Soft Computing (2020) 24:5517–5537
https://doi.org/10.1007/s00500-018-03703-8

FOCUS

Scalable detection of botnets based on DGA

Efficient feature discovery process in machine learning techniques

Mattia Zago[1] · Manuel Gil Pérez[1] · Gregorio Martínez Pérez[1]

Published online: 18 January 2019

Abstract
Botnets are evolving, and their covert modus operandi, based on cloud technologies such as the virtualisation and the dynamic fast-flux addressing, has been proved challenging for classic intrusion detection systems and even the so-called next-generation firewalls. Moreover, dynamic addressing has been spotted in the wild in combination with pseudo-random domain names generation algorithm (DGA), ultimately leading to an extremely accurate and effective disguise technique. Although these concealing methods have been exposed and analysed to great extent in the past decade, the literature lacks some important conclusions and common-ground knowledge, especially when it comes to Machine Learning (ML) solutions. This research horizontally navigates the state of the art aiming to polish the feature discovery process, which is the single most time-consuming part of any ML approach. Results show that only a minor fraction of the defined features are indeed practical and informative, especially when considering 0-day botnet identification. The contributions described in this article will ease the detection process, ultimately enabling improved and more scalable solutions for DGA-based botnets detection.

Keywords Botnet · Domain generation algorithm · DGA · Machine Learning · Natural language processing

1 Introduction and motivation

Computer networks enable sharing resources, immediate communications and distributed computation, which have been strengthened in recent years by the Cloud Computing paradigm where "everything" is being offered as an on-demand service (Mell and Grance 2011). Unfortunately, such network functionalities are misused by malicious entities aiming to compromise as many systems as possible in order to have them available for later use. Such compromised hosts are generically defined as *zombies* (or bots), being part of a network controlled by one or multiple command & control (C&C) servers. In order to conceal the infection and hide the botnet purposes, malwares are employing a variety of techniques such as code obfuscation and encryption (Vormayr et al. 2017; Gupta et al. 2016). Nevertheless, bots need to reach the C&C servers so as to receive commands and pull-out recorded data. There is a potential weak point from attackers perspective: once the C&C is taken out, or seized by the authorities, the botmaster–botnet owner will lose the control over the botnet (Leelasankar et al. 2018; Lerner 2014).

As a consequence, malwares are actively employing advanced evasion techniques to conceal the communications with C&C servers. A domain generation algorithm (DGA) represents a practical solution from the attacker's point of view. With DGAs, we indicate a family of algorithms that given a seed, often shipped with the malware as a pre-shared secret, generate strings of domain names that can be queried and resolved for locating the active C&C server. In addition, these evasion characteristics are currently being enhanced by using the cloud computing environment (Sharieh and Albdour 2017; Stergiou et al. 2018), in which both bots and C&C

Communicated by B. B. Gupta.

✉ Gregorio Martínez Pérez
gregorio@um.es

Mattia Zago
mattia.zago@um.es

Manuel Gil Pérez
mgilperez@um.es

[1] Department of Communications and Information Engineering, Faculty of Computer Science, University of Murcia, Campus de Espinardo s/n, 30100 Murcia, Spain

servers can be provisioned dynamically and being moved from one location to another or even between providers within the cloud (Bugiel et al. 2011; Hussain et al. 2017). This fact makes botnets more difficult to trace and detect in real time (Zhang et al. 2014). Moreover, cyber-criminals are continuously updating their malwares' DGA in order to evade regular patterns and signatures created by security vendors. These updated versions are considered as 0-day threats. To be more precise, a 0-day is a threat that was previously unknown, which can be either in form of a new variant of a known malware or an entirely new malware family.

Most DGAs algorithms are both time-dependent and deterministic, i.e. their generation parameters are retrievable and reusable to compute all the possible outcomes. Hypothetically, it is possible to reverse engineer each malware variant to obtain the generation algorithm and the seed, thus extracting the subset of valid domain names (DNs) for a given date and time. This approach is not viable when considering the number of malware families and variants (0-days) discovered every day. Moreover, even excluding exceedingly large algorithmically generated domains (AGDs) produced by malwares such as Conficker or Virut, which can generate an average of 50,000 domains per day (Plohmann et al. 2016), a blacklisting approach is not a practical solution. This huge number of information discovering C&C connectivity entails a key challenge to address the big data problem (Watkins et al. 2017). In this context, this work focuses on the detection approach, aiming to prevent malwares to contact C&C servers by distinguishing between legitimate and malicious DNs, taking into account big data analysis algorithms for reduction. And, by its own definition, AGDs are dynamically generated as pseudo-random strings or by combination of dictionary words; consequently, it is possible to separate legitimate DNs from the others by applying Machine Learning (ML) techniques. Under the hood, our ML framework leverages two separate families of metrics: on one side static and dynamic analysis of Domain Name Service (DNS) queries highlights suspicious behaviours of clients, while on the other side, natural language processing (NLP) techniques permit to accentuate linguistic differences between DNs. We called these two families Context-Aware and Context-Free, respectively.

Although different in nature, these two families of metrics are complementary. DNS queries analysis can provide evidence of non-human activity, such as daily similarity and repeating communication patterns. For example, it has been proven that legitimate users repeat daily patterns (Bilge et al. 2014). Similarly, NLP-based metrics help to identify non-human activities; that is, DNs are intended to be easily remembered by users, or at least mnemonic, since the main purpose of any DNS service is to provide human-readable association with IP addresses. By comparison, most malware families do not include such forethoughts (Bilge et al. 2014). DNS-related metrics can natively deal with the concept of fast-flux botnets, a subset of the DGA-based family. This technique, spotted in the wild since at least 2007 (with the Storm Worm (Holz et al. 2008), consists in registering multiple IP addresses within the same DNS-A record, leveraging the well-known round-robin DNS scheme to provide short-lived C&C rendezvous-points.

Due to privacy concerns, it is important to state also that metrics based on network analysis are difficult to obtain and use. However, Context-Free metrics are anonymous and more privacy-oriented since they do not require any contextual information from the users or the network state. In other words, users' confidentiality is compromised by packet inspection and network analysis. Additionally, the literature review (Sect. 2) and preliminary results (Sect. 3.3) suggest that these confidentiality-harming solutions may be subdued by more privacy-oriented solutions that do not require users' contextual information in order to achieve excellent detection results.

A key challenge in this field is the comparability of different models. From the theory of ML, it is clear that there are three key aspects of any learning model: data sources, feature characterisation and model optimisation. This work focuses on homogenising the existing Context-Free features in order to be able to test different models with the same data source.

Lastly, it is worth stating that in this research paper two different ML problems are dealt with; that is, given a set of fully qualified domain names (FQDNs), (i) binary separate legitimate domains from malware ones and (ii) categorise them according to their malware family.

Therefore, the contributions of this work are threefold:

- Firstly, a horizontal survey of the state of the art in terms of features used in ML and Deep Learning (DL) approaches to solve cybersecurity challenges related to DGA-based botnets;
- secondly, the analysis of the two aforementioned ML problems and their potential solution using privacy-preserving approaches; and,
- thirdly, a deep analysis of the previously mentioned features to highlight their properties with respect to the feature engineering process and their evaluation using six amongst the most important ML algorithms.

To do so, this article firstly recollects the usage of both families (Context-Aware and Context-Free) in the literature, aiming to establish which features are used, by whom and with which results. Section 2 presents these outcomes. Secondly, in Sect. 3, this work proposes an analysis of the Context-Free features using two different feature selection and extraction techniques and evaluated using six differ-

ent classifiers. Finally, conclusions and the further work are drawn and discussed in Sect. 4.

2 DGA-based botnet detection in the literature

Despite the efforts spent to fight them, DGAs are still thrusting a good portion of the most advanced malware families. As stated before, classic blacklisting approaches are inadequate to contrast malwares, just consider that the DGA powering the now-obsolete Conficker malware can generate up to 50 thousands DNs per day. As illustrated in Kührer et al. (2014), public blacklists were lacking in terms of DGA coverage with less than 1.2% of DGAs analysed by the authors being contained in any of the blacklists. Bruteforcing those generation algorithms is thus not going to solve the problem (especially when considering 0-days variants); however, Artificial Intelligence (AI) can help to tackle them. AI techniques and precisely ML algorithms have been not only proven applicable but also relevant and well suited when tackling these malwares (Tran et al. 2018; Zhang et al. 2016). Along with the classic network detection techniques, e.g. honeypot-based or signature-based, Alieyan et al. (2017) cited passive DNS-based anomaly detection techniques based on Graph Theory (GT), entropy, statistics, Neural Network (NN), Decision Tree (DT) and clustering. Our aim is to extend this taxonomy to provide a more complete and sound classification to achieve better performance when detecting DGA-based botnets; that is, we build a taxonomy based on:

1. The ML approach used, that is either supervised, not supervised or semi-supervised.
2. The families of features adopted in the learning model, that is Context-Aware features (e.g. DNS inspection), Context-Free features [e.g. lexical analysis (Fu et al. 2017] and, finally, a Featureless model [e.g. NN-based learners (Mac et al. 2017)].

To this extent, and with respect to the aforementioned taxonomy categories, this research article will adhere to the following definitions:

Definition 1 (*Context-Aware feature*) A feature that is dependent on the specific malware sample execution, which is realised in a precise environment with a specific configuration and in a particular time frame; for example, Features extracted upon DNS-response inspection.

Definition 2 (*Context-Free feature*) A feature that is related only to a FQDN and thus is independent of contextual information, including, but not limited to, timing, origin or any other environment configuration. First and foremost example of this family is the lexical analysis of the domain name.

In summary, the Context-Free feature family represents the complement set of the Context-Aware feature, that is, a feature can either belong to the Context-Aware or the Context-Free family, but not both.

With regards to the Context-Aware feature family, they are subject to a number of limitations that should be taken into account, so exiting data sets containing them are rare, outdated and generally partial. For example, on the one hand, AGDs lists such as Malware Domain List (2009), Abakumov (2016), Risk Analytics (2007) and OSINT are limited, fragmented and generally outdated. On the other hand, bigger repositories of network traces such as Biglar Beigi et al. (2014) and García et al. (2014) are crafted, heavily unbalanced and often include only short burst of malware packets. They are usually collected in a test environment or hand-crafted because mass-collecting real user data are, in fact, a direct breach of user's privacy, and can therefore only be gathered in an anonymous way after explicit consent. Nonetheless, the quality of these data sets is notable (Biglar Beigi et al. 2014; García et al. 2014), and when correctly used, they can be of great help to any detection model. On the contrary, data sets with Context-Free features (Malware Domain List 2009; Abakumov 2016; OSINT; Risk Analytics 2007) consist mainly of lists of FQDNs belonging to a specific malware family. They are natively privacy-oriented since users' data are not involved in any phase of the processing.

Finally, it is worth mentioning that in the literature exist several models that do not make use of hand-crafted, or even automatically guessed features. These models can take advantage of both types of data sets, thus we consider them as a third, distinct category. We define such models "Featureless":

Definition 3 (*Featureless model*) A Machine Learning model that does not require features in order to learn the training data set.

Both Mac et al. (2017), Woodbridge et al. (2016) and Vinayakumar et al. (2018) claim that feature-based detection systems can be easily circumvented by malware engineers in the context of the Adversarial Machine Learning theory. The motivation for such claims generically relies on the intrinsic difficulties of defining and analysing the feature set suitable for not only differentiate AGDs from legitimate FQDNs but also to separate and pinpoint different malware families.

On one side, it is clear that featureless detection systems can produce good results without the need of ideate a feature set; but on the other side, an extensive and deep knowledge about the specific subject represented by the data enables the feature engineering process to converge to an optimal feature set. Feature-based detection systems are in general more reliable and offer important qualities such as the transparency, efficiency, scalability and the capabilities of fine-tuning the

algorithms. Not all features, though, are strictly relevant or helpful during the detection process, i.e. a given feature might not be relevant *as-is* for the detection model, but in combination with others it may produce optimal results.

To reflect the taxonomy we propose in this research article, this section is divided into two subsections according to the ML paradigm chosen by the authors of the different related works of the literature that we have thoroughly analysed, either supervised or unsupervised learning.

Generally speaking, the act of labelling a data set such as a collection of domain names is an extremely complicate and resource-consuming task, especially because white and blacklists can help only to a certain extent. unsupervised learning is, unlike supervised, missing the knowledge related to the instances, i.e. the model does not know the label of each point in the data set. It is imperative to understand that supervised and unsupervised learning are different both in nature and in scope of application: on the one hand, supervised learning techniques partition a data set into subsets, named *clusters*, according to some common characteristics; on the other hand, unsupervised learning algorithms divide the instances into classes and use that knowledge to infer the class (label) of new and previously unseen elements.

With regards to all the aforementioned aspects, in each subsection, a list of the most used or relevant algorithms for that category is presented. Furthermore, each subsection will provide several tables that report:

- the reference of the work;
- the type of classifier or cluster used;
- an indication whenever the authors made a comparison with other works or other methods;
- whether their proposed framework is capable of realtime (RT) detection;
- the algorithm proposed;
- the usage of either Context-Aware or Context-Free features; and
- a generic field that considers the overall results — specifically, we consider as *poor* performances whenever the proposed results (in terms of precision and recall) are below 75%, *average* below 85%, *good* below 95% and *excellent* above 95%.

Remarkably, it is worth noticing that only a few authors have cited any challenge related to 0-day, either as part of their analysis (Nguyen et al. 2015; Pu et al. 2015; Tong and Nguyen 2016) or as potential future work (Ahluwalia et al. 2017; Fu et al. 2017; Thomas and Mohaisen 2014).

2.1 Supervised machine learning approach

A supervised ML algorithm is a function that associates a label (also known as outcome or dependent variable) to a given set of predictors (also known as independent variables). The learning process consists of optimising the internal parameters of the algorithm to associate the input set with the desired output. Examples of supervised ML algorithms are Decision Tree (DTs), Random Forests (RFs), k-Nearest Neighbours (kNN), Hidden Markov Models (HMMs), Neural Network (NNs), etc. Amongst them, we considered only the most effective that have been successfully used in the recent past to detect DGA-based malwares in the network. Here follows a brief list of such approaches.

2.1.1 Hidden Markov Model (HMM)

HMMs are the simplest class of dynamic Bayesian Networks (BNs) and, specifically, they are Markov Models in which the states are hidden (unobservable). HMMs have a proven record of successful applications in the linguistic field when applied to the extraction of grammars and information from texts.

In the literature, they have been used by several authors in order to distinguish legitimate DNs from malicious AGDs. On one side, for example, Mac et al. (2017), Tran et al. (2018), Woodbridge et al. (2016) showed that HMMs behave poorly in comparison with more complex featureless solutions such as Long Short-Term Memory Networks (LSTMs). In general, HMMs have unsatisfactory performances when applied to binary classification between AGDs and legitimate DNs (Woodbridge et al. 2016). HMMs, however, have been successfully deployed by Antonakakis et al. (2012) to trace back the C&C servers, achieving interesting results only when targeting specific classes of malware. Table 1 presents a comparison of these previous works, especially with Deep Learning (DL) and feature aware solutions.

On the other side, however, Fu et al. (2017) decided to use HMMs and probabilistic context-free grammars (PCFGs) to extract core properties of legitimate DNs, in order to build a new family of DGAs able to guarantee the generation of statistically undistinguishable AGDs when referring to lexical analysis (as specified below in Sect. 3). As reported by the authors, the inclusion of others lexical properties, in combination with network-based features, helps in overcoming this concealing technique. Extending the work proposed in Anderson et al. (2016), in terms of using Generative Adversarial Network (GAN) as an anti-detection approach, may result in interesting outcomes. Table 2 presents a comparison of the work used to evade the detection techniques.

Table 1 Supervised approach—Hidden Markov Models

Refs.	Classifier	Comparison	RT	Algorithm	Feature context		Results
					Aware	Free	
Antonakakis et al. (2012)	C&C identification	✗	✗	HMM	Featureless		Poor
Mac et al. (2017) and Tran et al. (2018)	MultiClass	Feature aware, DL	✗	HMM	Featureless		Poor
Woodbridge et al. (2016)	MultiClass	Feature aware, DL	✗	HMM	Featureless		Poor

Table 2 Supervised approach—avoiding detection

Refs.	Used for	Comparison	Algorithm
Anderson et al. (2016)	AGDs generation	Other DGAs	GAN, LSTM
Fu et al. (2017)	AGDs generation	Other DGAs	PCFG, HMM

2.1.2 Artificial neural network (NN) and deep learning

Artificial Neural Networks (NN) are mathematical structures that combine nonlinear functions to compute complex functions. They ultimately aim to resemble the structure of human neurons and interactions. One of the most impressive results of NNs is that it has been proved that they can approximate the result of any function [Universality Theorem (Haykin 1998)]. There is a catch, though, in using NN for such task: they can only approximate continuous functions. Deep Neural Networks come to improve this result.

Amongst the past decades, NNs have been widely and successfully used for image processing, speech recognition and text analysis (Abdel-Hamid et al. 2014). NNs have been adapted to several problems by changing the composition and the number of the inner layers to align with complex problems. As, for example, Baruch and David (2018) designed a NN with a single strongly connected hidden layer and a single output neuron to binary distinguish between legitimate domains and AGDs.

In the case of DGAs, Extreme Learning Machines (ELMs) have been proved effective in the classification of such domains (Mac et al. 2017; Tran et al. 2018; Shi et al. 2017). Nevertheless, NNs have a major shortcoming, the lack of any persistence mechanism. As a result, Recurrent Neural Networks (RNNs), LSTM (more commonly known as Deep Learning) and a bunch of other techniques, including Convolutional Neural Network (CNN) and Cost-Sensitive Neural Network (CS-NN), have been developed to include, respectively, short-term and long-term persistence. Mac et al. (2017) and Tran et al. (2018) studied and developed a specific variation of classic LSTMs to include binary and multiclass classification models with class-dependent cost-sensitive functions. Despite the very good performances in binary classification, they are still unable to distinguish malwares using pronounceable AGDs.

To be more precise, the performances of the most advanced DL techniques (Mac et al. 2017; Tran et al. 2018; Woodbridge et al. 2016) are achieving excellent results only in the binary case, i.e. when distinguishing between malware and legitimate FQDNs. In the multiclass classification, and especially with regards to the most advanced malware families (such as Kraken, CryptoWall or Qakbot), DL-based detection solutions achieve questionable results in terms of both precision and recall.

Woodbridge et al. (2016) also used LSTMs to learn the sequence of patterns generated by DGAs, ultimately classifying AGDs and legitimate DNs. Vinayakumar et al. (2018) also proved the excellent results of RNNs and LSTMs (and their combinations) when solving the binary classification problem of distinguishing legitimate and harmful domain names. Table 3 presents a comparison of the previous works.

As a final remark, it is worth mentioning that DL has many issues that are usually skipped during the evaluation phase of the proposed works. Although it is correct that they can offer impressive results, it is also correct that they are often overfitted and especially opaque. This lack of transparency ultimately leads to the impossibility of fine-tuning the algorithms and of explicating the reasons behind the results. As for HMMs, Deep Learning can be used to outshine the concealing capabilities of classic DGAs. It is the case studied and reported by Anderson et al. (2016), who developed a DGA specifically designed for crafting difficult DNs. The AGDs were used then to harden the proposed classifier, resulting in a GAN architecture able to lower the detection rate below any acceptable threshold. Related works used to evade the detection techniques have been presented and compared above in Table 2.

2.1.3 Decision trees (DTs) and derived

Contrarily to the Deep Learning solutions presented in Sect. 2.1.2, Decision Trees (DTs) offer transparent solutions that do not require any scaling or data normalisation. Moreover, since they are particularly resilient to outliers, missing values and nonlinear relationships, they are capa-

Table 3 Supervised approach—neural network and deep learning

Refs.	Classifier	Comparison	RT	Algorithm		Feature context		Results
				NN-based	LSTM	Aware	Free	
Baruch and David (2018)	Binary	Feature aware	✗	NN	✗	✗	✓	Good
Mac et al. (2017) and Tran et al. (2018)	MultiClass	HMM, feature aware	✗	CS-NN, ELM	CNN-LSTM	Featureless		Excellent[a]
Shi et al. (2017)	Binary	Feature aware	✗	ELM	✗	✓	✓	Excellent
Vinayakumar et al. (2018)	MultiClass	Feature aware	✗	I-RNN, CNN	CNN-LSTM	Featureless		Excellent
Woodbridge et al. (2016)	MultiClass	HMM, feature Aware	✗	✗	✓	Featureless		Excellent

[a]While classifying some classes. The scores are excellent when considering micro-averaging the per-class scores, but on macro-averaging the results are quite poor

ble of remaining consistent regardless of the data shape. Nonetheless, they are prone to errors and misconfiguration, i.e. without pruning and limitations they tend to overt the data, thus lowering the prediction accuracy. Although it is common to find a scenario-specific algorithm that performs better [typically DL or AdaBoost (AB)], they conventionally require more resources or time to be trained.

Several works have been proposed so far in the context of DTs, not surprisingly using the classic C4.5 as generation algorithm. Specifically, Ahluwalia et al. (2017) and Bilge et al. (2014) have used them to solve the binary classification problem of distinguishing AGDs from FQDNs, while Antonakakis et al. (2012), Mac et al. (2017), Stevanovic et al. (2017), Tran et al. (2018), Truong and Cheng (2016), Vinayakumar et al. (2018) have instead opted for the multi-classification problem of identifying the malware family.

Random Forests (RFs) have been proposed to help to fix the aforementioned overfitting bias of DTs. RFs require almost no parameter tuning, as DTs can handle any type of feature without scaling or normalisation. They do not require tweaking of the hyper-parameters, perform implicit features selection and train extremely fast. However, these accomplishments are often paid with the slow evaluation performances and the important amounts of resources required creating and storing them. Nevertheless, RF, out of the box, performs particularly well. Not surprisingly RFs have been widely used in the literature to approach the DGA-based botnet problem both in the binary (Ahluwalia et al. 2017; Xu et al. 2017) and in the multiclass (Luo et al. 2017; Song and Li 2016; Stevanovic et al. 2017; Truong and Cheng 2016; Vinayakumar et al. 2018; Woodbridge et al. 2016) forms.

As specified before, both DT-based and RF-based models need features in order to work. Generally speaking, the usage of Context-Free features (as defined in Sect. 2) is sufficient to have a good-to-excellent classifier. To be more precise though, it is worth mentioning that to the best of our knowledge, there are not related researches that only uses Context-Aware features for the classification with DT or RF algorithms. In fact, when considering Context-Aware metrics they tend to have them integrated with NLP-based ones (Bilge et al. 2014; Stevanovic et al. 2017; Xu et al. 2017), while on the contrary, researches featuring DT and/or RF solutions tend to focus purely on linguistic features (Ahluwalia et al. 2017; Luo et al. 2017; Mac et al. 2017; Song and Li 2016; Tran et al. 2018; Truong and Cheng 2016; Xu et al. 2017).

Table 4 presents a comparison of these previous works.

It may perhaps be observed that the authors have used a different subset of these features, both belonging to the Context-Aware and Context-Free families. To clarify this observation, Sect. 3.2 will highlight and present, to the best of our knowledge, the complete list and description of the used features.

It also worth mentioning how Vinayakumar et al. (2018) used a CNN to generate the features that later on have been analysed by the DT and the RF classifier. Moreover, highlight how these classifiers are compared with several algorithms that are intrinsically different, including classic approaches such as SVM, NB and SGD.

2.1.4 Other supervised approaches

Apart from DL and DT derived learners, historically exist several other decision algorithms. Amongst them, the five most used algorithms are presented in Table 5 and described in the following list.

- *Naïve Bayes (NB)*, a family of probabilistic classifiers that assume a strong (and thus naïve) independence between the features. In the context of AGDs detection, it has been applied with inconsistent results (Truong and Cheng 2016; Vinayakumar et al. 2018).
- *Regression*, both linear and logistic regressions represent a family of learners that attempt to define an explanation model by interpolating the data. To the best of our knowledge, this approach has not yet been studied and applied to the problem.

Table 4 Supervised approach—decision trees and derived

Refs.	Classifier	Comparison	RT	Algorithm		Feature context		Results
				DT	RF	Aware	Free	
Ahluwalia et al. (2017)	Binary	✗	✗	✓	✓	✗	✓	Excellent
Antonakakis et al. (2012)	MultiClass	✗	✗	✓	✗	✗	✓	Excellent
Bilge et al. (2014)	Binary	✗	✓	✓	✗	✓	✓	Excellent
Luo et al. (2017)	MultiClass	✗	✓	✗	✓	✗	✓	Good
Mac et al. (2017)and Tran et al. (2018)	MultiClass	SVM, DL	✗	✓	✗	✗	✓	Average
Song and Li (2016)	MultiClass	✗	✗	✗	✓	✗	✓	Good
Stevanovic et al. (2017)	MultiClass	✗	✓	✓	✓	✗	✓	Good
Truong and Cheng (2016)	MultiClass	NB, kNN, SVM	✗	✓	✓	✗	✓	Good
Vinayakumar et al. (2018)	MultiClass	AB, NB, DL	✗	✓	✓	CNN-generated		Excellent
Woodbridge et al. (2016)	MultiClass	HMM, DL	✗	✗	✓	✗	✓	Excellent
Xu et al. (2017)	Binary	AB, SGD	✓	✗	✓	✓	✓	Excellent

Table 5 Supervised approach—other machine learning techniques

Refs.	Classifier	Comparison	RT	Algorithm	Feature context		Results
					Aware	Free	
Baruch and David (2018)	Anomaly	DL	✗	SVM, kNN	✗	✓	Good
Han and Zhang (2017)	Binary	Accuracy (Luo et al. 2017)	✗	SVM	Filter	✓	Good
Mac et al. (2017) and Tran et al. (2018)	MultiClass	DT, RF, DL	✗	SVM	✗	✓	Good
Truong and Cheng (2016)	MultiClass	DT, RF	✗	NB, kNN, SVM, AB	✗	✓	Poor
Vinayakumar et al. (2018)	MultiClass	RF, DT, DL	✗	NB, AB	CNN-generated		Excellent
Xu et al. (2017)	Binary	DT, RF	✗	SGD, AB	✓	✓	Excellent

– *Support Vector Machines (SVMs)* represent a family of linear models that attempt to classify data by finding a hyperplane that maximises the data distance in an n-dimensional space. A common approach is to use the SVM as binary classifier to separate legitimate and malicious AGDs (Baruch and David 2018; Han and Zhang 2017), but also as multiclass classifier (Mac et al. 2017; Tran et al. 2018; Truong and Cheng 2016).
– *k-Nearest Neighbours (kNNs)*, a classifier that defines the boundaries of the classes by the distance from their neighbours through a majority vote. In one case, Truong and Cheng (2016) successfully applied this algorithm to the problem, obtaining poor results. However, Baruch and David (2018) have used this approach for anomaly detection, obtaining interesting results.
– *Stochastic Gradient Descend (SGD)*, a family of classification algorithms that approximate gradient optimisation. Xu et al. (2017) have reportedly used it for comparison.

As specified before, and similarly to the previous subsections, Table 5 reports the comparison of these works in terms of RT usage, algorithms, feature families and results.

2.2 Unsupervised machine learning approach

Amongst the algorithms making use of unsupervised learning techniques, it appears that in the literature the K-Means, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Hierarchical Clustering (HC) algorithms are the most used. To this extent, it is clear that unsupervised learning has a concrete and important advantage compared with supervised learning, as it does not require labelled data sets. However, the labels of each cluster's instance can be inferred by a few manually recognised examples. The main drawback of most of unsupervised learning algorithms resides in the fact that they need to be configured with the predicted number of clusters, an assumption that may not be available.

In the literature, unsupervised learning has been applied in many scenarios with different objectives, including, but not limited to:

– statistical filtering techniques to filter out the most basic AGDs (Grill et al. 2015);
– techniques to correlate bots and C&Cs (Han and Zhang 2017);

Table 6 Unsupervised approach—K-Means and derived

Refs.	Cluster	Comparison	RT	Algorithm		Feature context		Results
				K-Means	Other	Aware	Free	
Antonakakis et al. (2012)	Multiclass	✗	✗	X-Means	✗	✗	✓	Excellent
Bisio et al. (2017)	Multiclass	✗	✓	✓	✗	Filter	✓	Unclear[a]
Nguyen et al. (2015)	Multiclass	DBSCAN	✗	X-Means	✗	✓	✓	Average
Pu et al. (2015)	Multiclass	✗	✗	✓	✗	✗	✓	Unknown
Stevanovic et al. (2015)	Binary	✗	✗	✓	✗	✓	✓	Good
Tong and Nguyen (2016)	Multiclass	DBSCAN	✗	X-Means	✗	Filter	✓	Unclear[a]

[a] Authors are not providing detection metrics such as accuracy, precision and recall

– techniques to label groups of FQDNs according to their similarity (Berger and Gansterer 2013; Stevanovic et al. 2015);
– detection tools to separate legitimate from malicious FQDNs (Baruch and David 2018; Pu et al. 2015); and
– techniques to group users according to their network behaviour, e.g. users who query the same group of non-existent domains (NXDomains) (Antonakakis et al. 2012).

A clear trend in the literature is the usage of unsupervised learning techniques to perform some sort of data preprocessing prior to performing real classification with more resource-consuming supervised algorithms. Several works are, in fact, at least partially based on clustering and/or association, such as the ones presented in Antonakakis et al. (2012), Fu et al. (2017) and Nguyen et al. (2015), just to reference a few.

2.2.1 K-means and derived

The well-known K-Means algorithm is the first and the simplest unsupervised learning algorithm. It requires defining *a priori* the number of desired clusters and maps each point in the data set with the same label of the nearest cluster centroid. These centroids evolve and move during the iterations of the algorithm, ensuring the adaptability of the algorithm to different scenarios and applications. It is worth noticing that although it produces an output in a finite amount of time, it may not be the optimal result, which may be strongly dependent from the initial configuration. Essentially, it has three major drawbacks (Pelleg and Moore 2000), that is, it scales poorly, the number of clusters has to be supplied beforehand and it may output local optimal results.

Several authors Bisio et al. (2017), Pu et al. (2015), Stevanovic et al. (2015) have used the K-Means to, at least partially, solve the AGDs detection problem. Knowing the aforementioned issues, Antonakakis et al. (2012), Bisio et al. (2017), Tong and Nguyen (2016) have instead opted for a variant known as X-Means (Pelleg and Moore 2000) that estimates the number of clusters and their parameters, facing though a sensible amount of resources required (Nguyen et al. 2015). Table 6 presents a comprehensive list of literary works that have been reportedly used either this algorithm or a derived version.

Using an appropriate feature set, the K-Means algorithm is capable of associating collected FQDNs in clusters, which can be later on used to recognise 0-day AGDs with similar characteristics to previously discovered malwares (Tong and Nguyen 2016). Such module can be used as a filter to collect and then discard the AGDs generated by known malwares, in order to focus on the ones that are potentially generated by new DGAs.

2.2.2 Other unsupervised approaches

Amongst the historically notable unsupervised learning algorithms, it appears, to the best of our knowledge, that only a few of them have been studied and presented in the past decade to deal with the problem of detecting DGA-based botnets. In fact, apart from the K-Means and its variants highlighted in the previous section, there are two approaches to clustering: the Mixture Model (MM) and the HC, but their usage is quite limited and with questionable results.

To be more precise, the MM attempts to model the data by using a mixture of probability distributions, while the HC uses distance (and linkage) functions to separate (or join) groups of points in the data set. While the HC has been proposed a few times (Zhang et al. 2016; Thomas and Mohaisen 2014; Tu et al. 2015; Fu et al. 2017), to the best of our knowledge the MM clustering has not been applied so far to this subject. Similarly, the Bipartite Graph Clusterings (BGCs) approach used by Han and Zhang (2017) does not need to indicate a number of clusters and it has the ability to find any geometric clusters. But, in the course of running, this algorithm requires high computational resources when running online in comparison with K-Means (Tong and Nguyen 2016).

Finally, we can also find the DBSCAN algorithm that, similarly to the agglomerative HC approach, join together

Table 7 Unsupervised approach—other algorithms

Refs.	Usage	Comparison	RT	Algorithm	Feature family		Results
					DNS	NLP	
Han and Zhang (2017)	C&C identification	✗	✗	BGC	✓	✗	Unknown
Zhang et al. (2016)	Detection	✗	✗	HC	✗	✓	Average
Thomas and Mohaisen (2014)	Detection	✗	✗	HC	✗	✓	Average
Tu et al. (2015)	Correlation	✗	✗	HC	✓	✗	Good
Fu et al. (2017)	Correlation	✗	✗	HC	✗	✓	Average
Nguyen et al. (2015)	Identification	Schiavoni et al. (2014)	✗	DBSCAN	✓	✓	Average
Tong and Nguyen (2016)	Identification	K-Means	✗	DBSCAN	Filter	✓	Average

elements of the data set according to the density of the region in which they reside, being capable of modelling clusters of any spatial shape. A few authors Nguyen et al. (2015) and Tong and Nguyen (2016) have used it in order to circumvent the need of initialising the parameters of the aforementioned K-Means. Table 7 presents a list of these works that have just been discussed.

2.3 Discussion and key points

The proposed state of the art has highlighted a few important inconsistencies in terms of solutions for tackling DGA-based botnets. The foremost notable shortage is undoubtedly the lack of structured data sources, especially when considering those suitable for ML algorithms. The preprocessing phase for the *raw-data* such as the PCAP files or the network flows is not trivial, and its consequent exploratory analysis is considerably time-consuming. As a result, the evaluation of the proposed classifying and clustering algorithms is quite a challenge. Several algorithms have been proposed, and authors have reported discordant results. As previously cited, a common ground may lead to improved and, most importantly, comparable results.

For another thing, although most of the reported works are focusing on the multiclass analysis of malware families, the binary case should not be *a priori* excluded. Further researches are required in order to establish a signature for the legitimate FQDNs so to be able to filter out suspicious DNS queries for subsequent analysis.

Finally, it is worth noticing the staggering absence of solutions capable of targeting 0-day malware variants and families. In fact, only a few authors Nguyen et al. (2015), Pu et al. (2015), Tong and Nguyen (2016) have included at least a partial analysis of the subject, even if, intuitively, this may be explained through the native predisposition of unsupervised learning techniques towards unknown samples and classes. In brief, the 0-day detection still represents an open research topic in detection as well as in other phases of the cybersecurity process (Lobato et al. 2018; Nespoli et al. 2018).

To this extent, we define the following challenges and potential research lines that should be considered by the cybersecurity community and industry.

1. Firstly, privacy-oriented data sets must be researched and made publicly available;
2. secondly, it is mandatory to establish a series of shared best practices that lead and advise future researches related to ML applications for botnet detection;
3. thirdly, to enable the research community to focus on the study of new approaches and detection algorithms instead of data gathering and preprocessing, ML-oriented and ready-to-use data sets must be researched and made publicly available; and,
4. finally, having taken into consideration the delicate and complex nature of the data, *ad-hoc* nonlinear solutions might be worth investigating.

To summarise, the scientific community, but also the security vendors, might benefit from exploiting the aforementioned research lines. Ideally, by establishing a common ground in terms of data, feature sets, procedures and eventually reactions, the future researches might only focus on providing algorithms and advanced solutions for detection, reaction and mitigation purposes.

3 Defining a common base for feature analysis

In Machine Learning, the process of feature analysis is well known for being complex and extremely time-consuming (Bishop 2006; Almomani et al. 2018), conjecture that has been confirmed once again in the case of DGA-based botnet detection (Woodbridge et al. 2016). It also requires a deep knowledge of the data in conjunction with the algorithm that attempt to model the training data. In this context,

and as specified in Sect. 2, we consider two macro-categories for features related to DGA-based botnet detection, namely Context-Aware (Definition 2) and Context-Free (Definition 1) features. They are, respectively, dependent on the instantiation of the malware sample (e.g. DNS response Time-to-Live, TTL) and independent from it (e.g. the number of characters in the queried FQDN). Clearly, being the focus of this work on feature analysis, we *a priori* exclude the Featureless (Definition 3) solutions. Similarly, we exclude the automatic generated features solutions due to their strong dependence from the input data and not on the definition of the features itself.

This work only focuses on the Context-Free category that, being independent of the state of the network, permits us an evaluation not susceptible to the environment variations occurring where and when the malware is executed. As a consequence, further researches are required in order to analyse the Context-Aware feature family.

The features collected and presented in this work are *individual*; they are extracted from a single and precise FQDN (e.g. the domain length) so that the feature set does not include *aggregated* features such as the average of such domain length.

This section is divided into three parts. Section 3.1 briefly introduces the methodology and the data used for carrying out the comparison, Sect. 3.2 highlights the most informative features existing in the literature and, finally, Sect. 3.3 presents a detailed evaluation of these features through a series of experiments.

3.1 Methodology

In order to be as variate as possible, we retrieved a list of collected AGDs from public available data sets (Bader; Netlab 360; Plohmann 2015; Risk Analytics 2007) and malware blacklists (Malware Domain List 2009; OSINT). The following families using DGAs were taken as data source due to their importance, usage or complete reverse engineering status: Alureon, Conficker, CryptoLocker, Goz, Kraken, Matsnu, Murofet, Nymaim, Pushdo, QakBot, Ramdo, Rovnix, Shiotob, Simda, Tinba and Zeus. To establish a comparison with the real-world legitimate FQDNs, we retrieved the list of the top-ranking domains according to Majestic-12 Ltd: The Majestic Million (2018) backlink data set. Our collected data set includes a thousand distinct samples for each class, with the ones that we consider *a priori* legitimate. We extracted from each sample the list of features described in Sect. 3.2 and, based on the histogram analysis, performed the suggested data preprocessing operations, e.g. transformation, scaling and normalisation. The data have been preprocessed with alternatively Orange3 (Demšar et al. 2013) and Weka (Fran et al. 2016).

In order to do so, we will:

1. Present the list of features extracted from the state of the art in the category of individual Context-Free features.
2. Present a selection of distribution histograms and the correlation matrix from the new feature set obtained from the ranking method.
3. Apply the Principal Component Analysis (PCA) to extract n features that cumulatively cover at least 98% of the variance on the original feature set.
4. Evaluate the original data with the six most used classifiers in the state of the art (RF, NN, SVM, DT, AB and kNN), using the well-known accuracy, precision, recall, area under the curve (AUC) and F1 scores.

3.2 Most informative features

By reviewing the literature on the subject of DGA-based botnet discovery, we collected a total amount of 40 features, of which 17 are related to the NLP nGrams, and thus recalculated for every value of $n \in \{1, 2, 3\}$. Table 8 reports these findings. Specifically, the first 23 features,[1] presented in Table 8 are related to the metrics that can be extracted by analysing the domain name as if it were a string of text. It makes sense, as a consequence, to measure metrics such as its entropy (that measure the randomness of the string), its length and the length of the longest consecutive consonant sequence. Moreover, most of these features reflect real-world common practices such as the Search Engine Optimisation (SEO) and the netiquette. For example, SEO advice includes not only the ideal length of a domain name, which should be around 12–13 characters, but also suggestions like the reading easiness and the ability to spell the relay the FQDN to someone else.

The presented features in Table 8 can be assigned to two groups, on the one side classic string metrics such as the length (67% of cited works) (Schales et al. 2016; Ahluwalia et al. 2017; Antonakakis et al. 2012; Bisio et al. 2017; Han and Zhang 2017; Luo et al. 2017; Mowbray and Hagen 2014; Plohmann et al. 2016; Pu et al. 2015; Schiavoni et al. 2014; Shi et al. 2017; Song and Li 2016; Stevanovic et al. 2017; Tran et al. 2018; Truong and Cheng 2016; Xu et al. 2017), the number of vowels characters (17%) Schales et al. (2016), Ahluwalia et al. (2017), Song and Li (2016), Stevanovic et al. (2017) or the entropy (46%) Antonakakis et al. (2012), Bisio et al. (2017), Han and Zhang (2017), Luo et al. (2017), Plohmann et al. (2016), Pu et al. (2015), Shi et al. (2017),

[1] Including four features (`NLP-L-`x, `NLP-R-NUM-`x, `NLP-R-VOW-`x, `NLP-R-CON-`x) for each domain name level: the FQDN, the Second Level Domain Name (2LD) or all the others sub-levels as a whole (OLD).

Table 8 Context-Free features collected from the literature

Code	Description
(a) Context-Free features with their descriptions	
NLP-L-*x*	String length
NLP-LDN	Number of domain levels
NLP-R-NUM-*x*	Ratio of numerical characters
NLP-R-VOW-*x*	Ratio of vowel characters
NLP-R-CON-*x*	Ratio of consonants characters
NLP-LANG	Language hypothesis
NLP-LC-C	Longest consecutive cons. sequence
NLP-LC-V	Longest consecutive vowel sequence
NLP-LC-D	Longest consecutive number sequence
NLP-COV	Covariance matrix
NLP-R-MC	Ratio of meaningful characters[a]
NLP-LMS	Length of longest meaningful string[a]
NLP-WLU	Number of "word-like" units[a]
NLP-SQS	Domain squatting score[a]
NLP-LED	Levenshtein edit distance[a]
NLP-*n*G-FR	Frequency distribution (histogram)
NLP-*n*G-E	Entropy
NLP-*n*G-COV	Covariance[a]
NLP-*n*G-MEAN	Mean of frequencies
NLP-*n*G-MED	Median of frequencies
NLP-*n*G-VAR	Variance of frequencies
NLP-*n*G-STD	Standard deviation of frequencies
NLP-*n*G-PRO	Pronounceability score[a]
NLP-*n*G-NORM	Normality score
NLP-*n*G-PRT	Transition probability[a]
NLP-*n*G-PRA	Probability of appearance[a]
NLP-*n*G-PRI	Index probability[a]
NLP-*n*G-DST-KL	Kullback–Leiber divergence[a]
NLP-*n*G-DST-JI	Jaccard Index measure[a]
NLP-*n*G-DST-TH	Distance-threshold[a]
NLP-*n*G-DST-AF	Distance–avg. frequency[a]
NLP-*n*G-DST-AC	Distance–avg. count[a]

Used by	Code (NLP-)													
	L-*x*	LDN	R-NUM-*x*	R-VOW-*x*	R-CON-*x*	LANG	LC-C	LC-V	LC-D	COV	R-MC	LMS	WLU	SQS	LED
(b) Context-Free features and their usage in the literature															
Abbink and Doerr (2017)															
Ahluwalia et al. (2017)	✓		✓	✓	✓										
Antonakakis et al. (2012)	✓	✓													
Baruch and David (2018)															✓
Bilge et al. (2014)			✓									✓			
Bisio et al. (2017)	✓	✓													
Han and Zhang (2017)	✓		✓		✓		✓		✓						
Kintis et al. (2017)														✓	

Table 8 continued

Used by	Code (NLP-)												
	L-*x*	LDN	R-NUM-*x*	R-VOW-*x*	R-CON-*x*	LANG	LC-C	LC-V	LC-D	COV	R-MC	LMS	WLU	SQS	LED
Luo et al. (2017)	✓						✓								
Mac et al. (2017)															
Mowbray and Hagen (2014)	✓														
Plohmann et al. (2016)	✓														
Pu et al. (2015)	✓						✓	✓	✓						
Schales et al. (2016)	✓	✓		✓		✓									
Schiavoni et al. (2014)	✓		✓							✓	✓				
Shi et al. (2017)	✓						✓								

Used by	Code (NLP-)								
	*n*G-FR	*n*G-E	*n*G-COV	*n*G-MEAN	*n*G-MED	*n*G-VAR	*n*G-STD	*n*G-PRO	*n*G-NORM
(b) Context-Free features and their usage in the literature									
Abbink and Doerr (2017)									
Ahluwalia et al. (2017)									
Antonakakis et al. (2012)	✓	✓			✓	✓	✓		
Baruch and David (2018)									
Bilge et al. (2014)									
Bisio et al. (2017)		✓							
Han and Zhang (2017)		✓						✓	
Kintis et al. (2017)									
Luo et al. (2017)	✓	✓							
Mac et al. (2017)									✓
Mowbray and Hagen (2014)									
Plohmann et al. (2016)		✓							
Pu et al. (2015)	✓	✓							
Schales et al. (2016)									
Schiavoni et al. (2014)			✓	✓				✓	
Shi et al. (2017)		✓							
Song and Li (2016)		✓						✓	✓
Stevanovic et al. (2017)									
Thomas and Mohaisen (2014)									
Tong and Nguyen (2016)									✓
Tran et al. (2018)		✓							
Truong and Cheng (2016)		✓							
Xu et al. (2017)	✓	✓							
Yadav et al. (2010)									

Used by	Code (NLP-)							
	*n*G-PRT	*n*G-PRA	*n*G-PRI	*n*G-DST-KL	*n*G-DST-JI	*n*G-DST-TH	*n*G-DST-AF	*n*G-DST-AC
(b) Context-Free features and their usage in the literature								
Abbink and Doerr (2017)					✓			
Ahluwalia et al. (2017)								
Antonakakis et al. (2012)								
Baruch and David (2018)				✓	✓			
Bilge et al. (2014)								

Table 8 continued

Used by	Code (NLP-)							
	*n*G-PRT	*n*G-PRA	*n*G-PRI	*n*G-DST-KL	*n*G-DST-JI	*n*G-DST-TH	*n*G-DST-AF	*n*G-DST-AC
Bisio et al. (2017)								✓
Han and Zhang (2017)								
Kintis et al. (2017)								
Luo et al. (2017)	✓	✓	✓				✓	✓
Mac et al. (2017)								
Mowbray and Hagen (2014)								
Plohmann et al. (2016)								
Pu et al. (2015)				✓				
Schales et al. (2016)								
Schiavoni et al. (2014)						✓		
Shi et al. (2017)								
Song and Li (2016)								
Stevanovic et al. (2017)								
Thomas and Mohaisen (2014)					✓			
Tong and Nguyen (2016)								
Tran et al. (2018)								
Truong and Cheng (2016)		✓						
Xu et al. (2017)	✓							
Yadav et al. (2010)			✓		✓			

$x \in$ {FQDN, 2LD, OLD} denotes the domain levels [a] is about the English language $n \in [1, 2, 3]$ represents the n Gram size

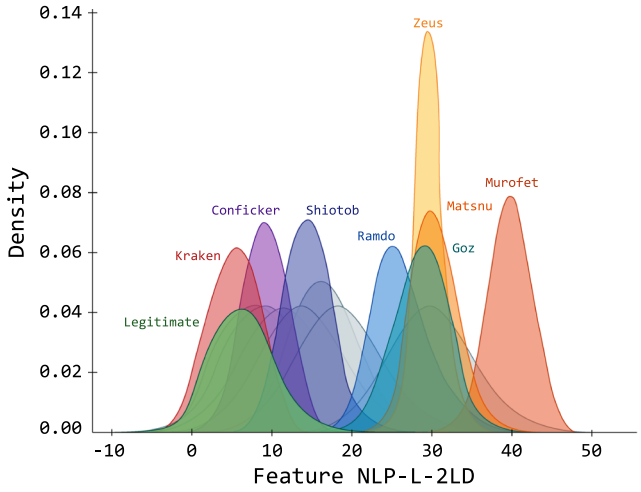
Song and Li (2016), Tran et al. (2018), Truong and Cheng (2016), Xu et al. (2017) shows attempts of solving the problem by using simpler but effective metrics; on the other side, features like the Jaccard Index measure (17%) (Abbink and Doerr 2017; Baruch and David 2018; Thomas and Mohaisen 2014; Yadav et al. 2010), the Kullback–Leiber divergence (0.08%) (Baruch and David 2018; Pu et al. 2015) or the probability of appearance (0.08%) (Luo et al. 2017; Truong and Cheng 2016) depict a deeper knowledge about the structure of the domain names and their usage from the linguistic point of view. Most importantly, however, is the fact that a standard pool of features is missing; that is, most of the presented solutions use arbitrary combinations of them, often with different names and unconventional mathematical definitions.

This work aims to homogenise the features by implementing them according to their theoretical definitions and common usage. Although with different formulations, authors like Ahluwalia et al. (2017) and Schales et al. (2016) have both proposed at a feature based on differentiating between vowels and consonants. Specifically, Ahluwalia et al. (2017) proposed to use the count of the occurrences of the consonants as feature, while Schales et al. (2016) proposed a Boolean flag that indicates whenever the domain name is composed only by consonants. Intuitively, the former is a discrete number whose value can range[2] from 3 to 255 characters that can be easily normalised with respect to the domain length. The latter, however, delineate a partial information by discarding data regarding the string composition. Both features are homogenised in our proposal by the NLP-R-CON-FQDN feature, which measures the ratio between the consonants and the length of the FQDN: on the one hand, it perfectly represents the number of occurrences normalised with respect to the length of the domain, while on the other hand not only includes the Boolean situation where the domain name is composed by all consonants but also provides additional information regarding the domain structure.

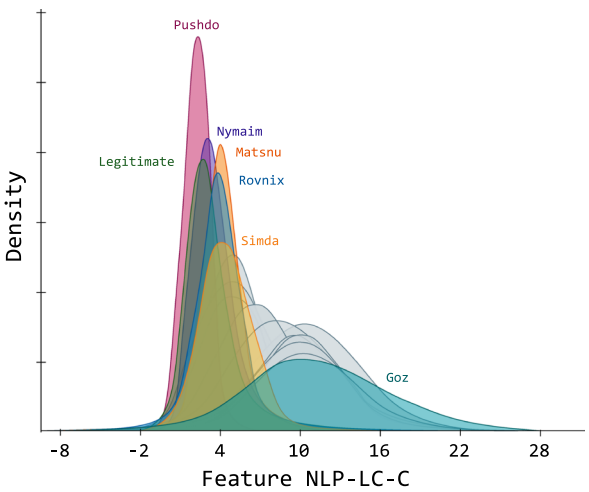
Likewise, a similar process has been completed for all the others features presented in Table 8; however, due to space and complexity concerns, such homogenisation process is not reported in this research item.

We included in Table 8b previous researches that are not strictly aligned with the DGA-based botnet detection, being either grouping client based on their connections (Schales et al. 2016; Yadav et al. 2010), applying filtering techniques (Abbink and Doerr 2017; Mowbray and Hagen

[2] According to ICANN specifics, the minimum length of a domain name without considering the Top Level Domain (TLD) is three characters. The maximum, including symbols and extensions, is 255, having a maximum length *per-level* of 63 characters.



(a) Distribution histogram of NLP-L-2LD.



(b) Distribution histogram of NLP-LC-C.

Fig. 1 Distribution histograms of two selected features from Table 8a

2014) or targeting a different problem (Kintis et al. 2017). Nonetheless, their feature definition and usage are clear and well used.

In order to realise a potential common data set of AGDs suitable for ML algorithms, the data are required to be normalised. To do so, each feature's histogram and relative distribution have been analysed and accordingly transformed to obtain, where possible, a normal distribution. Additionally, their graphical representations permit revealing some important aspects; for example, amongst all the features, we picked two normalised distributions from Table 8, namely `NLP-L-2LD` (length of the second level domain name) and `NLP-LC-C` (longest consecutive consonants sequence).

These features have been chosen as example to illustrate an important property that can be extracted from the distribution analysis, that is, a feature can be used to discriminate according to its value (Fig. 1a) or its shape (Fig. 1b). In fact, it is important to notice that malwares have different and well-defined distributions; for example, in the case of the feature `NLP-L-2LD` (Fig. 1a) the density histogram shows how different values of the feature permits to sort AGDs into their classes. The feature `NLP-LC-C`, however, presents a more overlapped distribution due to its nature (only a few malwares, such as Pushdo and Nymaim have an indistinguishable distribution with respect to the legitimate ones). Yet, its information is enough to separate them by exclusion from the simpler ones.

As for the features related to the *n*Grams, presented in Table 8, and similarly to the aforementioned habits, it is worth mentioning the pronounceability and normality scores, the transition and index probability and the different distances and divergences from the English language. These values are often calculated differently but having the same objective in mind, i.e. establishing "how random" are the domain names. However, they have been accordingly analysed, as illustrated, for example, for feature `NLP-R-CON-FQDN`, and reduced to a common formal definition. In order to analyse this feature

Table 9 Most informative features

Code	IG	IG ratio	Gini	χ^2
(a) Sorted by IG				
NLP-L-2LD	1.450	0.725	0.112	13296.997
NLP-L-FQDN	1.425	0.716	0.112	14143.591
NLP-1G-PRO	1.392	0.697	0.109	12441.981
NLP-2G-PRO	1.304	0.652	0.101	12291.421
NLP-1G-MEAN	1.000	0.500	0.081	8492.071
(b) Sorted by IG ratio				
NLP-R-NUM-OLD	0.306	0.821	0.044	1023.027
NLP-LC-D	0.306	0.820	0.044	13236.135
NLP-L-2LD	1.450	0.725	0.112	13296.997
NLP-L-FQDN	1.425	0.716	0.112	14143.591
NLP-R-NUM-FQDN	0.645	0.711	0.069	30694.025
(c) Sorted by Gini Coefficient				
NLP-L-2LD	1.450	0.725	0.112	13296.997
NLP-L-FQDN	1.425	0.716	0.112	14143.591
NLP-1G-PRO	1.392	0.697	0.109	12441.981
NLP-2G-PRO	1.304	0.652	0.101	12291.421
NLP-1G-MEAN	1.000	0.500	0.081	8492.071
(d) Sorted by χ^2				
NLP-R-NUM-FQDN	0.645	0.711	0.069	30694.025
NLP-R-NUM-2LD	0.634	0.699	0.065	30031.921
NLP-L-FQDN	1.425	0.716	0.112	14143.591
NLP-L-2LD	1.450	0.725	0.112	13296.997
NLP-LC-D	0.306	0.820	0.044	13236.135

Table 10 Correlation matrix of the most informative features

Features (NLP-)	R-NUM-FQDN	L-FQDN	L-2LD	LC-D	1G-PRO	2G-PRO	1G-MEAN	R-NUM-OLD	R-NUM-2LD
R-NUM-FQDN	–	0.51	0.51	−0.11	−0.14	0.04	−0.04	0.02	0.99
L-FQDN	0.51	–	0.96	−0.1	−0.81	−0.59	−0.16	−0.01	0.47
L-2LD	0.51	0.96	–	−0.36	−0.76	−0.55	−0.14	0.00	0.47
LC-D	−0.11	−0.10	−0.36	–	0.03	0.03	−0.01	0.00	−0.11
1G-PRO	−0.14	−0.81	−0.76	0.03	–	0.86	0.21	0.03	−0.08
2G-PRO	0.04	−0.59	−0.55	0.03	0.86	–	0.19	0.05	0.08
1G-MEAN	−0.04	−0.16	−0.14	−0.01	0.21	0.19	–	−0.01	−0.02
R-NUM-OLD	0.02	−0.01	0.00	0.00	0.03	0.05	−0.01	–	0.00
R-NUM-2LD	0.99	0.47	0.47	−0.11	−0.08	0.08	−0.02	0.00	–

set, we extracted the five most relevant features according to four different algorithms, reported in Table 9; namely:

- *Information Gain (IG)*, sorted in Table 9a. It gives a score regarding "how much" information is the feature bringing with respect to the classification target.[3] Substantially, it reflects the amount of information that was available before and after considering the feature.
- *IG Ratio*, sorted in Table 9b. Similar in concept to the IG, this score measures the ratio between the information provided by the actual label and the one provided by the feature. It compensates for high-entropy features, which are normally advantaged by the simple IG, score.
- *Gini Coefficient*, sorted in Table 9c. This metric measures the "purity" of the feature with regards to the actual label. The greater the value is, the better the feature is.
- *Chi-Square* (χ^2), sorted in Table 9d. This statistical score measures the independence (specifically its lack) between the feature and the actual label compared with the χ^2 distribution with one degree of freedom. The greater the value is, the more category information the feature contains.

As the features' names suggest, the length of the domain as a whole and the length of the second level domain name might be strictly correlated, as well as the number ratio for the second and the FQDN. We thus extracted the correlation matrix, presented in Table 10, which confirms our hypothesis by identifying a strong correlation between two different pairs of features.

Generally speaking, having highly correlated features affects the model performances, especially in supervised learning. However, to be more precise, both the improvement and the deterioration in model performances are questionable and require further investigations. Removing them may be the correct approach, but also may degrade the model:

- *Performance improvement*. Especially in terms of training speed, due to the well-known problem of the "Curse of Dimensionality".
- *Bias decrease*. As a rule of thumb, features that present low values of mutual correlation or multicollinearity, to the target are helpful and generally speaking they should be kept in the feature set.
- *Interpretability*. Fewer features lead to a model that is easier to justify and explain, thus it may worth to remove them, possibly paying in terms of precision and recall.

Within the context of our data set and the aforementioned applicability scenario, we decided to discard the feature NLP-L-2LD in favour of NLP-L-FQDN since former is stripped from the extensions, thus losing information with respect to the latter. Likewise, we discard the NLP-R-NUM-FQDN in favour of NLP-R-NUM-2LD due to the ICANN rules that prevent numerical characters from appearing in the extension part.

Furthermore, feature engineering algorithm such as the PCA may provide another hint towards a more suitable feature set. Specifically, the PCA decorrelates the data by mapping it to n orthogonal dimensions that maximise the variance and minimise the correlation between them. In Fig. 2, it is highlighted the PCA configuration to cover 99% of the variance of the data, where the upper line represents the cumulative variance covered, while the lower one represents the variance covered per component. However, knowing the structure and the complexity of the data analysed, the PCA may not represent the most suitable data transformation to improve the learning models performances. In fact, by observing Fig. 2, one could argue that the PCA is not producing the desired effects (i.e. reducing the number of features without losing too much information) since most of the principal components are not so informative.

In this context, and as demonstrated by the following evaluation Sect.3.3, blindly maximising the variance amongst the data leads to worse performances. This phenomena suggest

[3] The IG, is purely theoretic, it does not consider any particular classification algorithm.

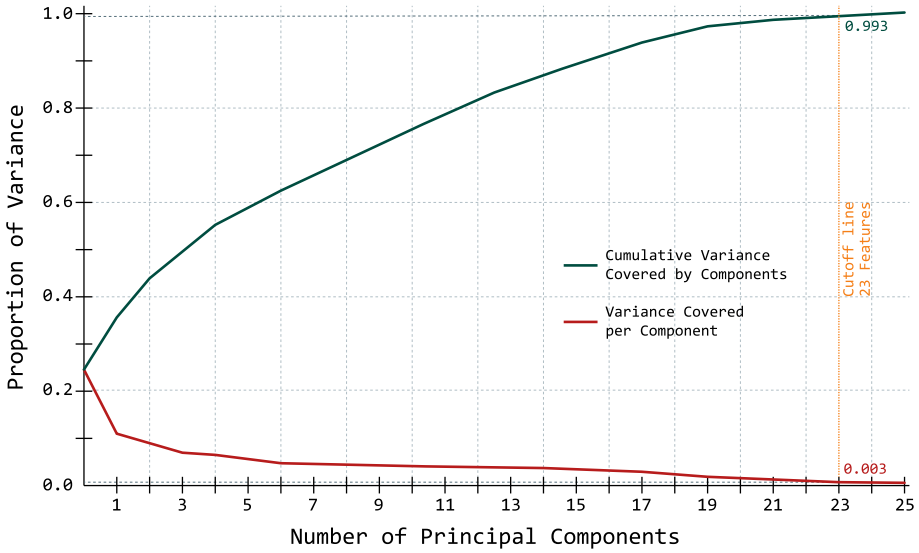
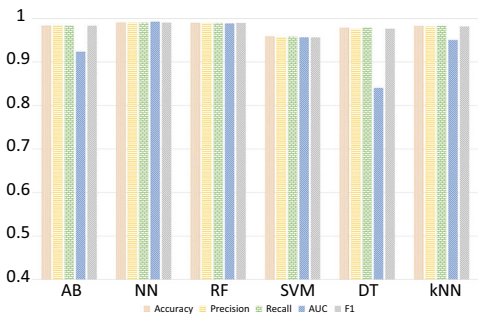
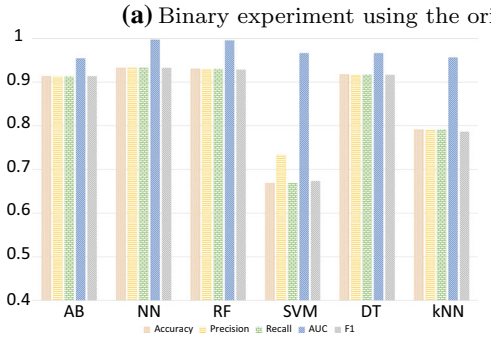


Fig. 2 PCA configured to cover 99% of the data variance



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.983	0.983	0.983	0.923	0.983
NN	0.990	0.990	0.990	0.992	0.990
RF	0.989	0.989	0.989	0.988	0.989
SVM	0.958	0.955	0.958	0.956	0.956
DT	0.978	0.976	0.978	0.840	0.976
kNN	0.982	0.981	0.982	0.950	0.981

(a) Binary experiment using the original feature set. (Exp. 1, U.C. 1)



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.913	0.913	0.913	0.954	0.913
NN	0.932	0.932	0.932	0.997	0.932
RF	0.930	0.928	0.930	0.995	0.928
SVM	0.668	0.733	0.668	0.966	0.673
DT	0.917	0.916	0.917	0.966	0.916
kNN	0.791	0.790	0.791	0.956	0.786

(b) Multiclass experiment using the original feature set. (Exp. 2, U.C. 1)

Fig. 3 Evaluation using the original feature set (U.C. 1)

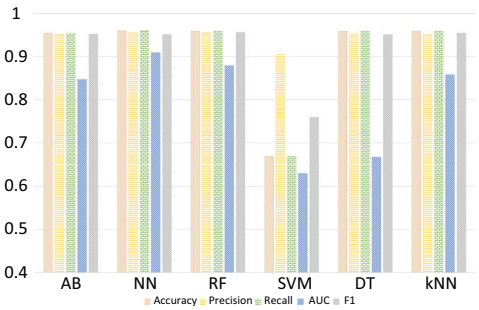
that an exploratory analysis regarding the nonlinear dimensionality reduction techniques may be indeed necessary.

3.3 Evaluation

In order to compare and confirm the findings of Sects. 2 and 3.2, we designed a series of experiments. These experiments were conducted on a Dell M3800 workstation with an Intel i7-4712HQ processor running at 2.30GHz, 16 GB of DDR3 RAM at 1600 MHz and a NVIDIA Quadro K1100M graphic processor. Experiments were run using Orange3 (Demšar et al. 2013), and for each set of data described in the aforementioned methodology Sect. 3.1, six classifiers were applied and cross-validated using a stratified tenfold approach. On the one hand, the six algorithms used are defined using the following configuration:

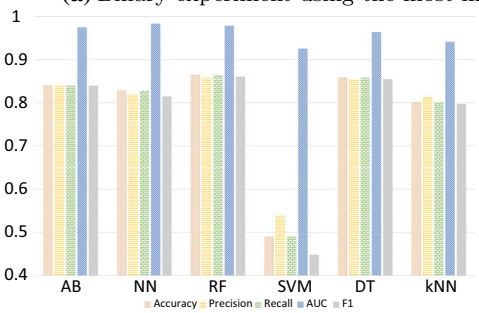
- *AdaBoost (AB)*—Using 50 trees as base estimators, with SAMME.R classificator (updates base estimators weight with probability estimates) and linear regression loss function.
- *Neural Network (NN)*—Single hidden layer with 100 nodes activated with the Rectified Linear unit (ReLu) function, weight optimised with the stochastic gradient-

Fig. 4 Evaluation using the most informative features (U.C. 2)



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.954	0.951	0.954	0.847	0.952
NN	0.960	0.956	0.960	0.909	0.951
RF	0.959	0.955	0.959	0.879	0.956
SVM	0.669	0.905	0.669	0.630	0.759
DT	0.959	0.953	0.959	0.667	0.951
kNN	0.959	0.953	0.959	0.858	0.954

(a) Binary experiment using the most informative feature set. (Exp. 1, U.C. 2)



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.840	0.839	0.840	0.974	0.839
NN	0.828	0.819	0.828	0.983	0.814
RF	0.864	0.859	0.864	0.978	0.860
SVM	0.490	0.538	0.490	0.925	0.448
DT	0.858	0.853	0.858	0.963	0.854
kNN	0.800	0.812	0.800	0.941	0.797

(b) Multiclass experiment using the most informative feature set. (Exp. 2, U.C. 2)

based optimiser (Adam), $\alpha = 0.0010$ and 200 max iterations.

- *Random Forest (RF)*—Using 10 trees, considering up to five attributes at each split and without splitting subsets smaller than five.
- *Support Vector Machine (SVM)*—Configured with $C = 1.00$, $\epsilon = 0.10$ and using the RBF Kernel.
- *Decision Tree (DT)*—Two minimum instances in leaves, do not split trees smaller than five, having a max depth of 100. Exiting condition when the majority reaches 95%.
- *k-Nearest Neighbours (kNN)*—Five neighbours using the Euclidean metric and a uniform weight.

While, on the other hand, the tables reported in Figs. 3, 4 and 5 are reporting the average values over the tested classes with a testing set obtained by cross-validation sampling (having 10 stratified folds).

Section 3.3.1 introduces the three evaluation use cases which are later discussed and commented in Sect. 3.3.2.

3.3.1 Experiments design and use cases

Two sets of experiments were run on with the same data set and the same configuration. In the first one, the malware families (reported in Sect. 3.1) are used as separate classes, while in the second one the malwares are considered as a single class, enabling the binary analysis of malware versus legitimate domain names. To be more specific:

Experiment 1 ***(Binary)*** *The binary experiment is designed to answer the ML question of separating legitimate FQDNs from malicious AGDs, considering all malware families as a single category.*

Experiment 2 ***(Multiclass)*** *The multiclass experiment is designed to go beyond the above-mentioned binary experiment in order to classify not only the legitimate FQDN but also sort malware samples according to their families.*

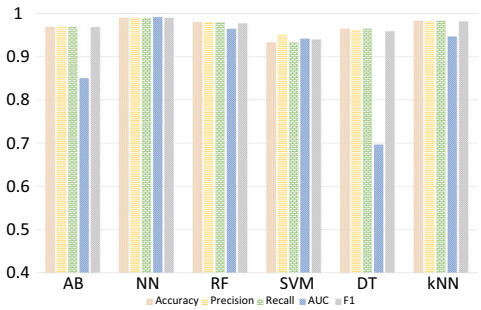
Moreover, in order to present the experiments results in combination with the aforementioned feature sets, three use cases are introduced and commented below.

Use Case 1 ***(Original features set)*** *In the first use case, both experiments were run using the original data set with the all the collected features, as presented in Table 8. The results are reported in Fig. 3. Specifically, it is possible to compare the results of the binary experiment and the multiclass one in Fig. 3a and b, respectively.*

Use Case 2 ***(Most informative features set)*** *Similarly, in the second use case the experiments were run using the data set updated with the features selected in the previous Sect. 3.2 and presented in Fig. 4, also divided into binary (Fig. 4a) and multiclass (Fig. 4b) experiments.*

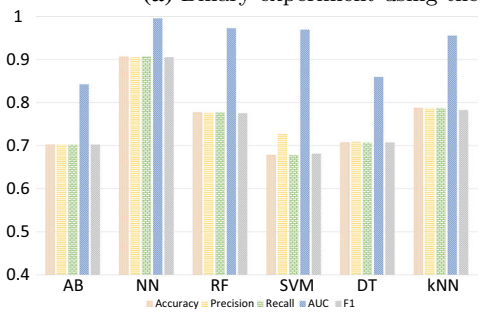
Use Case 3 ***(PCA features set)*** *Finally, and as reported above, the third use case features the experiments run using the data set manipulated with the PCA algorithm. Likewise, Fig. 5 presents the comparison tables and the relative charts for binary (Fig. 5a) and multiclass (Fig. 5b) experiments.*

Fig. 5 Evaluation using the PCA (U.C. 3)



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.968	0.968	0.968	0.850	0.968
NN	0.989	0.989	0.989	0.991	0.989
RF	0.979	0.978	0.979	0.964	0.977
SVM	0.932	0.950	0.932	0.941	0.939
DT	0.964	0.961	0.964	0.696	0.958
kNN	0.982	0.981	0.982	0.946	0.981

(a) Binary experiment using the PCA feature set. (Exp. 1, U.C. 3)



Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.702	0.702	0.702	0.842	0.702
NN	0.906	0.905	0.906	0.995	0.905
RF	0.777	0.775	0.777	0.972	0.775
SVM	0.678	0.726	0.678	0.969	0.681
DT	0.707	0.708	0.707	0.859	0.707
kNN	0.787	0.787	0.787	0.955	0.782

(b) Multiclass experiment using the PCA feature set. (Exp. 2, U.C. 3)

3.3.2 Discussion

When looking at the multiclass experiments (Exp. 2), it is important to restate that the values reported in Figs. 3b, 4b and 5b are referring to the average value calculated over the classes-specific values. As a consequence, values such as the F1 score are not to be considered inconsistent; that is, the average of all the F1 scores over the classes might be outside the range delimited by the average of the precision and recall scores. The same applies to the AUC results, a value that represents the discrimination, i.e. the ability of the specific classification algorithm to sort the member of the target class. Not surprisingly, by having 17 classes (16 malware families plus the legitimate one, see Sect. 3.1) the average value of the AUC is quite high because of the excellent performances with regard to some easily identifiable malwares.

As expected, the multiclass experiments are presenting worse performances than the binary ones due to the poor performances obtained when distinguishing similar malwares such as Qakbot or Matsnu. In the binary experiments, however, those samples are considered as a generic malware class, thus improving the overall detection. Similarly, the performances of the PCA use case (U.C. 3) are worse than the ones reported in the original use case (U.C. 1). This worsening trend is mainly due to the nature of the PCA itself, i.e. maximise the variability of the features, but not the classes' separability. Although very common and widely used, the PCA is an example of unsupervised analysis tool that is not suitable in this scenario, thus leading in general to worse performances. Further researches are required in order to establish which feature reduction strategy optimally approximates the data. Preliminary results using nonlinear feature reduction techniques seem promising.

In contrast with the expected results of the previous use cases, the feature selection use case (U.C. 2) reports unanticipated slightly worse performances. However, the theory of ML permits to explain such results, especially when comparing them to the ones described in the literature (outlined in Sect. 2). First and foremost, the data set used plays an important role; that is, by accurately selecting the data involved in the analysis, it is possible to boost the accuracy and generally increase the performances. Secondly, different algorithms require different optimal structures for the data, e.g. RF are indifferent to scaling and normalisation techniques, while NN require normal and scaled distributions. Last, but not least, fine-tuning each algorithm permits to extend further their efficiency (Mantovani et al. 2015).

To sum up, this evaluation process once again pointed out the lack of common elements that can be used so as to guarantee the reproducibility of the experiments and thus their comparative analysis. To this extent, a publicly available data set of AGDs may lead to better, but more importantly, comparable results.

4 Conclusions and future work

Although DGA-based botnets have been proved challenging, there are some positive results within the research field

that looks promising. In this context, in fact, ML approaches have been successfully employed both in a supervised and unsupervised manner aiming to achieve usable and scalable solutions.

After presenting the state of the art in terms of both algorithms and feature set, this work has defined and established a common-ground knowledge regarding the features used as a base for such algorithms, showing that the Context-Free feature family is more than capable of pinpointing DGA-based malwares without harming the users' privacy. Besides what is discussed in Sect. 3.3.2, the sole use of Context-Free features achieves excellent results in terms of performances, precision and recall both in the Binary and in the Multiclass scenarios.

Therefore, the main outcomes of this work are several. In fact, this research firstly collects and studies the most relevant literature works that attempted to deal with the DGA-based botnet detection problem; secondly, this research item enables the development of privacy-preserving solutions[4]; and, finally, by collecting and studying the literature and the proposed features, it prearranges a common ground that enables future researches to focus on the evaluation and the creation of new efficient detection algorithms in the subject of DGA-based botnet. To this extent, in fact, preliminary analyses carried out internally indicate positive and interesting results.

Therefore, to summarise and complement what is proposed in Sect. 2.3, future works might include (i) the study of the Context-Aware feature family to establish whether it may combine and consolidate detection solutions; (ii) the development of a full-fledged, public available and labelled data set of either Context-Aware and Context-Free features that might emerge as a common ground for the evaluation of existing and new ML solutions; (iii) the exploratory analysis of the above-mentioned data with nonlinear techniques in order to achieve improved classification results; and, finally, (iv) testing detection algorithms against 0-day resilience by adding both new malware families and variants.

Acknowledgements This study was founded by a predoctoral and a postdoctoral INCIBE Grant within the "Ayudas para la Excelencia de los Equipos de Investigación Avanzada en Ciberseguridad" program, with Codes INCIBEI-2015-27353 and INCIBEI-2015-27352.

Compliance with ethical standards

Conflict of interest The authors declare that they do not have any conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

[4] By experimentally demonstrating that users' data are not strictly required to recognise malwares in the wild. See Sect. 3.3.

References

Abakumov A (2016) andrewaeva/DGA. URL https://github.com/andrewaeva/DGA

Abbink J, Doerr C (2017) Popularity-based detection of domain generation algorithms. In: 12th international conference on availability, reliability and security, pp 79:1–79:8. https://doi.org/10.1145/3098954.3107008

Abdel-Hamid O, Mohamed Ar, Jiang H, Deng L, Penn G, Yu D (2014) Convolutional neural networks for speech recognition. IEEE/ACM Trans Audio Speech Lang Process 22(10):1533–1545. https://doi.org/10.1109/TASLP.2014.2339736

Ahluwalia A, Traore I, Ganame K, Agarwal N (2017) Detecting broad length algorithmically generated domains. In: Intelligent, secure, and dependable systems in distributed and cloud environments, chap. 2, pp 19–34. Springer International Publishing. https://doi.org/10.1007/978-3-319-69155-8_2

Alieyan K, ALmomani A, Manasrah A, Kadhum MM (2017) A survey of botnet detection based on DNS. Neural Comput Appl 28(7):1541–1558. https://doi.org/10.1007/s00521-015-2128-0

Almomani A, Alauthman M, Albalas F, Dorgham O, Obeidat A (2018) An online intrusion detection system to cloud computing based on Neucube algorithms. Int J Cloud Appl Comput 8(2):96–112. https://doi.org/10.4018/IJCAC.2018040105

Anderson HS, Woodbridge J, Filar B (2016) DeepDGA: adversarially-tuned domain generation and detection. In: 2016 ACM workshop on artificial intelligence and security, pp 13–21. https://doi.org/10.1145/2996758.2996767

Antonakakis M, Perdisci R, Nadji Y, Vasiloglou N, Abu-Nimeh S, Lee W, Dagon D (2012) From throw-away traffic to bots: detecting the rise of DGA-based malware. In: 21st USENIX security symposium, pp 491–506. Bellevue, WA. URL https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/antonakakis

Bader J. Domain Generation Algorithms. URL https://github.com/baderj/domain_generation_algorithms

Baruch M, David G (2018) Domain generation algorithm detection using machine learning methods. In: Cyber security: power and technology, pp 133–161. Springer International Publishing. https://doi.org/10.1007/978-3-319-75307-2_9

Berger A, Gansterer WN (2013) Modeling DNS agility with DNSMap. In: 2013 proceedings IEEE INFOCOM, pp 3153–3158. https://doi.org/10.1109/INFCOM.2013.6567130

Biglar Beigi E, Hadian Jazi H, Stakhanova N, Ghorbani AA (2014) Towards effective feature selection in machine learning-based botnet detection approaches. In: 2014 IEEE conference on communications and network security, pp 247–25. https://doi.org/10.1109/CNS.2014.6997492

Bilge L, Sen S, Balzarotti D, Kirda E, Kruegel C (2014) Exposure: a passive DNS analysis service to detect and report malicious domains. ACM Trans Inf Syst Secur 16(4):14:1–14:28. https://doi.org/10.1145/2584679

Bishop C (2006) Pattern recognition and machine learning. Springer, Berlin

Bisio F, Saeli S, Lombardo P, Bernardi D, Perotti A, Massa D (2017) Real-time behavioral DGA detection through machine learning. In: 2017 international carnahan conference on security technology, pp 1–6. https://doi.org/10.1109/CCST.2017.8167790

Bugiel S, Nürnberger S, Pöppelmann T, Sadeghi AR, Schneider T (2011) AmazonIA: when elasticity snaps back. In: 18th ACM conference on computer and communications security, pp 389–400. https://doi.org/10.1145/2046707.2046753

Demšar J, Curk T, Erjavec A, Gorup Č, Hočevar T, Milutinovič M, Možina M, Polajnar M, Toplak M, Starič A, Štajdohar M, Umek L, Žagar L, Žbontar J, Žitnik M, Zupan B (2013) Orange: data

mining toolbox in Python. J Mach Learn Res 14:2349–2353. URL http://jmlr.org/papers/v14/demsar13a.html
Fran E, Hall MA, Witten IH (2016) The WEKA Workbench. Tech. rep. URL https://www.cs.waikato.ac.nz/ml/weka
Fu Y, Yu L, Hambolu O, Ozcelik I, Husain B, Sun J, Sapra K, Du D, Beasley CT, Brooks RR (2017) Stealthy domain generation algorithms. IEEE Trans Inf Forensics Secur 12(6):1430–1443. https://doi.org/10.1109/TIFS.2017.2668361
García S, Grill M, Stiborek J, Zunino A (2014) An empirical comparison of botnet detection methods. Comput Secur 45:100–123. https://doi.org/10.1016/j.cose.2014.05.011
Grill M, Nikolaev I, Valeros V, Rehak M (2015) Detecting DGA malware using NetFlow. In: 2015 IFIP/IEEE international symposium on integrated network management, pp 1304–1309. https://doi.org/10.1109/INM.2015.7140486
Gupta B, Agrawal DP, Yamaguchi S (eds) (2016) Handbook of research on modern cryptographic solutions for computer and cyber security, 1st edn. IGI Global
Han C, Zhang Y (2017) CODDULM: an approach for detecting C&C domains of DGA on passive DNS traffic. In: 2017 6th international conference on computer science and network technology, pp 385–388. https://doi.org/10.1109/ICCSNT.2017.8343724
Haykin S (1998) Neural networks: a comprehensive foundation, 2nd edn. Prentice Hall PTR, Upper Saddle River
Holz T, Steiner M, Dahl F, Biersack E, Freiling F (2008) Measurements and mitigation of peer-to-peer-based Botnets: a case study on storm worm. In: USENIX security 2008. URL https://www.usenix.org/conference/leet-08/measurements-and-mitigation-peer-peer-based-botnets-case-study-storm-worm
Hussain SA, Fatima M, Saeed A, Raza I, Shahzad RK (2017) Multilevel classification of security concerns in cloud computing. Appl Comput Inform 13(1):57–65. https://doi.org/10.1016/j.aci.2016.03.001
Kintis P, Miramirkhani N, Lever C, Chen Y, Romero-Gómez R, Pitropakis N, Nikiforakis N, Antonakakis M (2017) Hiding in plain sight: a longitudinal study of combosquatting abuse. In: ACM SIGSAC conference on computer and communications security, pp 569–586. https://doi.org/10.1145/3133956.3134002
Kührer M, Rossow C, Holz T (2014) Paint it black: evaluating the effectiveness of malware blacklists. In: RAID 2014: research in attacks, intrusions and defenses, June, pp 1–21. Springer International Publishing. https://doi.org/10.1007/978-3-319-11379-1_1
Leelasankar K, Chellappan C, Sivasankar P (2018) Handbook of research on network forensics and analysis techniques, chap. successful computer forensics analysis on the cyber attack Botnet, pp 266–281. IGI Global. https://doi.org/10.4018/978-1-5225-4100-4.ch014
Lerner Z (2014) Microsoft the Botnet hunter: the role of public-private partnerships in mitigating Botnets. Harvard J Law Technol 28(1):237–261. URL http://jolt.law.harvard.edu/articles/pdf/v28/28HarvJLTech237.pdf
Lobato AGP, Lopez MA, Sanz IJ, Cardenas AA, Duarte OCMB, Pujolle G (2018) An Adaptive real-time architecture for zero-day threat detection. In: 2018 IEEE international conference on communications (ICC), pp 1–6. https://doi.org/10.1109/ICC.2018.8422622
Luo X, Wang L, Xu Z, Yang J, Sun M, Wang J (2017) DGASensor: fast detection for DGA-based malwares. In: 5th international conference on communications and broadband networking, pp 47–53. https://doi.org/10.1145/3057109.3057112
Mac H, Tran D, Tong V, Nguyen LG, Tran HA (2017) DGA Botnet detection using supervised learning methods. In: 8th international symposium on information and communication technology, pp 211–218. https://doi.org/10.1145/3155133.3155166
Majestic-12 Ltd: The Majestic Million (2018) URL https://majestic.com/reports/majestic-million
Malware Domain List (2009) URL https://www.malwaredomainlist.com/mdl.php
Mantovani RG, Rossi AL, Vanschoren J, Bischl B, Carvalho AC (2015) To tune or not to tune: recommending when to adjust SVM hyper-parameters via meta-learning. In: Proceedings of the international joint conference on neural networks, vol 2015-September, pp 1–8. https://doi.org/10.1109/IJCNN.2015.7280644
Mell P, Grance T (2011) The NIST definition of cloud computing, NIST Special Publication 800-145. URL http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf
Mowbray M, Hagen J (2014) Finding domain-generation algorithms by looking at length distribution. In: 2014 IEEE international symposium on software reliability engineering workshops, pp 395–400. https://doi.org/10.1109/ISSREW.2014.20
Nespoli P, Papamartzivanos D, Mrmol FG, Kambourakis G (2018) Optimal countermeasures selection against cyber attacks: a comprehensive survey on reaction frameworks. IEEE Commun Surv Tutor 20(2):1361–1396. https://doi.org/10.1109/COMST.2017.2781126
Netlab 360: DGA Families. URL http://data.netlab.360.com/dga/
Nguyen TD, Cao TD, Nguyen LG (2015) DGA Botnet detection using collaborative filtering and density-based clustering. In: 6th international symposium on information and communication technology, pp 203–209. https://doi.org/10.1145/2833258.2833310
OSINT: OSINT DGA List. URL http://osint.bambenekconsulting.com/feeds/
Pelleg D, Moore A (2000) X-means: Extending K-Means with efficient estimation of the number of clusters. In: 7th international conference on machine learning pp 727–734. https://doi.org/10.1007/3-540-44491-2_3
Plohmann D (2015) DGArchive. URL https://dgarchive.caad.fkie.fraunhofer.de
Plohmann D, Yakdan K, Klatt M, Bader J, Gerhards-Padilla E (2016) A comprehensive measurement study of domain generating malware. In: 25th USENIX security symposium, pp 263–278. Austin, TX. URL https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_plohmann.pdf
Pu Y, Chen X, Pu Y, Shi J (2015) A clustering approach for detecting auto-generated Botnet domains. In: Applications and techniques in information security, pp 269–279. https://doi.org/10.1007/978-3-662-48683-2_24
Risk Analytics: DNS-BH-Malware Domain Blocklist (2007). URL http://www.malwaredomains.com
Schales DL, Jang J, Wang T, Hu X, Kirat D, Wuest B, Stoecklin MP (2016) Scalable analytics to detect DNS misuse for establishing stealthy communication channels. IBM J Res Dev 60(4):3:1–3:14. https://doi.org/10.1147/JRD.2016.2557639
Schiavoni S, Maggi F, Cavallaro L, Zanero S (2014) Phoenix: DGA-based Botnet tracking and intelligence. In: 11th international conference on detection of intrusions and malware, and vulnerability assessment, pp 192–211. Springer International Publishing. https://doi.org/10.1007/978-3-319-08509-8_11
Sharieh A, Albdour L (2017) A heuristic approach for service allocation in cloud computing. Int J Cloud Appl Comput 7(4):60–74. https://doi.org/10.4018/IJCAC.2017100104
Shi Y, Chen G, Li J (2017) Malicious domain name detection based on extreme machine learning. Neural Process Lett. https://doi.org/10.1007/s11063-017-9666-7
Song WJ, Li B (2016) A method to detect machine generated domain names based on random forest algorithm. In: 2016 international conference on information system and artificial intelligence, pp 509–513. https://doi.org/10.1109/ISAI.2016.0114
Stergiou C, Psannis KE, Kim BG, Gupta B (2018) Secure integration of IoT and cloud computing. Future Gener Comput Syst 78(3):964–975. https://doi.org/10.1016/j.future.2016.11.031

Stevanovic M, Pedersen JM, D'Alconzo A, Ruehrup S, Berger A (2015) On the ground truth problem of malicious DNS traffic analysis. Comput Secur 55:142–158. https://doi.org/10.1016/j.cose.2015.09.004

Stevanovic M, Pedersen JM, D'Alconzo A, Ruehrup S (2017) A method for identifying compromised clients based on DNS traffic analysis. Int J Inf Secur 16(2):115–132. https://doi.org/10.1007/s10207-016-0331-3

Thomas M, Mohaisen A (2014) Kindred domains: detecting and clustering Botnet domains using DNS traffic. In: 23rd international conference on World Wide Web, pp 707–712. https://doi.org/10.1145/2567948.2579359

Tong V, Nguyen G (2016) A method for detecting DGA Botnet based on semantic and cluster analysis. In: 7th symposium on information and communication technology, pp 272–277. https://doi.org/10.1145/3011077.3011112

Tran D, Mac H, Tong V, Tran HA, Nguyen LG (2018) A LSTM based framework for handling multiclass imbalance in DGA Botnet detection. Neurocomputing 275:2401–2413. https://doi.org/10.1016/j.neucom.2017.11.018

Truong D, Cheng G (2016) Detecting domain-flux botnet based on DNS traffic features in managed network. Secur Commun Netw 9(14):2338–2347. https://doi.org/10.1002/sec.1495

Tu TD, Guang C, Xin LY (2015) Detecting Bot-infected machines based on analyzing the similar periodic DNS queries. In: 2015 international conference on communications, management and telecommunications, pp 35–40. https://doi.org/10.1109/ComManTel.2015.7394256

Vinayakumar R, Soman K, Poornachandran P, Sachin Kumar S (2018) Evaluating deep learning approaches to characterize and classify the DGAs at scale. J Intell Fuzzy Syst 34(3):1265–1276. https://doi.org/10.3233/JIFS-169423

Vormayr G, Zseby T, Fabini J (2017) Botnet communication patterns. IEEE Commun Surv Tutor 19(4):2768–2796. https://doi.org/10.1109/COMST.2017.2749442

Watkins L, Beck S, Zook J, Buczak A, Chavis J, Robinson WH, Morales JA, Mishra S (2017) Using semi-supervised machine learning to address the big data problem in DNS networks. In: 2017 IEEE 7th annual computing and communication workshop and conference, pp 1–6. https://doi.org/10.1109/CCWC.2017.7868376

Woodbridge J, Anderson HS, Ahuja A, Grant D (2016) Predicting domain generation algorithms with long short-term memory networks. CoRR **abs/1611.0**. URL http://arxiv.org/abs/1611.00791

Xu S, Li S, Meng K, Wu L, Ding M (2017) An adaptive malicious domain detection mechanism with DNS traffic. In: 2017 VI international conference on network, communication and computing, pp 86–91. https://doi.org/10.1145/3171592.3171595

Yadav S, Reddy AKK, Reddy ALN, Ranjan S (2010) Detecting algorithmically generated malicious domain names. In: 10th ACM SIGCOMM conference on internet measurement, pp 48–61. https://doi.org/10.1145/1879141.1879148

Zhang S, Zhang X, Ou X (2014) After we knew it: empirical study and modeling of cost-effectiveness of exploiting prevalent known vulnerabilities across IaaS cloud. In: 9th ACM symposium on information, computer and communications security, pp 317–328. https://doi.org/10.1145/2590296.2590300

Zhang H, Gharaibeh M, Thanasoulas S, Papadopoulos C (2016) BotDigger: detecting DGA Bots in a single network. Tech. rep., Colorado State University. URL http://www.cs.colostate.edu/TechReports/Reports/2016/tr16-101.pdf