Computer Science and Information Systems 19(1):205–227 https://doi.org/10.2298/CSIS210104046S

Applied Machine Learning in Recognition of DGA Domain Names

Miroslav Štampar[1] and Krešimir Fertalj[2]

[1] SekuriPy LLC, Mirka Račkog 10,
10360 Zagreb, Croatia
miroslav.stampar@sekuripy.hr
[2] Faculty of Electrical Engineering and Computing, Unska 3,
10000 Zagreb, Croatia
kresimir.fertalj@fer.hr

Abstract. Recognition of domain names generated by domain generation algorithms (DGAs) is the essential part of malware detection by inspection of network traffic. Besides basic heuristics (HE) and limited detection based on blacklists, the most promising course seems to be machine learning (ML). There is a lack of studies that extensively compare different ML models in the field of DGA binary classification, including both conventional and deep learning (DL) representatives. Also, those few that exist are either focused on a small set of models, use a poor set of features in ML models or fail to secure unbiased independence between training and evaluation samples. To overcome these limitations, we engineered a robust feature set, and accordingly trained and evaluated 14 ML, 9 DL, and 2 comparative models on two independent datasets. Results show that if ML features are properly engineered, there is a marginal difference in overall score between top ML and DL representatives. This paper represents the first attempt to neutrally compare the performance of many different models for the recognition of DGA domain names, where the best models perform as well as the top representatives from the literature.

Keywords: domain generation algorithm, binary classification, supervised machine learning, deep learning, blind evaluation.

1. Introduction

When attempting to establish a connection with *command and control* (C&C) server(s), a certain type of malicious programs (malware) create numerous *domain name system* (DNS) queries for domain names generated in a pseudo-random way, from which the majority never was and will never be registered. With the same initial value (i.e. *random seed*), usually associated with the run-time environment (e.g. current time), attackers can blindly share the same fresh list of domain names with infected hosts, without taking any intermediary steps. Thus, in case of a need for exchanging data with infected hosts, attackers have to register only a few domains from the current list and point them to the C&C server's IP address.

The family of algorithms intended for the described algorithmic creation of domain names is called *domain generation algorithms* (DGAs). Such algorithms can

pseudo-randomly generate a large number of *algorithmic* (or simply DGA) domain names (Table 1) to bypass potential security mechanisms used for detection and blocking of malicious network traffic [1]. Depending on the set of elements used in the pseudo-random generation, DGAs further split to *regular* (R) – using characters, and *dictionary* (D) – using a predefined set of words, where the latter represents the less common, but harder to detect class. In further text, DGA will refer to regular class if not explicitly declared.

Table 1. Examples of DGA domain names

DGA	Type	Example domain names
Bobax	R	*qrwxktojqz.yi.org, ttcwzadqxpm.dynserv.com*
Banjori	D	*earnestnessbiophysicalohax.com*
Cryptolocker	R	*bqwqeiswupyny.org, oocevdwyruhdi.co.uk*
Conficker	R	*ntpyocx.info, kfoqmgax.com, eiwzqeaosf.info*
Dyre	R	*aa1442a1beba3793bbde2582b4127b66ae.cc*
Locky	R	*hrgcmmihpxth.in, cbkmotlvy.yt, ecsiequ.pm*
Necurs	R	*vyguwpynyxaxld.in, caxadsjuygrem.ac*
Pushdo	R	*qaqicvofe.com, cumocuwupjo.com, cumocuwu.kz*
TinyBanker	R	*ghefvfkkxtgg.ru, mqsqytogddne.ru, hosgnecdevwt.ru*

Malware writers choose DGA to create resilient botnet infrastructures [2]. Resilience is primarily assured by disrupting the ability to block malware-related C&C communication, such as in the case of using blacklists of known malicious domain names [3]. Namely, if we consider the situation where each malware family uses its variant of DGA, we can conclude that the whole process of collection, distribution, and usage of blacklist(s) for all up-to-date DGA domains quickly becomes impractical. For example, malware from the Conficker family can generate 250 to 50,000 domain names per day, depending on the variant, while only one of these domains has to be registered by attackers to propagate new instructions to infected hosts [4].

Domain name is a sequence of labels split by dots (e.g. *www.example.com*), with *chosen prefix* (e.g. *example*) and *public suffix* (e.g. *.com*, *.co.uk*) as most distinguished parts. As a public suffix – also known as a *top-level domain* (TLD) – can contain more than one label (e.g. *.co.uk*), the term *effective TLD* (eTLD) is the more correct one [2]. Along with the use of regular registration of domain names (e.g. *3b580fa7.com*), there are cases where malware (e.g. Bobax, Corebot, Symmi, etc.) uses DGA to generate domain names with dynamic domain providers (e.g. *zqjotkxwrq.dyndns.org*). Thus, in this research, we will analyze only properties for the chosen prefix, which represents the arbitrarily selected label for the registered domain.

One important case which perfectly illustrates the necessity to inspect only the chosen prefix, while ignoring other labels, is the usage of *DNS Blacklist Lookup* (DNSBL) services. Such services provide a quick way to lookup entries in centralized databases through the usage of DNS protocol, where server response contains the lookup results. Entries can be anything requiring the additional check, such as suspicious domain names, IP addresses, file hashes, e-mail addresses, etc. Generally, as DNS labels have a

restricted set of ASCII characters that can be used, entries are specially encoded (e.g. Base32 format) and prepended to the domain name used for such service (e.g. *ff572stfjvxezcp5ueuzxsttvebqeaqbaeaq.a.e.e5.sk* – for Avast blacklist lookup). As a result, related domain names can appear to be DGA generated.

As the whole point of DGA is the evasion of potential security mechanisms, malware authors should be able to register any of the generated domain names, while in the case of DNSBL services main domain name is always the same. Otherwise, security providers could simply blacklist the main (i.e. common) domain name to deal with the DGA malware. Hence the necessity to analyze only the chosen prefix of the inspected domain name as it represents the part that can be registered within the eTLD or dynamic DNS registrar.

Thus, network security analysts should be able to programmatically detect C&C communication attempts toward DGA domains and neutralize infected hosts inside their organization(s). Related DNS traffic is generally "noisy" and should be relatively easy to detect with manual inspection, as queried domain names generally do not look like they have been generated by a human. Nevertheless, the real challenge is the automatic recognition of DGA domain names, with as much accuracy as possible, which is the main topic of our research.

2. Brief Introduction to Recognition of DGA Domain Names

One of the essential features of malware that uses DGA to communicate with C&C server(s) is the noticeable amount of failed DNS queries (Note: response code *NXDOMAIN – no such name*) [5]. As such behavior appears as an anomaly when compared to regular traffic, network security analyst should be able to fairly easily recognize artificially created DGA domain names (e.g. *mkhjbvxvuqznmcjmy.com*) – at least for regular DGAs – by manually inspecting the DNS log entries. Nevertheless, in this research, we are trying to find the best method to perform the recognition in an automated way.

In a general case, groups of DGA domain names can be discovered based on a list of responses for failed DNS queries generated by an infected host. Domain names in such groups usually share two or more common attributes, such as length, TLD, client IP address, high Shannon-entropy score [6], similar frequency of occurrences for different types of characters (e.g. vowels, consonants, numbers, etc.), and temporal proximity of associated DNS traffic. Such groups with shared attributes property are also referred to as *clusters* [7][8].

Heuristics (HE) are fast decision-making strategies based on limited information, yet frequently correct [8]. A *simplistic* HE method for identifying DGA domain names may include the search for all failed DNS queries generated per client for domain names having at least 8 characters long chosen prefix (e.g. *bsfwptsyobt.com*) and percentage of vowel occurrences of less than 10%. These conditions are based on general observation where DGA domain names should be sufficiently large to cover a significant number of combinations, while the percentage of vowel usage should be distinctively lower than in any spoken human language.

By processing entries in *Alexa Top 1 Million Sites* (ALEXA1M) [9], list of most popular domain names used in similar research, we found that the mean length of regular domain's chosen prefix is 10.35, percentage of vowel usage is 36.96%, while percentage of chosen prefixes that could trigger false-positive (FP) identification in proposed HE method is 0.08%. Additionally, if the condition where the corresponding DNS query has to result with the lookup failure is applied, the probability of FP identification effectively becomes negligible.

Running such HE method for 24 hours inside the *Class B* production network environment resulted in the detection of three infected hosts which generated queries for different clusters of non-existent domains. By running additional checks with help of specialized service *DGArchive* [10], we found that recognized clusters of DGA domain names are specific to the malware families Conficker, Necurs, and Nymaim. Thus, if the final goal is simply the detection of infected network hosts, running the described HE method should be sufficient in a general case.

While the proposed HE method could be used for a quick check of potential DGA malware presence in network traffic, it is not suitable for usage in systems where it is crucial to perform classifications with high *accuracy* (ACC), i.e. *Intrusion Detection System* (IDS) and *Intrusion Prevention System* (IPS). Hence, as found in related work, some form of supervised *machine learning* (ML) is commonly used to detect DGA domains.

In ML, a *feature* is an individual measurable property or characteristic of a phenomenon being observed [11], where the main challenge is finding the right set of features for learning purposes. In such a case, input data is described appropriately, so the underlying algorithm could find an optimal parametric model connecting input feature vectors with the expected results. In the case of DGA recognition, potential features include the length of chosen prefix and the percentage of vowel occurrences, as described in the simplistic HE method.

Deep learning (DL) is a special class of relatively new ML algorithms, based on *artificial neural networks* (ANN), where feature extraction is automatized [12]. This means that compared to conventional ML algorithms, where human engineer – with a considerable amount of engineering skills and domain expertise – has to choose what features (e.g. length, digit ratio, etc.) to include in the model, DL algorithms can automatically select "critical" features during the learning process. Since DL represents a distinguished field of ML, with key differences in methodology and philosophy when compared to conventional ML algorithms, the corresponding class of algorithms will be referred to as DL, while conventional ML algorithms will be referred to with just ML in further text.

Antonakakis et al. [7] used the statistical *Hidden Markov Model* (HMM) for classification, analyzing features: length and entropy of each label within a domain name, hierarchical level, n-gram distribution, etc. Positive class representatives used for learning were based on 59,144 DGA generated domains, collected inside the virtual environment for a relatively small set of malware families: Conficker, Murofet, Bobax, and Sinowal, while negative class representatives were based on the top 10,000 domain names from ALEXA1M. While authors achieved almost perfect results during the regular learning process, with a *true-positive rate* (TPR) of 99.72% and *false-positive rate* (FPR) of 0.1%, during the evaluation on real-life network traffic authors achieved TPR of 91% and FPR of 3%. To conclude on this point, in cases when research is

focused on severely limited datasets, with a narrow list of DGAs, experimental results are far from those achievable in real-life. Thus, in our research, we included representatives for the majority of known DGAs, which resulted in consistent performance between different datasets and better results in the production environment.

Ahluwalia et al. [13] used *Random Forest* (RF) for classification, analyzing features: number of consonants, number of vowels, number of digits, 3-gram distribution, and total length. Positive class representatives used in the training process were based on the analysis of 100,000 DGA domains for the following malware families: Cryptolocker, Zeus, Conficker, Tinba, Ramdo, Matsnu, Rovnx, and GameOver Zeus. Negative class representatives were based on the analysis of the top 100,000 domains from ALEXA1M. As a result, authors achieved results of TPR 98.96 % and FPR 2.1%. One of the conclusions was that the total length of the domain name is a key feature for identifying DGA domains and that FPR drastically rises for cases below 8 characters. During the feature selection process, we came to the same conclusion that length indeed represents one of the most important features.

Wang and Chen [14] used RF, *Support Vector Machine* (SVM), and *Naive Bayes* (NB) for classification, analyzing features: length and entropy of *second-level domain* (SLD), together with appearance probabilities of contained n-grams (3, 4, and 5), based on the probability lists calculated from most commonly used English terms and ALEXA1M domain names. The best results were achieved with RF – TPR 97.53% and FPR 0.20%, similar to our results for the standard dataset. It should be noted that while authors included statistical analysis for 5-grams as a feature too, we found in experiments that it does not add any additional value to the ML model in terms of performance.

Yu et al. [15] used DL for classification, particularly models based on *Long Short-Term Memory* (LSTM) and *Convolutional Neural Network* (CNN) neural networks, along with comparative ML algorithm RF. For regular training purposes, authors used domain names obtained from real-traffic, for both positive and negative class representatives, while using an additional *gold* set, consisting of ALEXA1M and DGA domain names retrieved from DGArchive, for "ground truth" validation. Final results of ground truth validation were: LSTM – TPR 74.05% and FPR 0.54%, CNN – TPR 72.89% and FPR 0.31%, RF – TPR 71.28% and FPR 1.33%. During the evaluation, we showed that the proposed feature set is inferior to ours. The main reason is the English-bias, resulting in worse performance and inconsistent behavior between different datasets.

Tran et al. [16] did research on the multi-class imbalance in DGA classification, where the distribution of dataset samples for different DGA families is not uniform. Trained models based on ML algorithm *Random Undersampling Boosting* (RUB) and DL algorithm *Long Short-Term Memory* (LSTM) had similar performance in binary classification task – F1 0.98, which is in pair with our results got for the standard dataset, while used DL models scored considerably better in a multi-class classification task. Even though it is not part of our research, based on related work, we assume that DL models are superior in the multi-class classification, mainly because of their intrinsic ability to memorize the prolonged lexicographical patterns of observed DGA domains [17].

Yu et al. [18][19] compared simple *Endgame* (single LSTM layer) and complex DL architectures: *Invincea* (parallel CNN layers), CMU (forward LSTM layer + backward

LSTM layer), MIT (stacked CNN layers + single LSTM layer), NYU (stacked CNN layers). One of the conclusions was that there is surprisingly little difference between evaluated DL models in terms of accuracy, prompting a preference for the simpler architectures, as they are faster to train and score, while less prone to overfitting. Additionally, the authors pointed that an interesting direction for future work would be to test the trained DL models more extensively on domain names generated by new and previously unseen malware families. In our research, we performed such evaluation, where we showed that ML models generally perform better than DL models, particularly in case of unseen regular DGAs.

Pereira et al. [20] used graph-based method *WordGraph* for extracting dictionaries used by dictionary DGAs. The proposed method is completely agnostic to the dictionary used by the DGA and should learn it by itself. The main proposition is that once these dictionaries are known, it should become straightforward to construct a domain name classifier based on them. For training purposes, authors used *ground truth data* based on samples collected from ALEXA1M and DGArchive, similar as in [15]. While results look promising, with an almost perfect score in all tested cases, a couple of questions arise. Particularly, authors state that they were able to extract 81 dictionaries in five days of real traffic, with 15 validated through service DGArchive, while claiming that they manually verified the remaining 66 dictionaries and confirmed they were malicious, without providing any details whatsoever. At the end, they classified those new dictionaries as generated by unknown malware.

At the beginning of our research, the main focus was on ML models. Based on expertise acquired in everyday network analysis and literature review, we chose an initial set of features, which we heuristically adapted during experimental runs. By analyzing the results for each attempt, we quickly concluded that feature engineering represents the critical part of ML modeling. Nevertheless, as DL became more popular in recent studies, mainly because of its flexibility and a lack of requirement for explicit feature declaration, we gradually extended the scope of our study. Based on initial findings, it became clear that DL modeling is the step forward in this field.

In this paper, we present the results of our study, where we objectively and extensively compare the performance of different ML and DL models for the DGA binary classification. For such a task, we engineered a robust feature set and created two independent datasets, including samples for the majority of known regular and dictionary DGAs. In the end, we evaluated the models, where the most interesting findings are related to the performance of best ML and DL models on samples representing previously unseen (i.e. untrained) DGAs, and the usability validation on historical one-year DNS logs collected from the production environment.

3. Methodology

To find the best model for recognition of algorithmic domains, binary classifiers based on the following ML algorithms were trained in a supervised manner and finally evaluated: NB, *Multilayer Perceptron* (MLP), *Linear Discriminant Analysis* (LDA), *Quadratic Discriminant Analysis* (QDA), *k-Nearest Neighbors* (KNN), SVM, *Decision Tree* (DT), *Extra Trees* (ET), RF, *Bagging* (BAG), *Gradient Boosting* (GB), *Extreme*

Gradient Boosting (XGB), *Adaptive Boosting* (AB) and RUB; along with simple DL models: *Simple Recurrent Neural Network* (SRNN), *Gated Recurrent Unit* (GRU), CNN and LSTM (*Endgame*), and complex DL models: *Invincea*, C2W, CMU, MIT, NYU.

Implementation was done using the programming language Python and third-party programming libraries *scikit-learn* (*sklearn*) [21], *keras* [22], *xgboost* [23], and *imbalanced-learn* (*imblearn*) [24], where each represents the de facto standard in its field of operation. Parameter values used during the instantiation of ML and simple DL models can be found in Table 2.

Table 2. Parameter values used in ML and simple DL model instantiations

Model	Library	Classifier / Base layer	Parameter values
NB	sklearn	*GaussianNB*	-
MLP	sklearn	*MLPClassifier*	*hidden_layer_sizes=(128,)*
LDA	sklearn	*LinearDiscriminantAnalysis*	*solver='SVD'*
QDA	sklearn	*QuadraticDiscriminantAnalysis*	-
KNN	sklearn	*KNeighborsClassifier*	*n_neighbors=15*
SVM	sklearn	*SVC*	*kernel='linear'*
DT	sklearn	*DecisionTreeClassifier*	*max_depth=None*
ET	sklearn	*ExtraTreesClassifier*	*n_estimators=128*
RF	sklearn	*RandomForestClassifier*	*n_estimators=128*
BAG	sklearn	*BaggingClassifier*	*n_estimators=128*
GB	sklearn	*GradientBoostingClassifier*	*n_estimators=128*
XGB	xgboost	*XGBClassifier*	-
AB	sklearn	*AdaBoostClassifier*	*n_estimators=128*
RUB	imblearn	*RUSBoostClassifier*	*n_estimators=128*
SRNN	keras	*SimpleRNN*	*units=128*
GRU	keras	*GRU*	*units=128*
LSTM	keras	*LSTM*	*units=128*
CNN	keras	*Conv1D*	*filters=128, kernel_size=4*

After extensive initial tests and analysis of obtained results, we decided to use default parameter values where changes did not result in noticeably better results. To avoid the potential issue with the different number of *estimators* used in ensemble [25] type of ML models and *units* or *filters* in DL models, we chose to set respective parameter values of *n_estimators*, *hidden_layer_size*, *units*, and *filters* to 128 where applicable. Even though it seems to be the popular choice in related work (e.g. [15] [16][18][19][26]), we experimentally confirmed the assumption that higher value should not yield with significantly better results in evaluated classifiers.

For such a task, we used sklearn's *GridSearchCV*, a specialized tool for an exhaustive search for best values over the specified list of parameters. It resulted in the (inconsistent) best case accuracy improvements of less than 0.08%, compared to experimental results got with our chosen parameter values, in case of all models except KNN. In that case, the change of initial value for *n_neighbors* from 5 to *GridSearchCV* suggested 15 resulted in the accuracy improvement of 0.16%. Furthermore, it should be

noted that, in the case of MLP, adding more hidden layers did not result in any improvements.

Based on expertise acquired in everyday network analysis and literature review, we chose the following self-explanatory features for ML modeling purposes: length (I), character (Shannon) entropy (II), (decimal) digit ratio (III), length of the longest vowelless sequence (IV), length of the longest common prefix that can be found in at least two ALEXA1M chosen prefixes (V), length of the longest common suffix that can be found in at least two ALEXA1M chosen prefixes (VI), mean positional distance of nearby vowels (VII), mean ASCII distance of adjacent characters (VIII), number of occurrences of numerical sequences (IX), mean frequency indices of 2-grams (X), 3-grams (XI) and 4-grams (XII) given the previously calculated lists of all possible n-grams found within ALEXA1M chosen prefixes sorted by the number of occurrences. It should be noted that the n-grams not appearing in the ALEXA1M have been treated as the last elements of notable lists, thus avoiding additional penalization.

The first chosen prefix (*example*) in the given example (Table 3) represents the regular domain name, the second chosen prefix (*spiderwjzbmsmu7y*) represents the Tor (anonymity network) domain name – included only for comparative purposes, while the last chosen prefix (*wxkjzdbmowq*) represents the DGA domain name. At first look, features based on ALEXA1M n-grams (X, XI, XII) seem to be the most promising for the recognition of DGA domain names, as there appears to be a significant difference in calculated values between different classes of domain names.

Calculated distances and sequence (run) lengths were specifically inspired by *Diehard* [27] and *FIPS PUB 140-2* [28], specialized batteries of statistical tests for testing the quality of *pseudo-random number generators* (PRNG). The main proposition was that as DGA domain names are generated in a pseudo-random way, they should have better statistical results when tested for pseudo-randomness, compared to regular domain names. While mentioned batteries require significantly larger binary sequences, we derived a couple of simplified tests – namely IV, VII, VIII, IX – to perform similar statistical analysis tests on chosen prefixes for domain names.

Table 3. Example of calculated feature values for different chosen prefixes

Feature	*example (.com)*	*spiderwjzbmsmu7y (.onion)*	*wxkjzdbmowq (.info)*
Length (I)	7	16	11
Character (Shannon) entropy (II)	2.52	3.75	3.28
Digit ratio (III)	0	0.06	0
Length of longest vowelless sequence (IV)	3 (*mpl*)	8 (*wjzbmsmu*)	8 (*wxkjzdbm*)
Length of longest (ALEXA1M) common prefix (V)	7 (*example*)	7 (*spiderw*)	2 (*wx*)
Length of longest (ALEXA1M) common suffix (VI)	7 (*example*)	2 (*7y*)	3 (*owq*)
Mean positional distance of nearby vowels (VII)	3	4	5.5
Mean ASCII distance of adjacent characters (VIII)	11.33	16.4	8.2
Number of occurrences of numerical sequences (IX)	0	1	0
Mean frequency index (ALEXA1M) of 2-grams (X)	142	424	570
Mean frequency index (ALEXA1M) of 3-grams (XI)	1,037	13,443	17,644
Mean frequency index (ALEXA1M) of 4-grams (XII)	4,935	147,773	230,265

As part of the data preparation phase, calculated feature vectors were standardized to normally distributed data, with the sklearn's preprocessing utility class *StandardScaler*. The main reason is the requirement of specific ML algorithms, such as KNN and SVM,

which assume that all features are centered on zero and have variance in the same order, while that same transformation does not affect the performance of other ML algorithms.

Because of DL algorithms' ability to recognize and learn patterns in long sequences, such as in text or images, in DGA classification task samples represent raw numerical representation of chosen prefixes for domain names. Hence, instead of using feature vectors as in the case with ML models, in featureless DL models, chosen prefixes are transformed to integer representations of contained characters. If we know that the maximum length of a label in the domain name is 63, each chosen prefix is transformed to a zero-padded vector of length 63, with elements representing character indices inside the lookup table of valid DNS characters. For example, chosen prefix *google* becomes a numerical vector [7, 15, 15, 7, 12, 5, 0, 0, 0 … 0].

Simple DL models – SRNN, GRU, LSTM, and CNN – are based on *Endgame* [29], mostly because of its simplicity, wide acceptance, and generally good performance, where LSTM can be considered as the *Endgame* itself. Pseudo-code for the creation of simple DL models can be found in the continuation, where italicized identifiers represent utilized keras-specific layers (i.e. classes):

```
// Valid DNS label characters
alphabet := "abcdefghijklmnopqrstuvwxyz0123456789-"
// Maximum length of DNS label
max_label_length := 63
// Length of embedding vector
embedding_vector_length := 128
// Dropout threshold value
threshold := 0.5

function CreateDLModel(main_layer_class)
  m := Sequential()
  m.add(Embedding(input_dim := LENGTH(alphabet)+1, output_dim
:= \
    embedding_vector_length, input_length := max_label_length))
  if main_layer_class = Conv1D then
    m.add(main_layer_class(filters := embedding_vector_length,
\
      kernel_size := 4))
    m.add(GlobalMaxPooling1D())
  else
    m.add(main_layer_class(units := embedding_vector_length))
  endif
  m.add(Dropout(threshold))
  m.add(Dense(1))
  m.add(Activation("sigmoid"))
  m.compile(loss := "binary_crossentropy", optimizer := "adam")
  return m
end function


srnn := CreateDLModel(SimpleRNN)
gru := CreateDLModel(GRU)
lstm := CreateDLModel(LSTM)
cnn := CreateDLModel(Conv1D)
```

Complex DL models – *Invincea*, C2W, CMU, MIT, and NYU – were implemented with the minimum modifications compared to the original work. In case that there were no details regarding certain aspects of the model in the original paper, we chose the preferred solution based on our initial findings and other related work. The basic architecture information can be found in Table 4.

Table 4. Architectures used in complex DL models

Model	Architecture	Reference
Invincea	Parallel CNN layers	[30]
C2W	Forward LSTM + backward LSTM layer	[31]
CMU	Forward GRU + backward GRU layer	[32]
MIT	Stacked CNN layers + single LSTM layer	[33]
NYU	Stacked CNN layers	[34]

It should be noted that we decided to use the name C2W for the LSTM-based model, inaccurately referred to as CMU by Yu et al. [19], while reestablishing the name CMU for the GRU-based model. The reasoning is based on the original (CMU) research [32], where Dhingra et al. explicitly state that the proposed GRU-based architecture "*uses a similar structure to the C2W model in* [31]*, with LSTM units replaced with GRU units*". The same inaccuracy can be found in other derived work (e.g. [35]).

While the process of setting up and training ML models in sklearn was straightforward, in the case of DL models we had to fine-tune certain aspects. Most notably, to prevent *overfitting* in DL models, and thus avoid poorer prediction performance on new datasets (e.g. blind dataset), we used *early stopping*. In this method, the *validation loss* – performance measure function used for cross-validation on a fraction of the training data (10% of training samples) – is calculated after each epoch. In case of no improvement, the whole training process is interrupted. Additionally, to avoid the *local extrema entrapment*, a *patience level* of 5 is used, thus training stops only in case of 5 consecutive epochs without training improvement, while the best model is preserved between epochs for future use.

To evaluate ML and DL models in a "blind trial", two independent datasets were used. Namely, along with the *standard* set, which is generally considered sufficient in similar DGA research, an additional (independent) *blind* set was used. While in the majority of related work standard set is the only one used and split in different ways (e.g. holdout, k-fold cross-validation, random subsampling, etc.) for different purposes, in our research we also used blind set created from unrelated sources and by using different filtering methods. Thus, while the standard set was used for regular training and standard evaluation, with 70% of samples for training and 30% for testing purposes, the blind set was used for additional unbiased blind evaluation of trained models.

Standard set is the primary (regular) dataset that was used for training and basic performance evaluation of corresponding models. Set is balanced, with 739,377 positive samples and the same number of negative samples. Positive samples consist of synthetic

chosen prefixes, representing uniformly distributed 51 regular[1] and 4 dictionary[2] DGAs, while negative samples consist of filtered ALEXA1M chosen prefixes.

Basic filtering of ALEXA1M chosen prefixes was done by excluding all entries, where length (i.e. 4 or less) or the character set used (i.e. non-alphanumeric) could in no way be associated with known DGA. Additionally, "problematic" chosen prefixes that could not be classified manually by network security analyst as non-algorithmic were also excluded, such as *132770*(.com) (decimal digits), *6f76b4c82656094f26*(.com) (hexadecimal digits) or *gtplkcbpl*(.com) (consonant-only). This way we reduced the possibility of introducing undesired noise into the standard set that could potentially affect the performance of trained models.

Synthetic chosen prefixes for regular DGAs were artificially generated based on descriptive regular expressions obtained from DGArchive, with a generalization that the distribution of pseudo-randomly chosen characters is uniform within the predefined character set. For example, Bamital DGA described with the regular expression *[0-9a-f]{32}\.(org|info|co\.cc|cz.cc)$*, resulted in the function $BAMITAL_{DGA}$, returning the string of length 32, pseudo-randomly chosen from a pool of hexadecimal digits. In this way, quality, quantity, and variety of positive classification data dramatically increased compared to other related work, although they were not generated by any existing DGA. The main proposition was that this way we could artificially generate any number of samples for positive classification, without a way to easily differentiate when compared to real DGA runs. As an example, chosen prefix *bde15d38ecc65c8 01a6ab50a59cea738* generated by real Bamital DGA does not have any distinct property – such as length, character domain, or character (Shannon) entropy – when compared to synthetic chosen prefix *f5c087c1905b38e110e30d5a2743469e* generated by synthetic function $BAMITAL_{DGA}$. Pseudo-code for the whole process of creation of synthetic chosen prefixes for regular DGAs is as follows:

```
letters := "abcdefghijklmnopqrstuvwxyz"
digits := "0123456789"


function Generate(alphabet, min_length, max_length)
  a := RandomInteger(min := min_length, max := max_length)
  r := RandomString(pool := alphabet, length := a)
  return r
end function

// Generates sample for BAMITAL DGA
function BAMITAL_DGA()
  a := CONCAT(digits, "abcdef")
  r := GENERATE(alphabet := a, min_length := 32, max_length :=
32)
  return r
end function
// ... functions for 49 more DGA algorithms ...
```

[1] Bamital, Bedep, Blackhole, Bobax, Conficker, Corebot, Cryptolocker, DNS Changer, DirCrypt, Dyre, EKforward, Emotet, Feodo, Fobber, Gameover, Gameover P2P, Gspy, Hesper, Locky, MadMax, Modpack, Murofet, Necurs, Nymain, Oderoor, PadCrypt, Proslikefan, Pushdo, Pushdotid, Pykspa, Pykspa 2, Qadars, Qakbot, Ramdo, Ramnit, Ranbyus, Rovnix, Shifu, Simda, Sisron, Sphinx, Sutra, Symmi, Szribi, Tempedreve, TinyBanker, Torpig, Urlzone, Virut, VolatileCedar, XxHex

[2] Banjori, Gozi, Matsnu, Suppobox

```
// Main procedure
procedure Main()
  a := {BAMITAL_DGA, ...}
  b := Input("# of samples to generate:")
  for i in {0..b} do
    c := RandomSample(pool := a, min_length := 1, max_length :=
1)[0]
    d := c()
    Print(d)
  end for
end procedure

Main()
```

In the case of dictionary DGAs, synthetic chosen prefixes were generated based on reverse-engineered algorithms found in public code repositories[3]. To eliminate the problem related to the usage of the same seed words, trait manifested with repetition of identical patterns across all related samples inside the time-constrained blacklists, we used different seeds found in the "wild". Therefore, in the example of Banjori DGA, we uniformly utilized 37 different characteristic seeds[4] in the process of generation, along with different pseudo-randomly chosen dates.

This way, we eliminated the potential bias specific for related research, where the training and evaluation of models are based on arguable dictionary DGA samples extracted from daily DGA blacklists, with evident excessive repetition of elongated patterns (e.g. *nvpnestnessbiophysicalohax*, *nxzmestnessbiophysicalohax*, *eoyoestnessbi ophysicalohax*, etc.). In our opinion, such unreasonable usage of excessive repetitions in datasets gives an unfair advantage to DL models, having the well-known ability to memorize prolonged lexicographical patterns [17] – while those same patterns usually turn out as useless outside the evaluation environment, mostly because of the narrow period of validity.

Blind set is the control dataset, independent of standard, created to provide the support for unbiased blind evaluation of trained ML models. Set contains 170,045 positive samples and the same number of negative samples. Negative samples consist of 170,045 valid (i.e. non-NXDOMAIN) chosen prefixes collected in the *Class B* production network environment during one month. Positive samples consist of chosen prefixes for real algorithmic domains collected from one week (5-11 July 2020) of DGA blacklist source DGArchive, for 81 DGAs – 76 regular and 5 dictionary DGAs, with 43 regular DGAs and 4 dictionary DGAs appearing in the standard dataset too. The discrepancy is preserved principally for dataset independence preservation, along with the opportunity to analyze model behavior in the expected case of the appearance of previously unseen DGA.

[3] https://github.com/baderj/domain_generation_algorithms and https://github.com/andrewaeva/DGA

[4] *abehmsigotg, alitydevonianizuwb, amentalistfanchonut, anarianaqh, ancorml, anerraticallyqozaw, ardenslavetusul, byplaywobb, ellefrictionlessv, enhancedysb, epictom, ererwyatanb, erionirkutskagl, estnessbiophysicalohax, fordlinnetavox, idablyhoosieraw, inaaforementionedagf, inalcentricem, iologistbikerepil, lcationgreedinessb, leasuredehydratorysagp, llaabettingk, machuslazaroqok, men, orcajanunal, orshipecmascriptivylv, partbulkyf, plefrostbitecycz, rasildeafeninguvuc, rgradienton, rsensinaix, sagabardinedazyx, satformalisticirekb, semitismgavenuteq, sikathrinezad, thoodivettewl, vinskycattederifg*

Negative samples consist of valid chosen prefixes for 437 different TLDs: *.com* (54.46%), *.net* (7.35%), *.hr* (5.61%), *.org* (3.97%), *.uk* (2.23%), *.de* (1.83%), *.it* (1.53%), *.ru* (1.44%), *.rs* (1.33%), *.info* (1.15%), etc. The main assumption used during the collection of negative samples from real traffic was that, in the general case, resolution of DGA domain name would either fail (i.e. NXDOMAIN) or result with the sinkholed response, while the probability for a resolution to a valid non-sinkhole IP address can be effectively ignored. For exclusion of sinkholed domain names, a list of 1,330 IP addresses for known sinkholes has been used, gathered from *Maltrail – Malicious traffic detection system* [36], a specialized IDS system for tracking of malware-related network activities. This way we practically reduced the probability for the inclusion of regular DGA domain names down to zero.

It should be noted that as ALEXA1M represents the list of most popular domain names on the Internet, there is an inherent overlap of negative samples between standard and blind datasets, with a percentage of 37.37%. Although the whole process of creation of datasets is kept independent, this is the single point of sample overlap. As the removal of shared entries from the blind dataset would potentially strengthen the regional bias and move the focus of evaluation on less popular domain names, while at the same time take out the realistic aspect of DNS traffic gathered in the production environment, we decided to leave them.

While a concept *blind* set used in our research is similar to the *gold* set used by Yu et al. [15], there are a couple of crucial differences. Gold set – based on ALEXA1M and DGA domain names from DGArchive – was used for ground truth validation, while blind set – based on real-domains gathered from everyday traffic and DGA domain names from the same source – was used for blind evaluation in our research. As in other related work, during research, we found by trial-and-error that ALEXA1M is the best source for negative samples used in the training process. Additionally, we used real DGA domain names for blind evaluation and synthetically generated positive samples for standard evaluation – specifically avoided by Yu et al., because of their concern on limited availability based on the usual approach with malware runs inside the virtual environment. Therefore, instead of establishing the gold truth dataset and "losing" the possibility of training models based on ALEXA1M, while at the same time providing the independence between datasets used in the standard evaluation and blind evaluation, a blind set was created. Furthermore, conducted blind evaluation can be roughly considered as the measurement of classifier performance in the real-network traffic environment, as all blind dataset samples were either gathered in the production environment or from a daily blacklist of current DGA domain names.

4. Evaluation Results

In preparation for the evaluation, after the initial runs, we noticed that in some cases reduced set of features resulted in slightly better overall results – particularly in the case with simpler ML models such as NB and QDA. Hence, to remove the possibility of training potentially weaker ML models, we performed the *feature selection* beforehand. In such a task, the feature set is being reduced, without significant performance

degradation in the recognition system [37], where features contributing the least in the decision process are discarded.

To ease the process, for each trained ML model based on tree-based algorithms (DT, RF, ET, GB, XGB, and AB) it is possible to extract the *feature importance* (FI) list. Such lists, consisting of calculated feature scores with values ranging from 0 (irrelevant) to 1 (single most important feature), can be used to find how each feature contributes to the overall classification process of the corresponding ML model.

Even though we expected that each ML model will score individual features differently by their importance, during our research we found that some features share a similar level of importance throughout all models (where FI is available). Thus, the observed phenomenon became the basis of our feature selection process, particularly in the case with the least relevant features.

Table 5. FI for different ML models

Feature	DT	RF	ET	GB	XGB	AB	$\bar{F}$
Mean ASCII distance of adjacent characters (VIII)	0.01	0.01	0.01	0.00	0.00	0.01	0.01
Number of occurrences of numerical sequences (IX)	0.00	0.00	0.01	0.00	0.00	0.02	0.01
Character (Shannon) entropy (II)	0.01	0.03	0.02	0.00	0.00	0.02	0.01
Mean positional distance of nearby vowels (VII)	0.01	0.03	0.02	0.00	0.00	0.02	0.01
Length of longest vowelless sequence (IV)	0.01	0.02	0.01	0.00	0.00	0.06	0.02
Mean frequency index (ALEXA1M) of 2-grams (X)	0.01	0.09	0.05	0.00	0.00	0.02	0.03
Digit ratio (III)	0.01	0.01	0.01	0.01	0.03	0.04	0.02
Length (I)	0.06	0.04	0.05	0.06	0.07	0.18	0.08
Length of longest (ALEXA1M) suffix (VI)	0.03	0.09	0.15	0.03	0.06	0.09	0.08
Mean frequency index (ALEXA1M) of 3-grams (XI)	0.02	0.19	0.12	0.01	0.01	0.12	0.08
Length of longest (ALEXA1M) prefix (V)	0.03	0.15	0.12	0.04	0.07	0.12	0.09
Mean frequency index (ALEXA1M) of 4-grams (XII)	0.80	0.34	0.43	0.85	0.76	0.30	0.58

Based on obtained FI (Table 5), we concluded that the mean frequency index (ALEXA1M) of 4-grams makes the most important feature across ML models. More importantly, all features based on lexical ALEXA1M properties generally have greater importance than other features. The relevance of the 4-gram feature (XII) stands out so much compared to others that our immediate impression was that it could be solely used as a HE method for recognition of DGA domains.

Therefore, performing the training process for the “shallow” (i.e. depth set to 1) DT model, primarily chosen due to the intuitive IF-THEN-ELSE resulting structure representing the trained model, we came to the value of 90,674 above which the mean frequency index (ALEXA1M) of 4-grams would have to be valued to classify the tested chosen prefix as DGA. Finally, for comparison purposes with other evaluated models, we created a simple HE method ALEXA4G based only on that single check (Table 6).

Furthermore, as a result of FI analysis, we came to an auxiliary hypothesis that we could use a HE approach in feature selection of evaluated ML models by simply removing features that were at least in one case marked as absolutely irrelevant (i.e.

value 0.00 – highlighted with dashed border in Table 5) – similar to a voting system with right of veto. Thus, we discarded the upper half (VIII, IX, II, VII, IV, X) of features listed in Table 5, while leaving the lower half (III, I, VI, XI, V, XII). To verify the hypothesis, we conducted the training and evaluation of all ML models and compared the performance in both full and reduced sets of features. As a result, in the case of reduced feature set, we got an overall 0.1% improvement of classification performance (Note: based on $\overline{F1}$ score) in all ML models, while the training time has been reduced on average by 73% compared to the original time. Hence, we continued the evaluation with a reduced set of features.

Finally, after the successful evaluation of ML and DL models, the following values were calculated: TPR_A, FPR_A, ACC_A, $F1_A$, TPR_B, FPR_B, ACC_B, $F1_B$, and $\overline{F1}$ (Table 6). The first eight values represent the performance of the corresponding binary classifier, depending on the used dataset (A – standard, B – blind), while the last $\overline{F1}$ represents the mean value based on $F1_A$ and $F1_B$. Basic measures TPR and FPR were chosen as they represent the most basic metrics used in related work. In general, TPR has to be as high, while FPR has to be as low as possible for a model to be acceptable, as otherwise, the detection mechanism could become unusable because of too many positive-misses or too many false-alarms.

Out of all statistical performance measures available for finding the "best" model, ACC and F1 represent the two most commonly used for binary classification in related work. ACC is the measure of all the correctly identified cases and is preferred when the detection of positive and negative classes is of equal importance. For example, in the case of a passive system like IDS, detection of a DGA domain could be considered of the same importance as the detection of a non-DGA domain, as the network security engineer analyzing its report would need to invest additional time "triaging" the false detections of any kind. In comparison, in the case of an active system like IPS, false detection of regular domains would most probably be detrimental to the network-user experience, while missing true detection of DGA-domains up to a certain threshold could be considered as acceptable.

Table 6. Evaluation results for standard (A) and blind (B) datasets

Model	Type	TPR_A	FPR_A	ACC_A	$F1_A$	TPR_B	FPR_B	ACC_B	$F1_B$	$\overline{F1}$
ALEXA4G	HE	0.9216	0.0467	0.9375	0.9364	0.8916	0.0561	0.9178	0.9156	0.9260
NB	ML	0.9295	0.0508	0.9394	0.9387	0.9085	0.0538	0.9274	0.9260	0.9323
RFYu	ML	0.9136	0.0656	0.9240	0.9232	0.9502	0.0530	0.9486	0.9487	0.9359
SRNN	DL	0.9485	0.0303	0.9591	0.9587	0.8855	0.0381	0.9237	0.9206	0.9397
QDA	ML	0.9523	0.0531	0.9496	0.9497	0.9380	0.0669	0.9356	0.9357	0.9427
CNN	DL	0.9477	0.0292	0.9592	0.9588	0.9011	0.0393	0.9309	0.9288	0.9438
C2W	DL	0.9585	0.0225	0.9680	0.9677	0.8835	0.0312	0.9262	0.9229	0.9453
LDA	ML	0.9345	0.0226	0.9559	0.9549	0.9159	0.0404	0.9377	0.9363	0.9456
DT	ML	0.9616	0.0402	0.9607	0.9607	0.9331	0.0698	0.9316	0.9317	0.9462
CMU	DL	0.9604	0.0249	0.9678	0.9675	0.8890	0.0331	0.9279	0.9250	0.9463
LSTM	DL	0.9618	0.0203	0.9707	0.9705	0.8900	0.0297	0.9302	0.9272	0.9488
GRU	DL	0.9639	0.0244	0.9698	0.9696	0.8980	0.0333	0.9324	0.9300	0.9498
MIT	DL	0.9665	0.0236	0.9714	0.9713	0.8948	0.0326	0.9311	0.9285	0.9499
Invincea	DL	0.9571	0.0232	0.9670	0.9667	0.9039	0.0307	0.9366	0.9345	0.9506
NYU	DL	0.9667	0.0254	0.9707	0.9706	0.9015	0.0333	0.9341	0.9319	0.9512
SVM	ML	0.9604	0.0276	0.9664	0.9662	0.9393	0.0566	0.9413	0.9412	0.9537
AB	ML	0.9648	0.0239	0.9705	0.9703	0.9419	0.0570	0.9425	0.9424	0.9564
RUB	ML	0.9636	0.0207	0.9714	0.9712	0.9349	0.0510	0.9419	0.9415	0.9564
ET	ML	0.9659	0.0220	0.9719	0.9717	0.9377	0.0519	0.9429	0.9426	0.9572
BAG	ML	0.9664	0.0199	0.9733	0.9730	0.9369	0.0505	0.9432	0.9428	0.9579
GB	ML	0.9656	0.0192	0.9732	0.9730	0.9380	0.0484	0.9448	0.9444	0.9587
RF	ML	0.9668	0.0192	0.9738	0.9736	0.9379	0.0496	0.9441	0.9438	0.9587
XGB	ML	0.9645	0.0177	0.9734	0.9731	0.9365	0.0457	0.9454	0.9449	0.9590
KNN	ML	0.9670	0.0188	0.9741	0.9739	0.9397	0.0488	0.9454	0.9451	0.9595
MLP	ML	0.9711	0.0224	0.9743	0.9742	0.9486	0.0567	0.9459	0.9461	0.9602

Even though ACC represents one of the most intuitive and obvious measures, the inclusion of F1-score has become the de facto standard in recent related work. One of the main reasons is the ongoing criticism, with claims that ACC solely cannot be considered as a reliable measure anymore, because it provides an over-optimistic estimation of the classifier ability on the majority class [38]. Nevertheless, as ACC and F1 represent the performance of the model distinctly, while at the same time making results comparable to other research, we included both as part of evaluation results and chose the mean value $\overline{F1}$ as the final score for each model.

Along with models described in the methodology part, the following comparative models were also included: ALEXA4G – representing the HE method based solely on the ALEXA1M 4-gram feature (XII) and RFYu – representing the ML model from related work (Yu et al. [15]), with English-biased set of features.

From the evaluation results, it is clear that in most cases there is a consistency in performance between datasets, where better models generally scored better in both

datasets. Thus, the blind evaluation could be considered as the verification of standard evaluation results, where comparative and simpler (probabilistic) ML models scored the worst, ensemble and more "powerful" ML models scored the best, while DL models were in the middle.

Only evident inconsistency in performance can be found in the case of the comparative RFYu. While in the case of the blind dataset it scored almost the same as correlative model RF, its performance has been remarkably poor in the case of the standard dataset, where even the comparative HE method ALEXA4G based on a single feature had better results. With close inspection of false recognitions, particularly FPs, we found that RFYu has a problem with regular domain names containing Internet characteristic non-English n-grams, such as *xpose360*, *mp3koka*, *newxxxvideos*, *1080ip*, *c365play*, *win7dwnld*, etc., same n-grams that could be better learned by the model if only the underlying features *nl2* and *nl3* [15][39] were based on real-domain names instead of the English language.

Better scoring of model RFYu in the case of the blind dataset could be explained by the considerably smaller size of the dataset compared to the standard dataset, where negative samples, consisting of regular domain names collected in real-network traffic, have a greater ratio of the most popular English-language oriented domains. To verify this, we calculated the mean of means for frequency indices of English 3-grams for negative samples, and got a value of 1,788.04 in the case of the standard dataset and a lower value of 1,602.55 in the case of the blind dataset, proving that blind dataset is indeed more English-oriented.

Thus, the results of Yu et al. [15] – particularly performance comparison of DL models and RFYu against the truth-marked *gold set* – and the conclusion of DL superiority should be re-evaluated with a better ML feature set. Used comparative ML model RFYu is English-biased with inconsistent performance between different datasets compared to other models (Table 6), and most importantly underperforming in the case of ALEXA1M – the same set of domains used as a source for negative samples in the gold set. Hence, this can be considered as a prime example of an assertion that feature engineering represents the critical part of ML modeling.

For a detailed comparison between ML and DL classes, we chose the best performing models – MLP (ML) and NYU (DL) – and further analyzed results for DGAs that had a TPR_B less than 0.90 in at least one of those models (Table 7). In the case of dictionary DGA Suppobox, NYU had a TPR_B of 0.46, while MLP underperformed with a TPR_B of just 0.01. As a result of sample analysis, we found in both standard and blind datasets multiple usages of Suppobox characteristic suffixes such as *sherburne*, *underhill*, *electricity*, and *blackwood*, from where we concluded that DL models are superior to ML models in similar cases where the same characteristic prolonged lexicographical patterns can be found in distinct datasets.

Nevertheless, in the case of dictionary DGAs Gozi, Matsnu, and Nymaim2 both models performed almost the same. While Nymaim2 represents the case of non-trained dictionary DGA, where both models perform badly as expected, Matsnu represents the case of trained dictionary DGA where even the DL model failed. By comparison of Matsnu samples in both datasets and the DGA algorithm internals, we concluded that this DGA is particularly problematic for both training and recognition because of lack of repeating patterns, mainly due to the extensive number of combinations generated from

large internal wordlists and the way they are combined (*nouns* and *verbs*) until the predefined chosen prefix length.

In the case of regular DGAs, the MLP model generally scored better than NYU, with the substantial difference in performance for untrained DGAs Darkshell, Qhost, and Qsnatch. By inspecting related samples in the blind dataset, we concluded that NYU most probably failed to recognize those because of their shortness and consequently the lack of information to classify those as positives. In the case of Darkshell, samples had a length of 6 (e.g. *r038zy*), in the case of Qsnatch samples on an average had a length of 5 (e.g. *4xxgz*), while in the case of Qhost samples shared the same prefix *ptmr*, with numeric suffix having a mean length of 4. Nevertheless, ML representative MLP could recognize those due to lack of usage of most common n-grams.

Table 7. TPR_B for "problematic" DGAs

DGA	Type	Trained	MLP (ML)	NYU (DL)
Conficker	R	⊤	0.91	0.89
Darkshell	R	⊥	1.00	0.13
Diamondfox	R	⊥	0.91	0.82
Gozi	D	⊤	0.41	0.41
Matsnu	D	⊤	0.05	0.06
Nymaim2	D	⊥	0.04	0.04
Pitou	R	⊥	0.56	0.56
Pushdo	R	⊤	0.58	0.59
Pykspa2	R	⊤	0.83	0.88
Qhost	R	⊥	1.00	0.29
Qsnatch	R	⊥	0.91	0.34
Simda	R	⊤	0.43	0.53
Suppobox	D	⊤	0.01	0.46
Symmi	R	⊤	0.67	0.67
Szribi	R	⊥	0.85	1.00
UD3	R	⊥	1.00	0.75
UD4	R	⊥	0.93	0.79
Vawtrak	R	⊤	0.66	0.64
Virut	R	⊤	0.76	0.75
Volatilecedar	R	⊤	0.55	0.47

By comparing overall performance concerning the complexity of models, the simplest ML models (NB, QDA, LDA, and DT) scored worse than more complex ML models, while in the case of DL there was no clear distinction. Even though complex NYU, *Invincea*, and MIT scored the best among DL models, simple GRU and LSTM scored better than complex CMU and C2W. Thus, we can confirm the results from Yu et al.

[18][19] and agree with the remark that simpler DL architectures should be preferred over complex DL architectures.

Although ML models generally scored better than DL models, the difference in the final score is marginal. The only significant difference can be found in the case of blind evaluation, where the best ML model MLP performed better in cases with untrained regular DGAs, while the best DL model NYU performed better in cases with trained dictionary DGAs where identical prolonged lexicographical patterns could be found in both datasets. Thus, when considering its flexibility, powerful ability to automatically memorize lexicographical patterns, and the fact that it does not require an extra step of careful feature engineering, a critical process that creates the competitive difference between comparative RFYu and correlative RF model, we came to the conclusion that DL can be considered as the preferred choice over ML for the recognition of DGA domain names.

Even though it was not formally included in the evaluation results (Table 6), we also evaluated the comparative simplistic HE method described in the introductory Section **Error! Reference source not found.**, based only on the chosen prefix length and vowel ratio. While at first glance it scored poorly with TPR_A 0.1708, TPR_B 0.2053, and $\overline{F1}$ 0.3160, it had remarkably low FPR_A 0.0006 and FRP_B 0.0012. If we assume that DGA malware generates at least 5 DGA DNS queries per day, we can conclude that it could be used to easily detect the infected client on the same day of infection, with an exceptionally low probability for FP detection.

As part of usability validation, we chose the best ML model MLP, the best DL model NYU and the simplistic HE method, and ran those against the historical DNS logs for 850 million queries collected inside the production environment for one year (Note: 1st July 2019 to 30th June 2020). As a result, we detected 2 clusters for the following DGAs: Conficker and Dromedan, where verification and DGA type recognition were based on query service provided by DGArchive. Cluster Conficker was active for 277 days, with 9 infected clients, while cluster Dromedan was active for 189 days, with 6 infected clients. All clusters have been detected by both models and a HE method, with a daily average TPR of 0.95 for both models and a daily average TPR of 0.31 for the HE method.

Consequently, during the execution, we realized that the usability of ML and DL models is surprisingly low, at least if used against the DNS queries. No matter the generally good evaluation results (Table 6), even FPR as low as 0.01 (i.e. 1%) effectively results in useless reports. For example, if the daily expected number of queries – as in our case – is 2 million, with 20,000 unique chosen prefixes, FPR of 0.01 results with 200 chosen prefixes being misclassified as DGA – or tens of thousands of wrongfully blocked DNS queries for non-DGA domains if used in an active system like IPS. Thus, the best candidate that could be used against the DNS queries, without additional fine-tuning of relevant thresholds, was the simplistic HE method based just on chosen prefix length and vowel ratio. Its exceptionally low FPR emerges it from other models in the real-life traffic environment. In our case, only 13% of all positive detections were FPs, which means that on the daily average positive detections of 62 DGA domains, only 8 domains were not related to a known DGA. As a result of further analysis, we found that the FPs were in majority of cases related to Ad-serving related networks, where operators deliberately use DGA-alike domains (e.g. *bmkz57b79pxk.com*, *01mspmd5yalky8.com*, *wk4x5rdtoz2tn0.com*) to make them more

resilient to potential blocking from clients. A common feature that differentiates them in DNS traffic from DGA domains is that those DGA-alike domains in general case resolve to valid IP addresses, which highlights the need for inspection of failed DNS responses (i.e. NXDOMAIN) in search of DGA traffic, at least in passive systems like IDS.

5. Conclusions

In this paper, we presented the results of standard and blind evaluations for 14 ML and 9 DL models, along with 2 comparative models, for the recognition of DGA domain names. For such a task, along with the standard dataset used for training and standard evaluation, we used an additional independent "blind" dataset for blind evaluation. By choosing blacklisted DGA domain names and regular domain names collected from production network traffic in case of the blind dataset, in comparison to synthetic creation of positive samples and ALEXA1M list of most popular domain names in case of the standard dataset, we successfully ensured the independence.

Based on calculated FI from different tree-based ML models, we performed a veto-based feature selection process and concluded that the mean frequency index (ALEXA1M) of 4-grams makes the most important feature across trained ML models. To verify this finding, we created a comparative HE method ALEXA4G based on this single feature, which during the evaluation scored only 3.5% worse than the best performing model MLP. Furthermore, we concluded that features based on lexical ALEXA1M properties (i.e. n-grams, longest common prefix, and longest common suffix) are overall more important than all other used features, meaning that the domain-based features should be the focus of related ML modeling.

As a result of the evaluation, we found that ML models generally score better than DL models, although the difference in overall score is marginal. Concerning the complexity of models, the simplest ML models (NB, QDA, LDA, and DT) scored worse than more complex ML models, while in the case of DL there was no clear distinction. Nevertheless, a substantial difference between ML and DL classes could be found with untrained regular DGAs Darkshell, Qhost, and Qsnatch having short chosen prefixes, where best ML model MLP performed considerably better than best DL model NYU, and in the case with trained dictionary DGA Suppobox, where DL model scored better because identical characteristic prolonged lexicographical patterns could be found in both standard and blind datasets.

Thus, when considering its flexibility, powerful ability to automatically memorize lexicographical patterns, and the fact that it does not require an extra step of careful feature engineering, a critical process that creates the difference between good and bad same-architecture ML models, we concluded that DL can be considered as the preferred choice over ML for the general recognition of DGA domain names. Nevertheless, in the case of the creation and usage of specialized models, ML should be the preferred choice for the recognition of regular DGAs, while DL should be the preferred choice for the recognition of dictionary DGAs.

Usability validation for best performing ML model MLP and DL model NYU was done on historical one-year DNS query logs, along with the comparative simplistic HE method based on chosen prefix length and vowel ratio. As a result, we detected 2

clusters for the following DGAs: Conficker and Dromedan, with both models and a HE method. Surprisingly, we found that the best candidate for usage against the DNS queries, without additional fine-tuning of relevant thresholds, was the simplest – simplistic HE method. Its exceptionally low FPR emerges it from other models in the real-life traffic environment, resulting in a practically acceptable number of positives from the perspective of network security analysts. To improve the usability of ML and DL models, the focus of DNS traffic analysis should be moved from queries to failed responses, inadvertently losing the possibility to use such models in an IPS.

Ideas for future work include further research related to specialized recognition of dictionary DGAs and usage of *hybrid* ML and DL models, where benefits should be taken from and drawbacks alleviated of both, to increase the prediction accuracy and decrease the computational complexity.

References

1. Y. Zhou, Q. Li, Q. Miao, and K. Yim, "DGA-Based Botnet Detection Using DNS Traffic.," J. Internet Serv. Inf. Secur., vol. 3, no. 3/4, pp. 116–123, 2013.
2. S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based botnet tracking and intelligence," in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, 2014, pp. 192–211.
3. M. Kührer, C. Rossow, and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in International Workshop on Recent Advances in Intrusion Detection, 2014, pp. 1–21.
4. M. Thomas and A. Mohaisen, "Kindred domains: detecting and clustering botnet domains using DNS traffic," in Proceedings of the 23rd International Conference on World Wide Web, 2014, pp. 707–712.
5. T. Wang, X. Hu, J. Jang, S. Ji, M. Stoecklin, and T. Taylor, "BotMeter: Charting DGA-botnet landscapes in large networks," in IEEE 36th International Conference on Distributed Computing Systems (ICDCS), 2016, pp. 334–343.
6. S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "FANCI: Feature-based automated nxdomain classification and intelligence," in 27th USENIX Security Symposium, 2018, pp. 1165–1181.
7. M. Antonakakis et al., "From throw-away traffic to bots: detecting the rise of DGA-based malware," in Proceedings of 21st USENIX Security Symposium, 2012, pp. 491–506.
8. C. Dietrich, "Decision making: Factors that influence decision making, heuristics used, and decision outcomes," Inq. J., vol. 2, no. 02, 2010.
9. "Alexa Top 1 Million Sites," Alexa Internet, Inc. [Online]. Available: http://s3.amazonaws.com/alexa-static/top-1m.csv.zip. [Accessed: 15-Mar-2021]
10. D. Plohmann, "DGArchive," Fraunhofer FKIE. [Online]. Available: https://dgarchive.caad.fkie.fraunhofer.de/. [Accessed: 15-Mar-2021]
11. C. M. Bishop, Pattern recognition and machine learning. springer, 2006.
12. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.
13. A. Ahluwalia, I. Traore, K. Ganame, and N. Agarwal, "Detecting broad length algorithmically generated domains," in International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments, 2017, pp. 19–34.
14. T. Wang and L.-C. Chen, "Detecting Algorithmically Generated Domains Using Data Visualization and N-Grams Methods," in Proceedings of Student-Faculty Research Day, CSIS, Pace University, 2017, pp. D4–1.

15. B. Yu, D. L. Gray, J. Pan, M. De Cock, and A. C. A. Nascimento, “Inline DGA detection with deep networks,” in IEEE International Conference on Data Mining Workshops, ICDMW, 2017, vol. 2017-Novem, pp. 683–692, doi: 10.1109/ICDMW.2017.96.
16. D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, “A LSTM based framework for handling multiclass imbalance in DGA botnet detection,” Neurocomputing, vol. 275, pp. 2401–2413, 2018.
17. L. Sidi, A. Nadler, and A. Shabtai, “MaskDGA: A black-box evasion technique against DGA classifiers and adversarial defenses,” arXiv Prepr. arXiv1902.08909, 2019.
18. B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, “Character level based detection of DGA domain names,” in 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–8.
19. B. Yu et al., “Weakly supervised deep learning for the detection of domain generation algorithms,” IEEE Access, vol. 7, pp. 51542–51556, 2019.
20. M. Pereira, S. Coleman, B. Yu, M. DeCock, and A. Nascimento, “Dictionary extraction and detection of algorithmically generated domain names in passive DNS traffic,” in International Symposium on Research in Attacks, Intrusions, and Defenses, 2018, pp. 295–314.
21. F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, no. Oct, pp. 2825–2830, 2011.
22. F. Chollet, “Keras - Deep Learning for humans,” 2015. [Online]. Available: https://github.com/keras-team/keras. [Accessed: 15-Mar-2021]
23. T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.
24. G. Lemaitre, F. Nogueira, and C. K. Aridas, “Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning,” J. Mach. Learn. Res., vol. 18, no. 1, pp. 559–563, 2017.
25. T. G. Dietterich, “Ensemble methods in machine learning,” in International workshop on multiple classifier systems, 2000, pp. 1–15.
26. R. Sivaguru, C. Choudhary, B. Yu, V. Tymchenko, A. Nascimento, and M. De Cock, “An evaluation of DGA classifiers,” in 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 5058–5067.
27. G. Marsaglia, “Diehard: battery of tests for random number generators,” CD-ROM, Department of Statistics and Supercomputer Computations Research Institute, Florida State University. 1995.
28. R. Brown and J. Burrows, “FIPS PUB 140-2 Security Requirements For Cryptographic Modules,” 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf. [Accessed: 15-Mar-2021]
29. J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, “Predicting domain generation algorithms with long short-term memory networks,” arXiv Prepr. arXiv1611.00791, 2016.
30. J. Saxe and K. Berlin, “eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys,” arXiv Prepr. arXiv1702.08568, 2017.
31. W. Ling et al., “Finding function in form: Compositional character models for open vocabulary word representation,” arXiv Prepr. arXiv1508.02096, 2015.
32. B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. W. Cohen, “Tweet2vec: Character-based distributed representations for social media,” arXiv Prepr. arXiv1605.03481, 2016.
33. 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.

34. X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in Advances in neural information processing systems, 2015, pp. 649–657.
35. C. Choudhary, R. Sivaguru, M. Pereira, B. Yu, A. C. Nascimento, and M. De Cock, "Algorithmically generated domain detection and malware family classification," in International Symposium on Security in Computing and Communication, 2018, pp. 640–655.
36. M. Stampar and M. Kasimov, "Maltrail - Malicious traffic detection system." 2014 [Online]. Available: https://github.com/stamparm/maltrail. [Accessed: 15-Mar-2021]
37. P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," Pattern Recognit. Lett., vol. 15, no. 11, pp. 1119–1125, 1994.
38. D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," BMC Genomics, vol. 21, no. 1, p. 6, 2020.
39. B. Yu, L. Smith, M. Threefoot, and F. G. Olumofin, "Behavior Analysis based DNS Tunneling Detection and Classification with Big Data Technologies.," in IoTBD, 2016, pp. 284–290.

Miroslav Štampar is an information security specialist at the SekuriPy LLC, Zagreb, Croatia. His major research interests are network security, malware analysis, machine learning, and vulnerability assessment in information systems. He is author of world-renowned free and open source applications. He has given talks in a number of international expert conventions on information security and programming.

Krešimir Fertalj is a full professor at the Faculty of Electrical Engineering and Computing (FER) at the University of Zagreb, Croatia, where he lectures a couple of computing courses on undergraduate, graduate, specialist and doctoral studies. His professional and scientific interest is in automated software engineering, complex information systems, project management and in software security. He led several scientific and research projects and a few dozen of development projects. He was a mentor to students for over 250 bachelor and graduate theses, nine MSc and eleven PhD theses. He has published near 200 scientific and technical papers. He is the founder of the Laboratory for Special Purpose Information Systems and of Postgraduate Specialist Study "Project Management" at FER. He is a senior member of IEEE and a full member of Croatian Academy of Engineering (HATZ). He served as a Head of Department at FER, a Secretary of Department of Information Systems of HATZ and was one of the founders and member of management board of PMI chapter in Croatia.

Received: January 12, 2021; Accepted: July 08, 2021.