unsolicited or phishing e-mails, eavesdrop on communication, steal e-mail addresses, encrypt the contents of the machine requesting from the user a ransom for the ability to decrypt, and many more malicious schemes. Large pools [17] of infected machines, called botnets [4] exist, which are controlled from *Command and Control* (C&C) servers (as depicted in Figure 1).

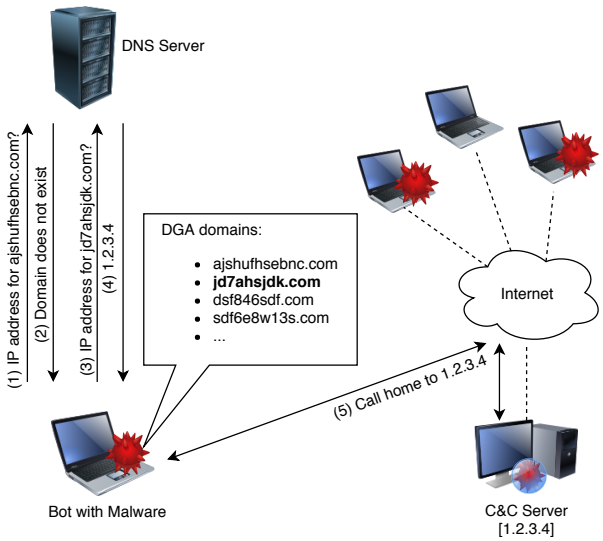

Figure 1: Bot using DGA to connect to a C&C Server

To prevent those C&C servers from being shut down or made unreachable, malware typically uses *Domain Generation Algorithms* (DGAs) [15] in order to create a new set of pseudo-random domain names every certain period of time. Many random domains are generated and then tried by the malware, only one of which the botnet owner must register in order for the botnet to be able to successfully reconnect to its C&C server. This makes taking down botnets a difficult task.

Conversely, recognizing algorithmically generated domain names can help in detecting infected hosts, as well as flag domain name registrations made with the intent to control a botnet. Machine learning classifiers that can successfully distinguish an algorithmically generated domain name from a human-created domain name are therefore useful tools to security researchers, law enforcement and network operations staff.

In the domain of security and privacy, deep neural networks have demonstrated their ability to autonomously find and extract relevant features as well as improved classification accuracies when compared to traditional machine learning methods [5, 11, 16]. Also for the detection of DGAs, there is recent related work proposing solutions based on both traditional machine learning and deep learning methods. However, malicious adversaries can exploit these AI classification methods too to evade detection of their malware. These observations reflect the dual arms race. The first race relates to further improving advanced deep learning methods to extract better features for offensive or defensive purposes. The second one is the cat and mouse game of adversaries trying to circumvent new defensive measures developed by security specialists, irrespective of the type of machine learning methods being used.

The first challenge that we address in this work is that existing detection methods are difficult to compare. Not only do they rely on different data sets and ground truths to assess their classification accuracy, assumptions made by some of these related works regarding the ground truth of malicious versus benign domain names may not always be valid. Both concerns make it difficult to interpret and compare the accuracy of DGA detection methods. The second challenge that we focus on in this work is that adversaries become smarter too in circumventing these DGA detection methods.

The main contributions of this work are as follows:

(1) We independently benchmark and compare two state-of-the-art DGA detection methods: (a) FANCI [19], a state-of-the-art Random Forest based classifier recently presented at USENIX Security 2018 using manually crafted features, and (b) Woodbridge's LSTM [24], a deep recurrent neural network based classifier.
(2) We illustrate the arms race using FANCI as a detection method and designing a new DGA that exploits knowledge about FANCI's feature extraction to circumvent detection. We also evaluate how well our new DGA is able to fool Woodbridge's LSTM for which the inherent features that characterize a DGA are more implicit.

In section 2, we briefly summarize relevant related work, elaborating on an approach using Random Forests with manually crafted features. In section 3, we explain the used methodology, the collection of the ground truth data (section 3.1) and the compared classifiers (section 3.2). We present the results of this comparison in section 4. In section 5, we show that knowledge of the used feature-set can be abused by DGAs to circumvent detection, by building a DGA which achieves exactly this. Finally, in section 6, we state our main conclusions.

2 RELATED WORK

Many different approaches were taken to detect DGA-generated domain names: Yadav et al. [25] employed DNS queries and responses for detecting malware. They used the entropy in the observed unigram and bigram statistics on the character sequences present in domain names to distinguish (often difficult to pronounce) random character strings from human-created (and usually better pronounceable) domain names. Antonakakis et al. [3] use a clustering approach on the length, level of randomness and character frequency distribution, including the n-gram distribution (with n = 1,..,4) of observed domain names that have been queried by overlapping sets of hosts. In a second phase, a *Hidden Markov Model* (HMM) is used for determining the likelihood that a domain name is a C&C domain generated by a particular DGA.

Mowbray and Hagen [12] described a procedure to detect even previously unseen DGAs from DNS query data, by spotting an unusual number of DNS requests for an unusual distribution in domain name length. While their method proved successful, it is doubtful if using domain name length only as a feature for detection will suffice when DGAs become more advanced.

Schiavoni et al. [18] created a system called Phoenix for detecting DGA-based botnets, which uses filtering using linguistic features (detecting meaningful subwords, n-gram normality scores and statistical linguistic features for the English language), followed by a clustering step and finally, a fingerprinting step to extract invariant properties of a DGA and subsequently label the collected domain names.

In an excellent paper by Woodbridge et al. [24], *Long Short-Term Memory* (LSTM) networks [8] are proposed to detect algorithmically-generated domain names. Woodbridge and his team at EndGame, Inc. show that LSTMs are far superior to Random Forests with manually engineered features, logistic regression with bigram features and HMM classifiers.

Pereira et al. [13] proposed a novel *WordGraph* method for detection of dictionary-based DGAs using graph-theory. They showed that random forests using lexical features perform badly on dictionary-based DGAs, likely due to the good resemblance of the features to manually chosen domain names. Their method outperforms convolutional neural nets for dictionary-based DGAs, which they have found to work well too on these DGAs, presumably due to the neural net's ability to learn interesting soft n-grams, which, in a sense, allows them to memorize dictionaries, too.

Recently, a paper by Schüppen et al. [19] presented FANCI at USENIX Security 2018, carrying no reference to any work using deep learning, in which manually engineered features were used, in combination with Random Forests to detect DGA-generated domains in NXDomain-failed DNS queries (i.e., queries to non-existent domain names).

In this work, we have re-implemented and re-evaluated two state-of-the-art DGA detection methods, one using deep learning, the other using traditional machine learning, i.e. respectively the

LSTM method presented in [24] and the FANCI Random Forest classifier presented in [19], and compared the performance of both approaches on the same data set. Such a systematic comparison is necessary as (1) different works rely on dissimilar methods to obtain the ground truth used to train the machine learning classifiers, and (2) previous research has shown that some works make invalid assumptions about the truthfulness of their ground truth data. Both solutions will be tested against the newly designed DGA.

3 METHODOLOGY

To compare the classification performance of two classifiers, we first need to obtain a 'ground truth' data set: in this case, a collection of domain names which are known with certainty to be algorithmically generated or not.

3.1 Ground truth

Ground truth data for both *malicious* (DGA) and *benign* (non-DGA) domain names is by the very nature of this research harvested from different sources:

Malicious ground truth was obtained from DGArchive [14], a service offered by Fraunhofer FKIE and maintained by Daniel Plohmann. Since only a limited number of top level domains (TLD) are available and the particular TLD used by a domain generation algorithm does not have any influence on classification accuracy, we removed the TLD from the DGA domain names. We selected all DGAs in DGArchive for which 100,000 or more unique recorded domain names were available when disregarding the TLD from the domain name. This resulted in a list of 26 DGAs shown in Table 1.

For **benign ground truth**, many authors choose the Alexa [1] top-*n* lists of most accessed web sites. However, recent work by Le Pochat et al. [10] has shown the Alexa top-*n* lists to be easily manipulatable. In fact, it turns out the Alexa lists already contain DGA-generated domain names. Moreover, the list of most popular domain names is very likely biased towards shorter, easier to pronounce and to remember domain names, compared to the average registered domain name. Therefore, the authors of this work have chosen to use the list of the first 100,000 registered domain names from a well known TLD in 2016, which were filtered according to the following criteria:

- The domain names did not end up on any of the following black lists: *Google's Safe Browsing List* [7], *Spamhaus DBL blacklist* [21], the *SURBL blacklist* [20].
- No internationalized domain names were used (since at the time of writing none of the DGAs use internationalized (IDN) domain names – which would make classification of internationalized domain names trivial)
- The domain names were not known in DGArchive.

For each DGA, we constructed the ground truth data as a shuffled mix of 100,000 benign and 100,000 malicious domain names, using the same benign ground truth data for each DGA evaluated.

3.2 DGA Classifiers

In this section, we briefly describe the main characteristics of the two state-of-the-art DGA classifiers we used for our experiments.

Table 1: Domain Generation Algorithms Evaluated

DGA	Length		Comments
banjori_dga	7 → 26	†	banking trojan
chinad_dga	16		
conficker_dga	5 → 11	†	worm
cryptolocker_dga	12 → 18	†	ransomware
dnschanger_dga	10		
dyre_dga	34		banking malware
emotet_dga	16	†	information stealer
gameover_dga	19 → 28		banking, info stealer
gameover_p2p	11 → 32	†	botnet Zeus
gozi_dga	12 → 24	†	information stealer
locky_dga	5→ 17		ransomware
murofet_dga	7 → 16	†	
murofetweekly_dga	32 → 47		
necurs_dga	7 → 25		
nymaim_dga	5 → 13		infrastr. + ransomware
padcrypt_dga	16 → 19		ransomware
proslikefan_dga	6 → 12		
pushdo_dga	8 → 12		
pykspa_dga	6 → 12	†	
qadars_dga	12		
qakbot_dga	8 → 25		
ramnit_dga	8 → 19		file infector
ranbyus_dga	14 → 17		
rovnix_dga	18		
symmi_dga	13 → 20		
virut_dga	6		file infector

†: Malware not active anymore in 2018.

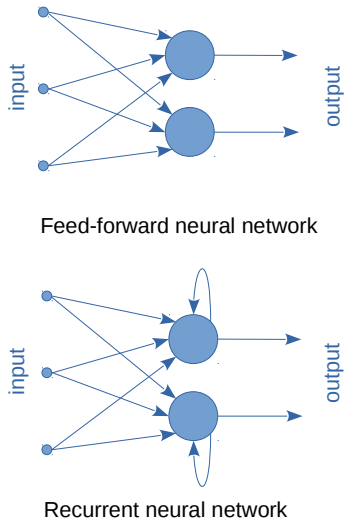

Figure 2: Feed-forward vs. recurrent neural networks

3.2.1 The Woodbridge LSTM. We implemented the Long Short-Term Memory (LSTM) neural network as proposed in the aforementioned work by Woodbridge et al. [24].

An LSTM is a specific type of recurrent neural network (RNN). RNNs are frequently used for recognizing or predicting patterns in sequential data. Contrary to feed forward neural networks, RNNs have an internal short-term memory to persist important things about the input they receive. They achieve this by copying the output and looping it back into the network, as depicted in Figure 2. This characteristic enables RNNs to create a deeper understanding of a sequence and its context, and to predict what is coming next. LSTMs extend RNNs by enabling such networks to remember their inputs over a longer period of time, thereby extending their memory capacity beyond two time steps. A cell in an LSTM has a state that can be read, written or reset via a set of programmable gates. These gates modulate both the input connection and recurrent connection with a value between 0 and 1 allowing the current state to remain the same between time steps, to be forgotten in the next time step or a combination thereof.

LSTMs are a good fit to recognize DGAs as they can learn and generalize the pattern generation process of many DGAs without the need to manually craft higher level features based on the raw input. The authors also argue that LSTMs work as a black box making it difficult for adversaries to circumvent a classifier without the same training set. The DGA classifier proposed by Woodbridge et al. consists of following sequential layers:

- An **embedding layer**, which converts variable length sequences of domain name characters to a fixed length, zero-padded array of features.
- The **LSTM layer**, which receives its input from the embedding layer with a dimensionality of 38 (encoding 26 characters, 10 digits, the dash character and an end token), and generates an output dimension of 128.
- A **dropout layer** of 0.5, preventing over-fitting.
- A dense **output layer**, with one output dimension followed by a sigmoid activation function.

To ensure the validity of our classification results, we re-used the implementation available on the EndGame Inc. GitHub repository [9].

We evaluate the performance of the classifier for each DGA separately, using *5-fold cross validation*: the network is trained with 4/5 of the data in 10 epochs, using a batch size of 128; the remaining 1/5 of the data is then used for testing the trained network. This is repeated 4 more times with different folds of the data, discarding the previously trained network, thus ensuring testing data was never used for training.

3.2.2 Random Forests and FANCI features. We implemented the FANCI feature extraction as described in the aforementioned work by Schüppen et al. [19] and fed the 41 extracted features[1] to a Random Forest of 100 trees, each constructed considering 6 random features. We again used 5-fold cross validation to evaluate its performance.

In this comparative work, we wish to compare the efficacy of the deep-learning LSTM network, which will find its own features from the input data, versus the manually engineered features from the FANCI system in detecting algorithmically generated domain names, purely based on the domain name character sequences. The FANCI system not only detects DGA generated domain names by examining domain name character sequences, but also looks at other features, harvested from NXDomain DNS queries. E.g., FANCI feature 5, *Has Valid TLD* is a binary feature, indicating that the domain name has a valid top level domain. It is clear that no DGA will ever output domain names with an invalid TLD, since those domain names would never resolve and are therefore useless. This characteristic is a useful feature in the FANCI system, since it can easily detect human typing errors in the TLD-part of domain names, which are therefore clearly never DGA-generated domain names. Our methodology to fairly compare the two classifiers on their ability to detect an algorithmically generated domain name, consists of feeding both with a mix of 50% generated domain names by a known DGA and 50% benign ground truth data, both of which carry no TLD nor subdomains. Therefore, a number of features used by the FANCI system will in practice not be used in our comparative test. These features are listed in Table 2. It is therefore to be expected that the classification scores for FANCI as listed in this work will be lower than those published in the FANCI paper [19].

[1]FANCI uses 21 features, however, feature #20 is a feature vector of 21 values, thereby resulting in 41 values

Table 2: FANCI features that are not used in the comparison

No	Feature	Output
2	Number of Subdomains	int
3	Subdomain Length Mean	real
4	Has www Prefix	bool
5	Has Valid TLD	bool
6	Contains Single-Character Subdomain	bool
8	Contains TLD as Subdomain	bool
9	Ratio of Digit-Exclusive Subdomains	real
10	Ratio of Hexadecimal-Exclusive Subdomains	real

4 DGA CLASSIFICATION RESULTS

The classification results for each of the tested DGAs are listed for both classifiers in Table 3. For all of the tested DGAs, the LSTM-classifier yielded a better or identical *True Positive Rate* (TPR), *False Positive Rate* (FPR), *Precision* and *Accuracy*. On average, the FPR of the Random Forest classifier is over 4 times higher than the LSTM. The average accuracy of the LSTM-classifier is 98.7%, while the average Random Forest classifier reaches 93.7%.

The standard deviation on the FPR is 3 times higher for the Random Forest classifier, and the standard deviation for the accuracy is 4 times higher for the Random Forest classifier, indicating that the LSTM not only yields better classification results, but also a higher consistency over different DGAs. Notably, the Random Forest with the FANCI features perform considerably worse on the `gozi_dga`, `locky_dga`, `nymaim_dga`, `pushdo_dga` and `pykspa_dga` DGAs.

One possible explanation for the better consistency of the LSTM classifier is that the LSTM network learns its used classification features automatically during the training process, whereas the Random Forest needs manually engineered features, which may be better or worse suited for different DGAs. There is, however, another danger in the use of manually engineered features: knowledge

of which features are used can be used to the advantage of DGA developers, who can modify the Domain Generation Algorithm to become less easily detectable.

In the following section, we will do exactly this: we will construct a new DGA, with the knowledge of the FANCI feature set to make it less easily detectable.

5 ATTACKING THE MANUALLY CRAFTED FEATURESET

The goal of this exercise is to demonstrate that it is possible to create a new DGA, which we will name `deception_dga`, which takes the features of a classifier into account in order to circumvent detection. An adversary creating such a DGA will need to implement the feature set used by the classifier, obtain a classifier implementation, and obtain the benign ground truth.

We used the following approach:

- We implemented the FANCI feature set of 41 features as defined in the FANCI paper [19] in a Python implementation. However, as mentioned in section 3.2.2, the features listed in Table 2 are unused in our comparison, the goal of which was to compare two different machine learning techniques objectively on their ability to detect algorithmically generated domain names, purely based on domain name character sequences. Therefore, in the following, we will examine the effect of influencing a certain number of the 33 features actually used.
- We used the WEKA[23] workbench and its implementation of Random Forests to test the performance of our newly built DGA.
- An adversary creating a DGA will require benign ground truth data for testing too: we assume an adversary would probably use the publicly available Alexa [1] list. We used the top 1 million sites of September 16th 2018, and used 100,000 unique domain names (excluding the TLD), starting with domain 100,001 (`senenews`) and ending with domain 203,824 (`teatrebarcelona`)[2].
- We generate 100,000 domain names with our DGA with a given seed as the malicious ground truth, add the 100,000 Alexa-domains as the benign ground truth, shuffle these records and perform classification on this data set with 5-fold cross validation.

With the results of the classification, adversaries can iteratively improve their DGAs (as illustrated in Figure 3), until a desired sub-optimal classification result is achieved, therefore effectively thwarting DGA-detection classifiers. While creating our `deception_dga` DGA, we only used data from the Alexa list, mimicking the information available to a DGA author. However, for fairness of comparison, we evaluated the classifier performance of the Random Forest using the FANCI feature set and the Woodbridge LSTM on domains generated by `deception_dga`, using the benign ground truth which we used for evaluating the two classifiers in section 4.

[2]Some domains appear multiple times in the Alexa list, with different TLDs. Those duplicates were skipped.

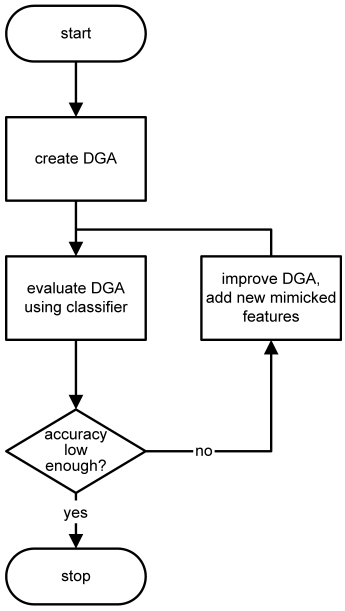

Figure 3: Iterative DGA development process

5.1 The development of deception_dga

As mentioned, the development of the DGA used an iterative approach, with the goal of reducing the accuracy of the DGA detection classifiers. The different versions of the DGA are described below.

5.1.1 Version 0 - baseline. We created a baseline version, which would generate domain names with a length uniformly distributed between 13 and 18 characters, using the alphabet of 'a' - 'z', also uniformly distributed.

- Accuracy of detection: Random Forest: 0.987 LSTM: 1.000
- FANCI features influenced: 0/33

5.1.2 Version 1 - Modelling domain name length. FANCI's first feature is the domain name length. This version of the DGA varies the length of the domain names generated, to make it follow the same statistical length distribution as observed in the Alexa benign ground truth. `deception_dga` now outputs domains with a length varying between 5 and 42 characters.

- Accuracy of detection: Random Forest: 0.873 LSTM: 0.993
- FANCI features influenced: 1/33

5.1.3 Version 2 - Adding vowel ratio. On top of version 1, this version also varies the vowel distribution in generated domain names, as observed in the Alexa ground truth for domain names of a specific length.

- Accuracy of detection: Random Forest: 0.715 LSTM: 0.949
- FANCI features influenced: 2/33

5.1.4 Version 3 - Using character probability distribution. This version abandons generation of characters with a uniform distribution between 'a' and 'z', and instead uses the character probability distribution present in the domain names recorded in the Alexa list. The full domain name alphabet 'a' - 'z', '0' - '9' and '-' is used. Using the character distribution of the Alexa domains, implies that the

Table 3: Comparison of LSTM and Random Forest performance on individual DGAs. The Min, Average, Max and StdDev exclude our specially crafted *deception_dga*. Better scores are indicated in boldface.

	LSTM				Random Forest			
DGA	TPR	FPR	Prec	Acc	TPR	FPR	Prec	Acc
banjori_dga	**1.000**	**0.000**	**1.000**	**1.000**	0.950	0.050	0.951	0.950
chinad_dga	**0.999**	**0.001**	**0.999**	**0.999**	0.998	0.002	0.998	0.998
conficker_dga	**0.967**	**0.048**	**0.952**	**0.959**	0.871	0.129	0.871	0.871
cryptolocker_dga	**0.992**	**0.004**	**0.996**	**0.994**	0.936	0.064	0.937	0.936
dnschanger_dga	**0.994**	**0.012**	**0.988**	**0.991**	0.964	0.036	0.965	0.964
dyre_dga	1.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000
emotet_dga	**0.997**	**0.002**	**0.998**	**0.998**	0.986	0.014	0.986	0.986
gameover_dga	1.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000
gameover_p2p	**1.000**	**0.000**	**1.000**	**1.000**	0.989	0.011	0.989	0.989
gozi_dga	**0.957**	**0.044**	**0.956**	**0.957**	0.857	0.143	0.859	0.857
locky_dga	**0.976**	**0.028**	**0.973**	**0.974**	0.862	0.138	0.863	0.862
murofet_dga	**0.997**	**0.002**	**0.998**	**0.997**	0.960	0.040	0.960	0.960
murofetweekly_dga	1.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000
necurs_dga	**0.984**	**0.021**	**0.979**	**0.981**	0.879	0.121	0.880	0.879
nymaim_dga	**0.968**	**0.049**	**0.952**	**0.960**	0.864	0.136	0.864	0.864
padcrypt_dga	**1.000**	**0.000**	**1.000**	**1.000**	0.992	0.008	0.992	0.992
proslikefan_dga	**0.968**	**0.038**	**0.963**	**0.965**	0.880	0.120	0.880	0.880
pushdo_dga	**0.997**	**0.013**	**0.987**	**0.992**	0.865	0.135	0.875	0.865
pykspa_dga	**0.972**	**0.035**	**0.966**	**0.969**	0.857	0.143	0.857	0.857
qadars_dga	**1.000**	**0.000**	**1.000**	**1.000**	0.994	0.006	0.994	0.994
qakbot_dga	**0.987**	**0.016**	**0.984**	**0.985**	0.905	0.095	0.905	0.905
ramnit_dga	**0.986**	**0.016**	**0.984**	**0.985**	0.883	0.117	0.883	0.883
ranbyus_dga	**0.996**	**0.002**	**0.998**	**0.997**	0.973	0.027	0.973	0.973
rovnix_dga	**1.000**	**0.001**	**0.999**	**0.999**	0.998	0.002	0.998	0.998
symmi_dga	**0.989**	**0.020**	**0.980**	**0.984**	0.936	0.064	0.936	0.936
virut_dga	**0.991**	**0.033**	**0.968**	**0.979**	0.975	0.025	0.976	0.975
deception_dga	***0.871***	***0.160***	***0.844***	***0.855***	*0.599*	*0.401*	*0.599*	*0.599*
Min	**0.957**	0.000	**0.952**	**0.957**	0.857	0.000	0.857	0.857
Average	**0.989**	**0.015**	**0.985**	**0.987**	0.937	0.063	0.938	0.937
Max	1.000	**0.049**	1.000	1.000	1.000	0.143	1.000	1.000
StdDev	**0.013**	**0.017**	**0.016**	**0.014**	0.055	0.055	0.054	0.055

VowelRatio feature is automatically modeled according to the Alexa ground truth, as well as the *DigitRatio* and *Alphabet Cardinality* FANCI features, as well as its list of seven 1-gram features (*mean*, *stddev*, *min*, *max*, *median*, *lower_quartile* and *upper_quartile*). This change did not have a very big impact though: classification accuracy of the Random Forest even slightly increased, while the LSTM accuracy dropped, albeit significantly higher still than the Random Forest accuracy.

- Accuracy of detection: Random Forest: 0.722 LSTM: 0.929
- FANCI features influenced: 10/33

5.1.5 Version 4 - Using character probability, given the previous character. Further in the direction of version 3, we used the overall character probability distribution for the first character (for which we only allowed 'a' - 'z', no dash or digits). All subsequent characters use the distribution frequency as recorded in the Alexa ground truth for a character, given a specific previous character. This approach essentially models bigram-probabilities. This version of the DGA tries to present expected values to the following FANCI features: *Domain Name Length*, *Contains Digits*, *Vowel Ratio*, *Digit Ratio*, *Alphabet Cardinality*, *Ratio of Repeated Characters*, *Ratio of Consecutive Consonants*, *Ratio of Consecutive Digits*, the seven *1-gram features* and the seven *2-gram features*.

- Accuracy of detection: Random Forest: 0.599 LSTM: 0.855
- FANCI features influenced: 22/33

5.2 Results

An overview of the effect in classification accuracy when different features are being 'emulated' by the DGA is shown in Figure 4 [3]. As a general trend, as more features are emulated by the DGA, the Random Forest classifier accuracy starts to degrade. There is one remarkable exception, which is when switching from `deception_dga`

[3]Please, note that the y-axis scale starts at 0.5, since accuracy below 0.5 is meaningless for binary classifiers.

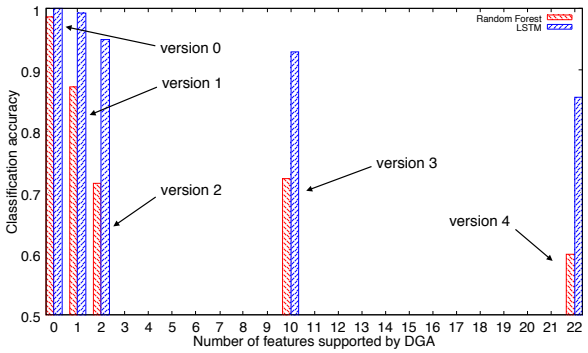

Figure 4: Change of classification accuracy when a certain number of features is actively 'emulated' by the DGA.

version 3 to 4, the number of emulated features increases from 2 to 10, however the Random Forest classification accuracy slightly increases from 0.715 to 0.722 . One possible explanation might be that *Domain Name Length* and *Vowel Ratio* are very heavy weighing features in the decision process of the Random Forest and the *Vowel Ratio* distribution produced when using the character frequency distribution of the Alexa domains may not be as exact as when the *Vowel Ratio* distribution is specifically enforced while generating domain names.

The new DGA also gives the LSTM a hard time, dropping its accuracy down to 0.855, which is the lowest recorded so far for the LSTM, at par with the accuracy of the Random Forest for the `gozi_dga`, `locky_dga`, `nymaim_dga`, `pushdo_dga` and `pykspa_dga` DGAs. Increasing the number of epochs on the LSTM from 10 to 15, marginally increases the accuracy to 0.860. Changing the LSTM output space dimensionality from 128 to 256, results in a classification accuracy of 0.865. Applying both a larger output space dimensionality and increasing the number of training epochs to 15 combined, still results in a classification accuracy of 0.864.

5.3 Practical feasibility of the deception_dga DGA

The version of `deception_dga`, described in section 5.1.5 consists of 535 lines of Python code, including comments and whitespace. This also includes the statistical parameters it collected from the Alexa benign ground truth, which we believe will not change much over time and need never be updated in the code. It does not call (import) any libraries and in fact implements its own *linear congruential generator* for generating random numbers, given a particular seed.

The code, using a single thread on a 2.6 GHz Intel Core i5 processor, generates over 6000 random domain names per second and is a perfectly usable DGA. It completely breaks the Random Forest using the FANCI feature set, since clearly, a *False Positive Rate* of 40% and a *True Positive Rate* of 60% prevent any useful deployment of this classifier.

5.4 Other reflections and observations

We demonstrated that the countermeasures used to evade detection with one specific classifier (Random Forest classifier for which the input features were documented), also affected the classification accuracy of another state-of-the-art LSTM-based classifier, albeit to a lesser extent. Although Woodbridge et al. claimed that LSTMs make it difficult for adversaries to circumvent a classifier without the same training set, our experiments showed that this claim turned out to be not entirely correct. Indeed, adversarial examples that are designed to mislead machine learning models at test-time often transfer, i.e. the same adversarial example fools more than one model. This transferability property has been explored before by Tramèr et al. [22]. An alternative evaluation path that could further confirm this hypothesis would have been to build a new DGA that aims to evade detection by the LSTM based detector, and then evaluating how well it is also able to circumvent the Random Forest based detector.

In the previous experiments, we have used a state-of-the-art Random Forest based DGA classifier and built a new DGA that circumvented this particular classifier. One can argue that given the fact that we explained the countermeasures used by the new DGA to evade detection, security specialists could in turn use this knowledge to further improve their classifiers. Building upon the previous observation, one could automate this arms race between adversaries and defenders with a deep learning method known as Generative Adversarial Networks (GAN) [6], as used in DeepDGA [2]. This class of unsupervised machine learning relies on two deep learning networks, where (1) the generative neural network produces new candidates with the objective to increase the error rate and (2) the discriminative neural network aims to distinguish malicious and benign instances thereby decreasing the error rate. During the training phase, each neural network further improves its own objective until an equilibrium is reached. The adversary would use the generative neural network to produce new domain names, whereas the defender would use the discriminative neural network to improve detection.

6 CONCLUSIONS

In this work, we have compared two state of practice approaches in machine learning for the detection of algorithmically generated domain names, when a sufficient number of DGA-generated domain names is available for training. We establish that the deep learning approach consistently outperforms Random Forests with manually engineered features, where the classification accuracy over a set of 26 real-world DGAs averages 98.7% for the neural net, versus 93.7% for the Random Forest. A significantly lower standard deviation for the neural net accuracy over the different evaluated DGAs also indicates better classification consistency over the different DGAs.

Additionally, we investigated the concern that knowledge of manually engineered features may be abused by DGA developers to create new versions of DGAs which will become less detectable. To this end, we developed a new DGA, using the knowledge of the feature set used by the FANCI system. This DGA indeed renders the Random Forest classifier useless, with its classification accuracy reduced to 59.9%. The deep learning classifier is also affected by the approach of this new DGA, reducing its accuracy to 85.5%.

As future work, we consider the use of the generative neural network of a Generative Adversarial Network (GAN) to strengthen the circumvention capabilities of our DGA against existing deep learning based DGA detectors, while also evaluating whether there is any classification accuracy improvement by the discriminative neural network of this GAN.

ACKNOWLEDGMENTS

This research is partially funded by the Research Fund KU Leuven. The authors would like to thank Daniel Plohmann for his great work on the DGArchive and for making the results of his work available to the research community; as well as the reviewers for their helpful feedback.

The authors also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

[1] Alexa Internet, Inc. . 2018. Alexa. https://www.alexa.com/topsites. [Online; accessed 16-September-2018].

[2] Hyrum S. Anderson, Jonathan Woodbridge, and Bobby Filar. 2016. DeepDGA: Adversarially-Tuned Domain Generation and Detection. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security (AISec '16)*. ACM, New York, NY, USA, 13–21. https://doi.org/10.1145/2996758.2996767

[3] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. 2012. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*. USENIX, Bellevue, WA, 491–506. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/antonakakis

[4] Evan Cooke, Farnam Jahanian, and Danny McPherson. 2005. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. (07 2005).

[5] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *CCS*. ACM, 1285–1298.

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14)*. MIT Press, Cambridge, MA, USA, 2672–2680. http://dl.acm.org/citation.cfm?id=2969033.2969125

[7] Google. 2017. Google Safe Browsing List. https://developers.google.com/safe-browsing/. [Online; accessed August-2017].

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Comput.* 9, 9 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[9] Hyrum Anderson. 2018. Endgame GitHub. https://github.com/endgameinc/dga_predict/. [Online; accessed 20-August-2018].

[10] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS 2019)*.

[11] Zhen Li, Deqing Zou, Shouhuai Xu, Xinyu Ou, Hai Jin, Sujuan Wang, Zhijun Deng, and Yuyi Zhong. 2018. VulDeePecker: A Deep Learning-Based System for Vulnerability Detection. In *NDSS*. The Internet Society.

[12] Miranda Mowbray and Josiah Hagen. 2014. Finding Domain-Generation Algorithms by Looking at Length Distribution. In *25th IEEE International Symposium on Software Reliability Engineering Workshops, ISSRE Workshops, Naples, Italy, November 3-6, 2014*. IEEE Computer Society, 395–400. https://doi.org/10.1109/ISSREW.2014.20

[13] Mayana Pereira, Shaun Coleman, Bin Yu, Martine DeCock, and Anderson Nascimento. 2018. Dictionary Extraction and Detection of Algorithmically Generated Domain Names in Passive DNS Traffic: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings. (09 2018), 295–314.

[14] Daniel Plohmann. 2018. DGArchive. https://dgarchive.caad.fkie.fraunhofer.de/. [Online; accessed 10-September-2018].

[15] 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)*. USENIX Association, Austin, TX, 263–278. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/plohmann

[16] Vera Rimmer, Davy Preuveneers, Marc Juárez, Tom van Goethem, and Wouter Joosen. 2017. Automated Feature Extraction for Website Fingerprinting through Deep Learning. *CoRR* abs/1708.06376 (2017). arXiv:1708.06376 http://arxiv.org/abs/1708.06376

[17] Christian Rossow, Dennis Andriesse, Tillmann Werner, Brett Stone-Gross, Daniel Plohmann, Christian J. Dietrich, and Herbert Bos. 2013. SoK: P2PWNED - Modeling and Evaluating the Resilience of Peer-to-Peer Botnets. *2013 IEEE Symposium on Security and Privacy* (2013), 97–111.

[18] Stefano Schiavoni, Federico Maggi, Lorenzo Cavallaro, and Stefano Zanero. 2014. Phoenix: DGA-Based Botnet Tracking and Intelligence. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, Sven Dietrich (Ed.). Springer International Publishing, Cham, 192–211.

[19] Samuel Schüppen, Dominik Teubert, Patrick Herrmann, and Ulrike Meyer. 2018. FANCI : Feature-based Automated NXDomain Classification and Intelligence. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 1165–1181. https://www.usenix.org/conference/usenixsecurity18/presentation/schuppen

[20] SURBL.org. 2017. SURBL - URI Reputation Data. http://www.surbl.org/. [Online; accessed August-2017].

[21] The Spamhaus Project Ltd. 2017. The Domain Block List. https://www.spamhaus.org/dbl/. [Online; accessed August-2017].

[22] Florian TramÃĺr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2017. The Space of Transferable Adversarial Examples. *arXiv* (2017). https://arxiv.org/abs/1704.03453

[23] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. 2016. *Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques* (4th ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[24] Jonathan Woodbridge, Hyrum S. Anderson, Anjum Ahuja, and Daniel Grant. 2016. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. *CoRR* abs/1611.00791 (2016). arXiv:1611.00791 http://arxiv.org/abs/1611.00791

[25] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan. 2012. Detecting Algorithmically Generated Domain-Flux Attacks With DNS Traffic Analysis. *IEEE/ACM Transactions on Networking* 20, 5 (Oct 2012), 1663–1677. https://doi.org/10.1109/TNET.2012.2184552