UMUDGA: a dataset for profiling DGA-based botnet

Mattia Zago[a,*], Manuel Gil Pérez[a], Gregorio Martínez Pérez[a]

[a]*Department of Information Engineering and Communications, University of Murcia, Campus Espinardo s/n 30100 Murcia (Spain)*

Abstract

Advanced botnet threats are natively deploying concealing techniques to prevent detection and sinkholing. To tackle them, machine learning solutions have become a standard approach, especially when dealing with Algorithmically Generated Domain (AGD) names. Nevertheless, machine learning state-of-the-art is non-specialist at best, having multiple issues in terms of rigorousness, reproducibility and ultimately credibility. This research focuses on the first critical step of the training phase, that is, the collection of data suitable for being analysed by algorithms. We have detected a common lack of scientific rigorousness in the literature regarding the aforementioned AGD analysis and, therefore, we advocate two major contributions in this article: *i)* a thorough analysis of the cyber panorama in terms of botnets that make use of Domain Generation Algorithms (DGAs) as evasive techniques, that flows into *ii)* a full-fledged machine-learning-ready labelled dataset that features over 30 million AGDs sorted in 50 malware variant classes. This mature dataset aims to fill the gap in the comparability between the different researches published in the literature. Lastly, two minor contributions are also included in this article: *iii)* we designed an exploratory analysis of the proposed dataset to provide both data characteristics and potential future research lines, which eventually emerges as *iv)* a collection of suggested guidelines. When proposing a machine learning solution, researchers should adhere to it in order to achieve scientific rigorousness.

Keywords: Domain Generation Algorithm (DGA), Natural Language Processing (NLP), Machine Learning, Botnet, Network Security

1. Introduction

The 2019 cybersecurity landscape is seriously perilous, in fact, as for the past few years, several technical reports from major security stakeholders [1, 2, 3, 4, 5, 6, 7] have confirmed that cybercriminal activities are rising in almost every sector. Although 2018 has seen a decrease in the number of malware variants available in exploit kits (-63% according to [1]), the threat posed by botnets is increasing [2, 5, 6], especially when considering the ones with backdoor functionalities (+34% in private *vs* +173% in enterprises [2]). One could argue that the reason behind such "positive result" (from the cybercriminals standpoint) is due to the extensive usage of well-known evasion techniques such as obfuscation, live encryption and Domain Generation Algorithm (DGA) [8, 9, 10]. These techniques are employed by almost every major malware in the wild in order to bypass Intrusion Detection Systems (IDSs) inspection measures since the infamous Kraken and Conficker malwares, back in 2008. To be more precise, a DGA is a technique that makes use of pseudo-random routines and external factors (such as time, data feeds, etc. [9]) to generate multiple Fully Qualified Domain Names (FQDNs) to use as *rendezvous-point* (*i.e.*, algorithmically generated domains, or *AGDs*) for the botnets' Command & Control (C&C) servers [10]. DGAs are a notably effective evasion technique that consists of generating thousands, often millions, of pseudo-random domain names. Their strength relies in the asymmetry of resource required by the attacker(s) (*i.e.*, the botmaster(s), the malicious actor(s) in control of the C&C servers and thus in control of the botnet) and the defenders (*i.e.*, Internet Service Providers (ISPs), cyber security vendors and, in general, the scientific community). That is to say, the defenders needs to detect and react against all AGDs, while the attacker(s) to be able to communicate with the botnet only require a single, undetected and working domain name.

To grasp the threat it is imperative to firstly understand its magnitude. That is to say, the most recent technical reports estimate the number of malicious FQDNs to be around 9.9% of the total domain names, of

*Corresponding author

Email addresses: `mattia.zago@um.es` (Mattia Zago), `mgilperez@um.es` (Manuel Gil Pérez), `gregorio@um.es` (Gregorio Martínez Pérez)

URL: `https://webs.um.es/mattia.zago` (Mattia Zago), `https://webs.um.es/mgilperez` (Manuel Gil Pérez), `https://webs.um.es/gregorio` (Gregorio Martínez Pérez)

which, 1 in 5 belongs to DGA-based botnets (around 1.8% of all the registered domain names) [1]. To be more precise, according to the Spamhaus project [6] (and confirmed by [11], among others), the most abused domain names extensions are by far `.com` and `.uk`. Secondly, it is clear that the botnet specialisation are versatile and they normally change depending on the main actors behind them [3]. That is to say, botnets belonging to the same family are instantiated and managed by different operators, with varying objectives. In other words, botnets are offered as cloud-based malware-as-a-service [12]. Although this is a well-known security issue [13], accordingly to Spamhaus [5], ISPs are not following the best practices for customer verification enabling cybercriminals to automatic sign-up fraudulent accounts (61% of the observed C&C servers in 2018). For example, Cloudflare has been identified as the most abused ISP for hosting C&C servers [5].

Finally, and as previously mentioned, evasive techniques are widely used by botnets, especially DGA-based ones, to avoid detection. It has been estimated that the average dwell time, *i.e.*, the number of days an attacker is present on a victim network from first evidence of compromise to detection, is to be measured in months [3, 7]. Furthermore, the fact that most botnets present C&C servers in multiple countries (njRAT, DarkComet and NanoCore malwares have them in over 80 countries) further demonstrate the limitation of sinkholing and, in general, reaction techniques.

In general, endpoint countermeasures (such as blacklisting) have already been proved ineffective [14]. Therefore, the cybersecurity community is actively researching and designing machine learning (ML)-based solutions to overcome this limitation. To be more precise, there are two potential areas of application of ML-powered products, namely the detection of actively queried AGDs and the dynamic reaction against either the malware spreading and/or the infected machine communications. This research focuses on this first area of application, that is the detection of AGDs in the wild using ML techniques. Our claim is that it is mandatory to shift the detection of this peculiar class of botnets from the attack phase (*i.e.*, when the infected machines are actively engaging in malicious activities) to the early stages, in which the botnets and the C&C servers are being instantiated and configured.

The research inhere proposed leverages this idea, eventually aiming to enable cybersecurity operators to perform preemptive analysis of services to flag suspicious associated domain names. However, the very first step required in order to deploy a ML-powered solution to achieve this objective is to obtain trustworthy and reliable data to be used as a training set. As we sill demonstrate in the related work Section 2, this still represents a major challenge. In fact, as reported by [10] among others, the shortcoming of mature, ML-ready and publicly available datasets dedicated to DGA-based detection represent nowadays a critical setback.

Our proposal is to ease and eventually standardise the future researches on this subject by providing firstly a mature dataset (which will be publicly released as publicly available dataset [15], code repository [16] and documentation [17]); secondly, a complete state-of-the-art in terms of potential third-party data source (Section 2); and, thirdly, an exploratory analysis to guide future practitioner toward the open challenges in terms of machine learning applications, which also have been previously discussed by the authors [10].

To this extent, our endeavour is to propose the research community, but also the registrars, the ISPs and the cybersecurity vendors, to focus on creating innovative ML solutions for identifying DGA-based botnet threats.

Therefore, the main contributions of this research are twofold:

i) firstly, the in-depth state-of-the-art analysis regarding publicly available datasets that might provide the solid ground for ML-powered detection frameworks; and,

ii) secondly, the public release of a full-fledged, privacy-aware and ML-ready dataset, called `UMUDGA`, featuring samples from 50 malware variants for a total of 30+ million domain names.

Additionally, to support the findings and further extend the state-of-the-art, this article includes another two minor contributions, which may prove useful to future researchers, namely

iii) the exploratory analysis that serves both as data descriptor and as data breakdown and interpretation; and, finally,

iv) the discussion sprang from the analysis that flows into a collection of suggested guidelines that should be followed to achieve the required scientific rigorousness.

The structure of this article is the following. Section 2 introduces the comparison metrics and the current state-of-the-art in terms of publicly available datasets, while Section 3 presents the architecture, the methodology and in general, the characteristics of the proposed `UMUDGA` dataset. In addition, Section 4 reports a brief analysis of the data, and as a consequence, Section 4.3 pinpoints the main challenges identified and the guideline for future works. At last, Section 5 summarises-up and concludes the article.

2. Related Works

Building a formal and strict comparison of the existent related works in terms of malware datasets that includes DGA-based botnets is a non-trivial research task. Moreover, the innate scope differences between them further aggravate the shortcoming of shared and acknowledged comparison techniques.

Table 1: Comparison of datasets embodying DGA-based botnets

Name	Year	Characteristics								
		SYNT (2.1)	GNRL (2.2)	RPST (2.3)	BLNC (2.4)	EXTS (2.5)	VRFB (2.6)	PROR (2.7)	MLRD (2.8)	LABL (2.9)
PCAP Based										
ISCX IDS [18]	2012	✓	✗	≈	≈	✓	✓	✓	✗	✓
CTU [19]	2014	≈	✓	✗	✗	≈	✗	✓	≈	≈
CONTAGIO [20]	2015	✗	✗	✗	✗	✗	✗	✗	✗	✓
CyberData1 [21, 22]	2015	✗	≈	✓	✗	✗	✗	✓	✗	✗
ISOT HTTP [23]	2017	✓	✗	≈	✗	≈	≈	≈	✗	✓
CTU Extended [19]	2018	≈	✓	✗	✗	≈	✗	✓	≈	≈
Network Flows Based										
ISOT [24]	2013	✓	✗	≈	✗	≈	≈	≈	✓	✓
UNB Botnet [25]	2014	✓	✗	✗	✗	≈	≈	✓	✓	✓
PUF [26]	2018	✗	✗	≈	✗	✗	✗	✓	✓	✓
FQDNs Based										
SuperCowPowers [27]	2013	✓	✗	≈	≈	✓	✓	✓	≈	≈
Andrewaeva [28]	2014	✓	✗	✗	✗	✗	✗	✓	≈	✓
Pchaigno [29]	2015	✓	✗	≈	≈	✓	✓	✓	≈	✓
BaderJ [11, 30]	2015	✓	✓	≈	≈	✓	✓	✓	≈	✓
AmritaDGA [31]	2018	✓	≈	✓	✓	≈	✗	✓	≈	≈
Features Based										
NSL-KDD [32]	2009	✓	✗	✗	✓	✗	✗	✓	✓	✓
UCSD URL [33]	2009	✗	≈	≈	✗	✗	✗	✓	✓	≈
UMUDGA [15]	2019	✓	✓	✓	✓	✓	✓	✓	✓	✓

Legend: ✓ Yes – ✗ No – ≈ Partially

Characteristics abbreviations, as defined in Section 2:
Synthetic (SYNT), General (GNRL), Representative (RPST), Balanced (BLNC), Extensible (EXTS), Verifiable (VRFB), Privacy-Orientation (PROR), Machine Learning Ready (MLRD), Labelled (LABL)

Nevertheless, in the past decade there are several notable researches that achieved to provide great support to the cybersecurity community that study innovative solutions for tackling network threats. The scope of this survey is to analyse them as ML data sources. As will become clear in the following sections, nearly half of the existing datasets are ready "out-of-the-box" to be used in ML-powered solutions. Besides the clear issues in terms of generality and representativeness, important weaknesses will be highlighted in terms of stating whenever a data source is verifiable, reproducible and extensible. To be more precise, we have collected 9 fundamental characteristics that a ML dataset should achieve according to both our view and the researches available in literature. In the following paragraphs, we will discuss these characteristics and highlight how the publicly available datasets struggle to excel in all of them. The outlines of such comparison can be found in Table 1. In the table, the last line is dedicated to summarise the achievements of our proposed dataset, UMUDGA, which properties will be discussed in Section 3.

Finally, a review of literature approaches in terms of ML solutions for tackling DGA-based botnets is available at [10].

The first and foremost is the property that establishes whether the dataset is composed of real data or it is an artificial artefact. We thus introduce the first property 2.1.

Definition 2.1: Synthetic (SYNT). The dataset is artificially created either by generating the samples or by mixing multiple sources.

Firstly, the SYNT does not represent a binary "good *versus* bad" feature. Instead, both values are important and legitimate required depending of the application scope and purpose. That is to say that on the one hand, a ✗ value represents a dataset that makes use of data that has been organically captured from real networks and real infected machines [20, 21, 22, 34, 35]; on the other hand, a ✓ value represents a synthetic dataset, that can be generated (with different degrees of randomness). The former requires to complete the considerable task of having the data labelled for training purposes, while the latter, if accurately and unbiased generated, might potentially represent a more reliable source of new labelled data.

It is important to notice that synthetic does not implies unrealistic data, however, it is possible that the generation process introduces dependencies that ML algorithms can detect [25]. That is to say that the action of mixing two different data sources does not imply better performances. To provide some examples, the following scenarios are considered synthetic: i) the injection of previously captured malware traces in any traffic data; and ii) the generation of malware data by executing the malware in a controlled environment.

As previously mentioned, and as summarised in Table 1, the dataset strictly composed by real data collected from real network are rare (CONTAGIO [20], CyberData1 [21, 22], PUF [26] and UCSD URL [33]). Most authors prefer to inject malware data into background traffic, generate it in controlled environment, or mixing the captures from different sources (ISCX IDS [18], CTU [19], ISOT HTTP [23], ISOT [24], UNB Botnet [25] and NSL-KDD [32]). A noticeable trend is identifiable with respect to the FQDNs-based datasets, in which the data consists of AGDs lists collected from multiple sources like security vendors or bulletins, often including also an implementation of the DGA (SuperCowPowers [27], Andrewaeva [28], Pchaigno [29], BaderJ [11, 30] and AmritaDGA [31]).

Secondly, in parallel with Property 2.1, there are the *General* ***GNRL***, *Representative* ***RPST*** and *Balanced* ***BLNC*** characteristics, that reflect the realism of the dataset. Although a dataset cannot be at the same time synthetic and real, it surely can be both synthetic and realistic. In fact, by including a wide range of malware families (Property 2.2, GNRL) [18, 19, 25, 34, 35], each one represented by a sizeable (Property 2.3, RPST) [18, 25, 26, 34, 35] amount of samples, comparable to the other classes (Property 2.4, BLNC) [36], might result in a realistic representation of a real environment. Thus, the following definitions hold:

Definition 2.2: General (GNRL). The dataset covers a wide range of malware families rather than being composed by a few specific examples. To be more precise, the volume of the data is enough to accurately represent a real-world scenario.

Definition 2.3: Representative (RPST). The dataset includes, for every category, enough instances to accurately reflect the characteristics of the larger population.

Definition 2.4: Balanced (BLNC). The dataset has a comparable number of samples for each category, *i.e.*, the number of instances belonging to a class should not outnumber any other class.

One could say that a dataset including several instances of a specific malware execution is representative of the variant (Property 2.3, RPST), but not general (Property 2.2, GNRL) nor balanced (Property 2.4, BLNC). Moreover, a dataset that features multiple instances of several malware variants might be general (Property 2.2, GNRL) with respect of the network threats, representative of the malware families examined (Property 2.3, RPST), but not balanced with respect to legitimate or background traffic (Property 2.4, BLNC). As previously mentioned, these three characteristics are quite difficult to find altogether in a dataset, in fact, most of the PCAP-based repositories are not general (Property 2.2, GNRL) nor representative (Property 2.3, RPST), let alone balanced (Property 2.4, BLNC). Nevertheless, a few exceptions shine, even if only in a single category, namely the completeness of CTU [19] and BaderJ [11, 30], in terms of number of malware families (Property 2.2, GNRL) and the amount of samples in each class represented by CyberData1 [21, 22] and AmritaDGA [31] (Property 2.3, RPST). To the best of our knowledge, there is no dataset (apart from AmritaDGA [31] and NSL-KDD [32]) that presents clearly balanced data samples.

Thirdly, to ensure the reusability and the consistency of the results derived from the dataset, two properties are defined aiming to: i) measure whether the research community can extend and eventually enhance the data (Property 2.5, EXTS) [34]; and ii) verify the data, when the whole replication process is not doable (Property 2.6, VRFB) [34].

Definition 2.5: Extensible (EXTS). The dataset is publicly available and well documented to enable the research community to extend or combine it with other data sources aiming to improve its reusability.

Definition 2.6: Verifiable (VRFB). The data included in the dataset provide enough means to permit the research community to prove the consistency, the accuracy and the genuineness of the data, ideally resulting in a fully reproducible dataset.

Datasets composed by PCAP files have medium-to-low extensibility due to the fact that mixing it with other traffic sources might not result in a effective dataset. Moreover, dataset obtained by eavesdropping a real network is to be considered not verifiable and not replicable, because of the strong dependence from the context and the environmental conditions. However, a capture that is obtained from a controlled environment, *i.e.*, a testbed, is to be considered verifiable and potentially replicable, depending on the generation process. Finally, the simultaneous replay of multiple well-known traffic captures is to be considered both verifiable and replicable. Not surprisingly, both properties are generally satisfied by the FQDNs-based category, while the other datasets generally achieve partial success at best. Notably, the ISCX IDS [18] is the only dataset in its category to achieve both properties.

Fourthly, as previously stated by [10, 19], a dataset should be made publicly available without doubts or concerns about harming users' privacy. Property 2.7, PROR, has been defined to target this aspect.

Table 2: Dataset comparison in terms of number of FQDNs available for analysis and number of classes (including the legitimate one, when available).

Name	Year	Legit	AGD	Unique	Valid	Classes
ISOT [24]	2013	30,699	32,952	63,651	31,297	5
SuperCowPowers [27]	2013	1,000,000	2,670	1,002,670	986,081	2
Andrewaeva [28]	2014	1,000,000	694,173	1,694,173	1,694,167	9
CTU [19]	2014	73,020	153,999	227,019	62,620	8
UNB Botnet [25]	2014	46,440	15,734	62,183	46,474	17
CONTAGIO [20]	2015	0	8,612	3,637	3,620	110
ISOT HTTP [23]	2017	3,114	105	3,219	1,298	9
AmritaDGA [31]	2018	2,498,076	1,072,418	3,570,494	3,405,238	21
`UMUDGA` [15]	2019	1,000,000	30,799,449	31,799,449	30,799,449	51

Definition 2.7: Privacy-Orientation (`PROR`). The dataset has been designed to not include any privacy-harming content nor is required for the research community to harm the users' privacy in order to deploy or include the data in their experiments.

Any form of network traces that also includes the payload (*e.g.*, PCAP files) is natively including personal data that potentially harm the users' privacy. Although the network flows format dictates to strip the payload of the packets in order to aggregate them, the IP addresses are still considered to certain extent as personal information. As expected, almost all dataset composed by network flows, AGDs lists or features do not contain personal information or user-specific data (UNB Botnet [25], PUF [26], SuperCowPowers [27], Andrewaeva [28], Pchaigno [29], BaderJ [11, 30], AmritaDGA [31], NSL-KDD [32] and UCSD URL [33]). It is also interesting to note that, generally, the PCAP-based solutions presents some sort of data anonymisation even when they are not built as privacy-oriented solutions (ISCX IDS [18], CyberData1 [21, 22], ISOT HTTP [23] and CTU (both original and Extended) [19]).

Finally, one of the scopes of this research is to explore the state-of-the-art oriented toward ML applications. To achieve this, two properties are defined to measure how easy it is to use "as-is" (Property 2.8, `MLRD`) (as suggested by [26]) and to indicate whether the dataset is labelled or not (Property 2.9, `LABL`) [18, 19, 26, 34, 35, 36].

Definition 2.8: Machine Learning Ready (`MLRD`). The dataset is composed by carefully curated samples. There are no missing values nor unwanted characters. Moreover, the data format is consistent across all the samples and it is suitable for usage with the leading tools.

Definition 2.9: Labelled (`LABL`). Each sample is carefully characterised with one or more class attributes, eventually providing a variable granularity of the labels.

For example, a dataset composed by network flows is directly suitable of being directly plugged into ML solutions, provided that the target platform can support string and date features. By nature, both network flows based solutions (ISOT [24], UNB Botnet [25] and PUF [26]) and feature based (NSL-KDD [32] and UCSD URL [33]) ones are natively pluggable in ML algorithms, while FQDNs lists (SuperCowPowers [27], Andrewaeva [28], Pchaigno [29], BaderJ [11, 30] and AmritaDGA [31]) are directly usable only by a subset of them (*e.g.*, deep learning or text processing). With regards to the labels, only CyberData1 [21, 22] does not present any form of class separation. In order to provide an overall view, and as previously mentioned, in Table 1 it is possible to find the relevant datasets compared according to the aforementioned properties.

It is worth mentioning, that although not pinpointed in this section, our proposed dataset, `UMUDGA`, meets all the properties here described. A detailed discussion of such achievements is offered in Section 3,

Finally, for a few selected datasets (either for their importance or their properties) we realised also a quantitative analysis in terms of number of legitimate and malicious domain names and number of classes. Table 2 reports these findings. The validation column is obtained by processing each FQDN with Google Guava library, and specifically its `InternetDomainName` class's method which checks if the domain name is syntactically valid using lenient validation [37].

To be more precise, we excluded the remaining datasets for being obsolete (NSL-KDD [32] and UCSD URL [33]), not labelled (CyberData1 [21, 22]) or labels not aligned with the scope of this article (ISCX IDS [18] labels attacks, not malwares) and finally, for not being yet publicly released (PUF [26]). With regards of the CTU [19] and the CTU Extended [19], we have considered them as a single dataset, extracting the data from both. Pchaigno [29] and BaderJ [11, 30] are datasets of DGAs, thus the quantitative comparison based on the number and properties of the FQDNs is not applicable. To be more precise, and as reported both in Section 3 and [17], the generators used in our dataset, `UMUDGA`, are using both [29] and [30] as source, among others.

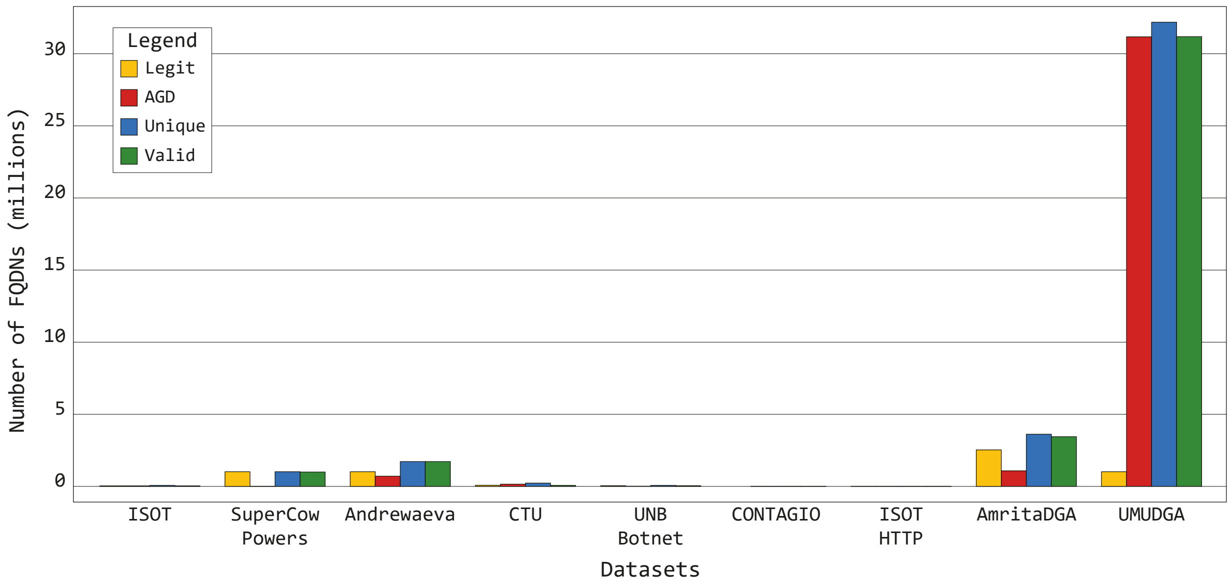
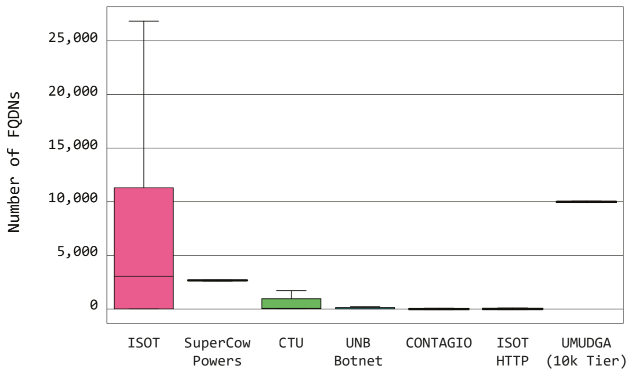
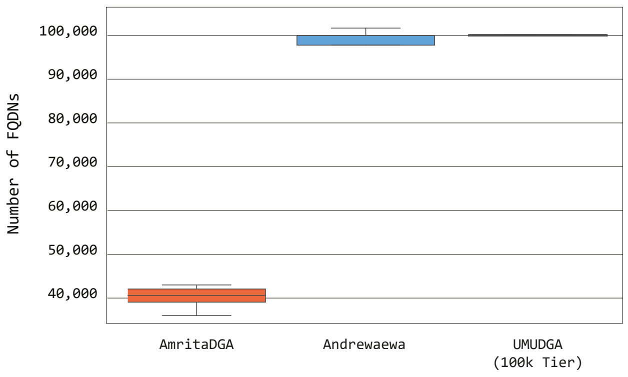


Figure 1: Datasets metrics, as reported in Table 2.



(a) Datasets with less than 30 thousands FQDNs per class.



(b) Datasets with more than 30 thousands FQDNs per class.

Figure 2: Boxplots representing the number of FQDNs per class. The ideal dataset present a small interquartile range (*i.e.*, more balanced) and a high median (*i.e.*, more samples per class).

The numbers reported in Table 2, have been graphically highlighted also in Figure 1 (in terms of total number of samples) and in Figure 2 (in terms of number of samples per class). In particular, Figure 2a and Figure 2b presents the different datasets with a suitable scale. In the figures, the ideal dataset presents a small interquartile range (all the classes have roughly the same amount of samples, *i.e.*, it satisfies Property 2.4, `BLNC`) and a high median (the average number of samples per class is elevated, thus potentially achieving Property 2.3, `RPST`). In both figures, outliers are not represented and our proposed `UMUDGA` has been included with the appropriate Tier (*i.e.*, the size of each class sample set, as will be described in Section 3).

In Table 2, the first two columns, namely the *legit* and the *AGD* columns, represent the amount of FQDNs obtained from the lists or PCAP files, and thus do not include those databases that do not include domain names [21, 22, 29, 30, 32, 33] and those that are not publicly available [26]. The *legit* column reports the number of FQDNs considered legitimate while the *AGD* one reports the number of malicious domains, using wireshark [38] to extract the `dns.qry.name` field for PCAP-based datasets. The two lists are then combined, sorted and all duplicates are removed, results are reported in the *unique* column. Moreover, the fourth column (*valid*) is obtained by processing each FQDN with Google Guava library, and specifically its `InternetDomainName` class's method which checks if the domain name is syntactically valid using lenient validation [37].

Finally, the datasets have been analysed according to their overlaps in terms of FQDNs and Table 3 identifies the amount of collisions registered within each dataset. To be more precise, the overlap is defined as the percentage of the dataset that is shared with the others, *i.e.*, giving any two lists of FQDNs, namely A and B, the percentage of collision is calculated as follows:

$$\text{overlap}(A, B) = 100 \cdot \frac{|A \cap B|}{|A|} \tag{1}$$

Table 3: Comparison of datasets in terms of overlapping percentages

Dataset	Year	Percentage of overlap with									
		[24]	[27]	[28]	[19]	[25]	[20]	[23]	[31]	`UMUDGA`	`Alexa`
ISOT [24]	2013	N.A.	12%	13%	16%	100%	1%	—	19%	10%	10%
SuperCowPowers [27]	2013	—	N.A.	44%	1%	—	—	—	41%	22%	22%
Andrewaeva [28]	2014	—	25%	N.A.	—	—	—	—	38%	24%	18%
CTU [19]	2014	8%	10%	10%	N.A.	22%	1%	—	17%	8%	8%
UNB Botnet [25]	2014	67%	9%	10%	29%	N.A.	2%	—	20%	7%	7%
CONTAGIO [20]	2016	12%	9%	9%	21%	24%	N.A.	2%	34%	12%	7%
ISOT HTTP [23]	2017	3%	15%	16%	7%	12%	7%	N.A.	81%	16%	16%
AmritaDGA [31]	2018	—	12%	19%	—	—	—	—	N.A.	16%	12%
`UMUDGA` [15]	2019	—	1%	1%	—	—	—	—	2%	N.A.	3%

With `Alexa` we indicate the top one million domains [39], 2018 update.
With "—" we indicate that the overlap is either non-existent or smaller than 0.1%.

It is important to notice that the function is not symmetric, that is to say, a permutation of the input variables changes the result value, *i.e.*, overlap$(A, B) \neq$ overlap(B, A). Table 3 does not report values smaller than 0.1%.

As shown in Table 2, and to the best of our knowledge, this is the first attempt to provide a comprehensive and representative dataset to be used for tackling DGA-based botnets. In using our proposed dataset, one of the main advantages that the research community might acquire relies upon the formal definition [17] of the features and the verifiable feature set obtained as result [15].

3. UMUDGA: University of Murcia Domain Generation Algorithm Dataset

One of the main outcome of this article is the public release of a ML-ready and privacy-aware dataset of AGDs. As previously reported in Table 1, our solution matches all the defined properties. To be more precise, as as reported in Section 3 and its subsections, the `UMUDGA` dataset meets those properties as follows:

- Property 2.1, Synthetic (`SYNT`) — The dataset is generated by executing malware DGAs and collecting the resulting data, thus achieving the requested `SYNT` property.
- Property 2.2, General (`GNRL`) — The dataset includes 38 malware families (Table 4), presenting more than 30 million FQDNs distributed over 50 malware variants besides the legitimate class (Table 2). To the best of our knowledge, this covers the vast majority of publicly known DGA-based malwares.
- Property 2.3, Representative (`RPST`) — As summarised in Table 4, all variants include at least 10,000 FQDNs (*i.e.*, first Tier in Table 4), having most of them 1 million valid and unique FQDNs (*i.e.*, highest Tier in Table 4).
- Property 2.4, Balanced (`BLNC`) — As summarised in Table 4, the data are sorted in tiers of different sizes. Within each tier, all the malware variants are fully balanced.
- Property 2.5, Extensible (`EXTS`) — The code for generating the AGDs is available on Mendeley Data [15].
- Property 2.6, Verifiable (`VRFB`) — All the data sources are publicly available online. Moreover, the data repository itself reports the formal mathematical definition for each feature presented.
- Property 2.7, Privacy-Orientation (`PROR`) — The dataset is composed only by *context-free* features, which are natively anonymous and privacy-oriented. They, in fact, do not require any contextual information from the users or the network state [10].
- Property 2.8, Machine Learning Ready (`MLRD`) — The data have been preprocessed to assure the absence of missing or corrupted data. The repository provides the data in both raw TXT FQDNs lists, CSV and ARFF formats [15].
- Property 2.9, Labelled (`LABL`) — The data are available as collection of malware variants data sources, natively tagged with the correct label.

As previously stated [10], the public release of a ML-ready dataset represents an innovative response to a well-known challenge in the cybersecurity field, however, future researches are still required to extend it to both *context-free* and *context-aware* features.

The following Section 3.1 presents the architecture of the data processing and collecting framework, while Section 3.2 will describe the methodology used for designing and building the dataset.

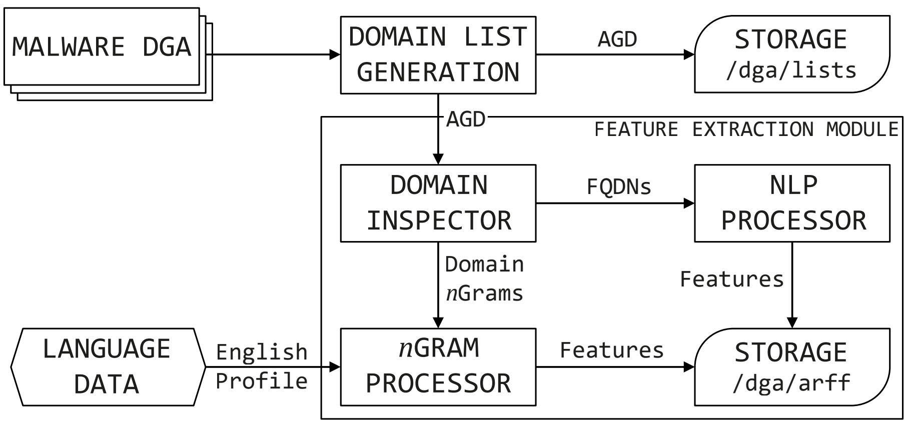


Figure 3: Architecture for the dataset generation showing both the required inputs (the malware DGAs and the English Language Data) and the provided outputs (the AGD lists and the AGD features sets).

3.1. The UMUDGA Architecture

The generation framework flow consists in executing malware DGAs to collect the AGDs and then process them to obtain the relevant features. Figure 3 illustrates the data flow.

To be more precise, and as explained in detail in Section 3.2, the raw FQDNs are obtained by executing in a controlled environment the malwares' DGAs, which in turn are both saved as raw lists and processed to become ML-ready data. The AGDs are firstly processed by the `Domain Inspector` procedure, entry point for the `Feature Extraction Module`, which takes care of validating each domain name and extract relative nGrams. The validation is carried out with both the Google Guava `InternetDomainName` class [37] and the Apache Commons Validator library [40]. The former performs syntax validation while the latter evaluates the domain names according to the standards RFC 1034 [41, Section 3] and RFC 1123 [42, Section 2.1]. A detailed explanation of the technical validation is offered in a companion article that provides the dataset description [15].

Firstly, the domains are analysed by a Natural Language Processing (NLP) procedure that extracts 15 common features such as the length of the domain, its vowel ratio, among others; while, secondly, the nGrams are analysed by the corresponding procedure to extract 31 features for each nGram size (with $n = 1, 2, 3$), *e.g.*, entropy, frequencies. As previously mentioned, the features, among other technical aspects of the dataset, are described in [15].

The n`Gram Processor` primarily compares the nGrams distribution with the corresponding distribution of the English language, provided by the Leipzig Corpora [43].

The following methodology Section (3.2) discuss in detail the content of each module and procedure.

3.2. Methodology for building the UMUDGA dataset

This section aims to illustrate the procedures, the assumptions and the tools used to collect, filter, and generally prepare the data. The process is twofold, *i.e.*, the data are firstly collected in form of raw FQDNs lists (see Section 3.2.1); secondly, the domains lists are processed and the resulting features files are saved in the dataset (see Section 3.2.2).

3.2.1. Generation and collection of FQDNs

Malware lists such as [44, 45, 46, 47] are quite common and used on daily basis by multiple firewalls and anti-malware providers. However, the provided data are rarely identified with the malware family or variant, thus they are more often labelled as generic threats. As a consequence, the approach that we took for building this dataset is slightly different. That is to say, instead of collecting lists of AGDs from multiple online sources like those, we have been looking for the study and the actual implementation of the malwares' DGAs. Therefore, our data are exclusively generated by executing DGAs implementations in a controlled environment (achieving the Property 2.1, `SYNT`). The source code has been adapted from three of the main providers of DGAs implementations [28, 29, 30] and will be released in a public repository (thus achieving Property 2.5, `EXTS`).

Each algorithm's random function is initialised with a fixed seed and when required, this random module is also used to derive both dates and other arguments. For example, whenever a DGA requires a new date to calculate the corresponding AGDs, the support infrastructure provides a random datetime string derived from the initialisation seed (Property 2.6, `VRFB`).

The generated AGDs are guaranteed to be unique within the class, however, this property is not forced across the dataset when considered as a whole. To be more precise, there are 551 collisions shared among 10 malware variants (*e.g.*, `Pizd` shares 441 AGDs with first version of `SuppoBox`). However, as highlighted by [11] and proved

Table 4: DGA families

Tier	Families collected
10,000	CCleaner, Kraken*, Murofet*, Pizd†, Pykspa, SuppoBox*,†, Vawtrak*
50,000	Vawtrak*, Gozi*,†
100,000	Pykspa-noise, QakBot, Ramnit, Tempedreve, Gozi*,†
500,000	Banjori, Murofet*
1,000,000	Alureon, Bedep, ChinAd, CoreBot, CryptoLocker, DirCrypt, Dyre, Fobber*, Kraken*, Locky, Matsnu†, Necurs, Nymaim, PadCrypt, Proslikefan, Pushdo, Qadars, Ramdo, Ranbyus*, Rovnix†, Shiotob, Simda, Sisron, Symmi, Tinba, Vawtrak*, Zeus-NewGoz.

† Wordlist-based family.
* Multiple variants

Table 5: List of features generated by the `NLP Processor` for each FQDN.

Code	Description
`L`-x	String length of x domain level
`N`	Number of domain levels
`LC-C`	Longest consecutive consonance sequence
`LC-D`	Longest consecutive number sequence
`LC-V`	Longest consecutive vowel sequence
`R-CON`-x	Ratio of consonants characters
`R-LET`-x	Ratio of letter characters
`R-NUM`-x	Ratio of numerical characters
`R-SYM`-x	Ratio of symbolical characters
`R-VOW`-x	Ratio of vowel characters

where $x \in \{\texttt{FQDN}, \texttt{2LD}, \texttt{OLD}\}$ denotes the domain levels.

by our analysis (551 over 30.8 million total domains), collisions are quite rare among different malware families, and negligible when considering also the legitimate category. Therefore, removing eventual collisions does not produce a statistically significant change in the malware family distribution, while substantially increasing the quality of the data for ML usage. To be as complete as possible, the full list of colliding FQDNs is available in the repository [15].

Each DGA is executed until it stops generating new domain names or it reaches 1 million AGDs (Property 2.3, `RPST`). Resulting FQDNs are then truncated to the highest completed tier, *i.e.*, 10k, 50k, 100k, 500k, 1M. Table 4 reports the list of the families sorted according to their highest tier, *e.g.*, at least one variant of `Kraken` consists of only 10k AGDs. To the best of our knowledge, these 50 variants are covering the vast majority of known DGA-powered malwares (Property 2.2, `GNRL`).

To complete the dataset, the last set of FQDNs is obtained by joining the Alexa [39] and the Majestic Million [48] domain lists. From the two lists, a million unique domains are extracted and allegedly considered as *legitimate*. However, two main problems aroused when validating those domain names, in fact a total amount of 178 FQDNs fail to pass the validation procedure. To be more precise:

- 38 of them use one of the new generic top level domains (gTLDs) which are still not included in the list of accepted gTLDs as per the last update of the validation library (Apache Commons Validator [40] – v1.6, 04/02/2017). Namely, `.africa` (delegated on 14/02/2017), `.charity` (04/06/2018), `.hotels` (03/04/2017), `.inc` (16/07/2018) and `.sport` (08/01/2018).
- 140 domains are technically invalid because of the presence of at least one underscore character ("`_`"): the validation library checks the domains against the RFC 1123 [42], which limits host names to letters, digits and hyphen. The policy for the underscore character has been clarified later with the RFC 2181 [49, Section 11].

3.2.2. Preprocessing and feature extraction

To begin with the inner mechanisms of such module, as illustrated in Figure 3, it is imperative to restate what firstly proposed by Zago et al. [10], *i.e.*, the generated features belong to the *Context-Free* family. That is to say, the features are related only to a FQDN and are independent of contextual information.

Firstly, the FQDNs lists obtained in the previous steps are analysed and syntactically validated against RFC 1034 [41, Section 3] and RFC 1123 [42, Section 2.1] by the `Feature Extraction Module` reported in Figure 3.

Table 6: List of features generated by the n`Gram Processor` for each nGram set.

Code	Description
n`G-DIST`	Number of distinct nGrams
n`G-25P`	25th percentile of frequencies
n`G-50P`	50th percentile of frequencies
n`G-75P`	75th percentile of frequencies
n`G-MEAN`	Mean of frequencies
n`G-QMEAN`	Quadratic Mean of frequencies
n`G-SUMSQ`	Squared sum of frequencies
n`G-VAR`	Variance of frequencies
n`G-PVAR`	Population VAR of frequencies
n`G-STD`	Standard deviation of frequencies
n`G-PSTD`	Population STD of frequencies
n`G-SKE`	Skewness of frequencies
n`G-KUR`	Kurtosis of frequencies
n`G-PRO`	Pronounceability score^1
n`G-NORM`	Normality score
n`G-DST-CA`	Canberra Distance^1
n`G-DST-CH`	Chebyshev Distance^1
n`G-DST-EM`	Earth Movers Distance^1

Code	Description
n`G-REP`	Number of repeated nGrams
n`G-E`	Entropy
n`G-COV`	Covariance^1
n`G-KEN`	Kendall's Correlation^1
n`G-PEA`	Pearson's Correlation^1
n`G-SPE`	Spearman's Correlation^1
n`G-TSUMSQ`	Squared sum of target language frequencies^1
n`G-TVAR`	Variance of target language frequencies^1
n`G-TPVAR`	Population VAR of target language frequencies^1
n`G-TSTD`	Standard deviation of target language frequencies^1
n`G-TPSTD`	Population STD of target language frequencies^1
n`G-TSKE`	Skewness of target language frequencies^1
n`G-TKUR`	Kurtosis of target language frequencies^1
n`G-TSUM`	Sum of target language frequencies^1
n`G-DST-KL`	Kullback-Leiber divergence^1
n`G-DST-JI`	Jaccard Index measure^1
n`G-DST-EU`	Euclidean Distance^1
n`G-DST-MA`	Manhattan Distance^1

where ^{1} is about the English language, and n`G` indicates the size of the nGram collection used for the group of features.

All the invalid domain names are replaced with other unique samples obtained by the corresponding DGA. The dataset documentation provides the required mathematical formalism for each implemented feature [17].

With regards to Figure 3, the first process is the `NLP Processor`, which analyses each domain name as string, with little to none knowledge about its structure as FQDN or its language. The `NLP Processor` extracts 22 features from the domain name, which are listed and described in Table 5. To improve the readability, Table 5 presents a list of meta features, that indicates that a specific mathematical formula has been applied to multiple targets. That is to say, we indicate with x the domain level used as argument for calculating the feature, having x equals to either "`FQDN`" (that stands for Fully Qualified Domain Name), "`2LD`" (that stands for Second Level Domain) and "`OLD`" (that stands for Other Level Domain, which comprehends any domain level below the second).

The second process, namely the n`Gram Processor`, is the one that analyses the domain name as a collection of tokens, called nGrams. Firstly, each FQDN is divided into chunks of size n and then compared to the reference ones belonging to the English language. These last collections are obtained by preproccessing the Leipzig Corpora [43] which includes 1 million words from Wikipedia (2016 update). There are a total of 29 features extracted from such analysis; Table 6 presents them. Despite having these features listed once in Table 6, the dataset includes them applied to 1Grams, 2Grams and 3Grams for a total of 87 features. That is to say, the 29 features are mathematically defined independently from the chosen length (n) of the chunks used for the analysis and thus can be applied to any nGram size.

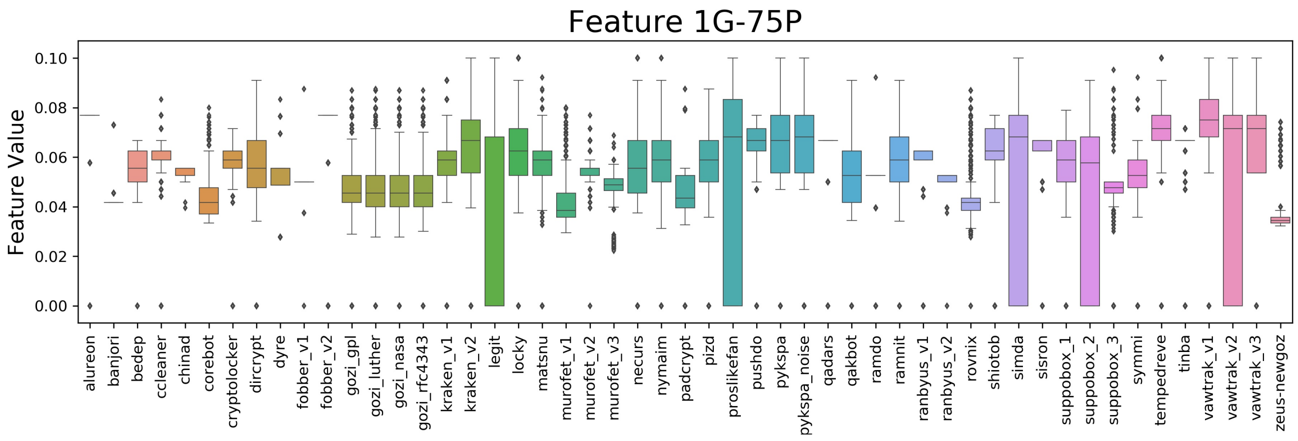
As a side note, some of these features applied to 2Grams and 3Grams are (almost) constant and thus (practically) irrelevant. They are nevertheless included in the dataset for completeness. Figure 4 presents one of such features, namely the specific case of the 75th percentile of the nGrams distributions. For example, Feature `1G-75P` (Figure 4a) have been proven sufficiently informative for ML applications, despite having its counterparts zeroed-out. In fact, by considering the nature of the feature itself, it does not surprise that both the `2G-75P` (for 2Grams, showed in Figure 4b) and the `3G-75P` (for 3Grams, showed in Figure 4c) present a very skewed distributions, where only a few of them have actually a value. Both of them are nevertheless included for symmetry and completeness.

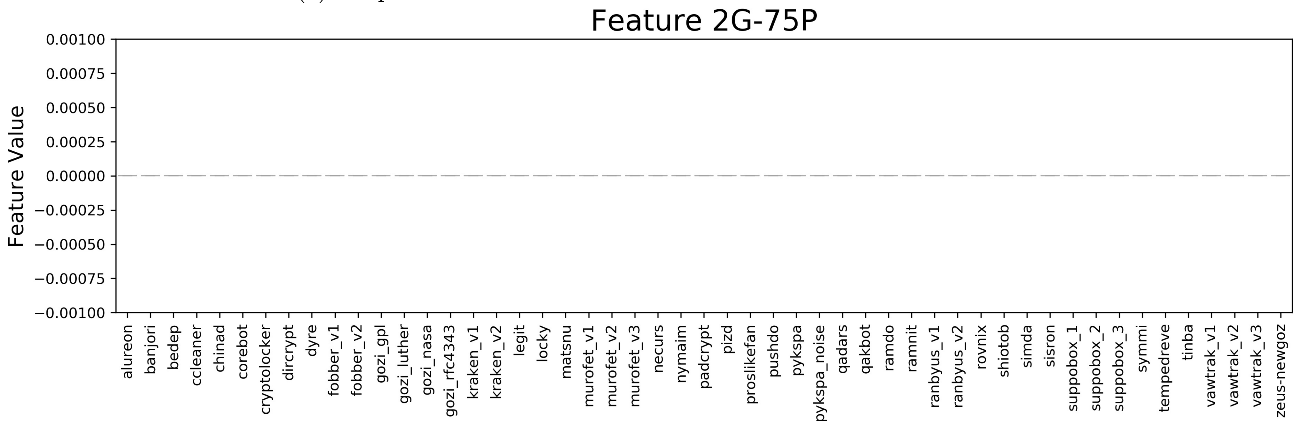
Due to space concerns, the full list of features, with their distributions and descriptions is not included here. Nonetheless, it is publicly available at [16].

Finally, a survey in terms of where these features originated and when have been used in literature to power ML-based solutions can be found at [10].

4. UMUDGA Dataset Analysis

Generally speaking, a first exploratory analysis of the data is suggested and often required before applying more sophisticated ML algorithms. This section aims to provide a brief characterisation of the malware variants


(a) Boxplots for the Feature `1G-75P` for the 1Grams distributions.


(b) Boxplots for the Feature `2G-75P` for the 2Grams distributions.

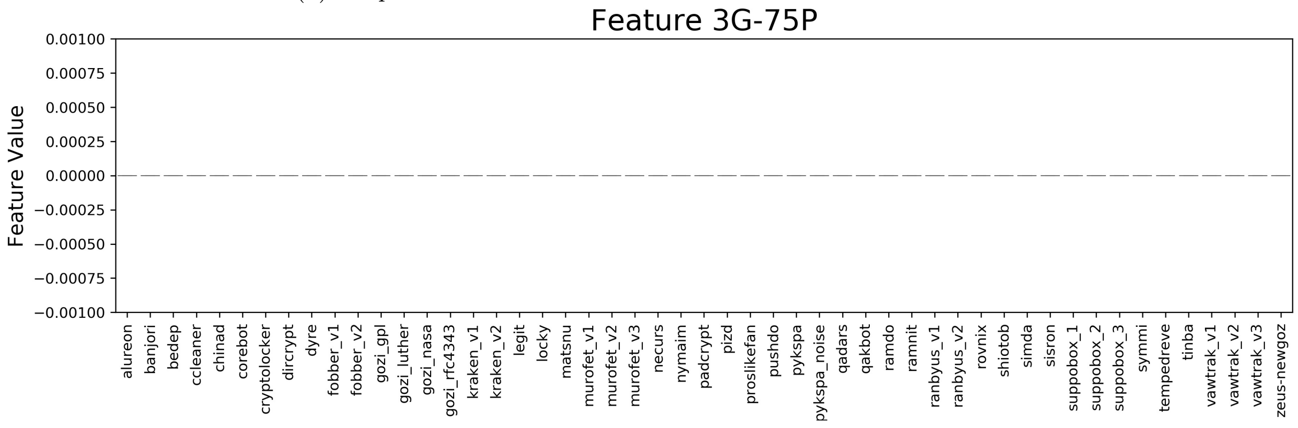

(c) Boxplots for the Feature `3G-75P` for the 3Grams distributions.

Figure 4: Boxplots comparisons for the Feature n`G-75P` that indicates the 75th percentile of the nGrams distributions. (See Table 6).

and their properties by presenting and discussing the results of a first and naïve analysis. The analysis carried out in the section below is performed over the first tier of malwares, as depicted in Table 4.

Among the features described in Section 3, few of them present signs that indicate their low quality with regards to the data analysed.

Firstly, there are features in which nearly all the values are identical, to be more precise the percentiles of frequencies, their mean and median for $n = 1, 2, 3$ are statistically indistinguishable. Moreover, the different ratios are mostly alike when considering the second level domain (2LD) and the other level domain (OLD) parts. Lastly, the number of repeated nGrams and their covariance with respect to the $n = 3$ size are also practically identical. Lastly, there are multiple features that are highly correlated, thus considering the ones with a correlation index lower than 0.9.

On the one hand, when considering the structure of the AGDs, *i.e.*, most of them are composed only by two domain levels, it is not surprise that the NLP features like the ratios or the lengths are correlated to each other, specifically when looking at the ones calculated over the FQDN and the 2LD.

However, giving the nGram analysis in combination with the shortness of the domain names, it is not unexpected that the features that are most sensible to zeros have been discarded. For example, the number of distinct nGrams, their variance, standard deviation, skewness, etc., presents high correlation grades with each other. Also the Manhattan distance, calculated as the sum of the absolute deviation, tends to be highly correlated with other distances included in the data.

In order to further explore the data we designed two ML classification tasks. These experiments were conducted on a virtual server with 18 cores running at 2.30GHz and 50 GB of DDR3 RAM at 1600 MHz. Experiments were run using Orange3 [50]. Six classifiers were applied and cross-validated using a stratified 10-fold approach, using the following configuration and ML techniques:

- **AdaBoost (AB)** — Using 50 trees as base estimators, with SAMME.R classificator (updates base estimator's 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-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 Machines (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.

In the following sections, unless otherwise stated, the experiments were conducted using all the variants belonging to the Tier 10,000, *i.e.*, a balanced dataset with 10,000 samples for each class.

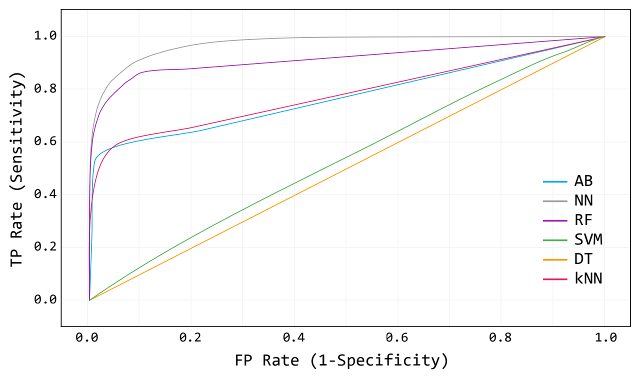
The two classification tasks are defined as follows:

Experiment 1: Binary. The Binary task 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 task is designed to classify not only the legitimate FQDN, but also sort malware samples according to their variants.

Figure 5 and Figure 6 report both the results in terms of classifiers performances and the Receiver Operating Characteristic (ROC) curves for the Binary (Exp. 1) and the Multiclass (Exp. 2) experiments, respectively. It is worth mentioning that the ROC curves are generated considering the legitimate (legit) class as target, thus showing the support in correctly predicting this class on average with respect to the 10 folds analysed. Figure 5a and Figure 6a have been simplified by firstly interpolating the values and then by smoothing the curve to reduce the number of points (accepting a loss of 5% precision).

From the two images, it is possible to notice that the results are somewhat different from the high-grade extremely precise results obtained in literature over subsets of the same classes [10]. One could argue that the data are different, which is somewhat correct; data sources are different, and rarely publicly shared for subsequent analysis (*i.e.*, Property 2.6, `VRFB`). We assume that the data obtained from the generators, as described in Section 3, for a specific malware variant are taken from the same space, thus having similar characteristics. It can also be stated that the features used are not the same, nor calculated in the same way. This assertion surely holds. However, as showed in our early research [10], most literature works made usage of context-free features, which have been collected, analysed and re-implemented as presented in Section 3.

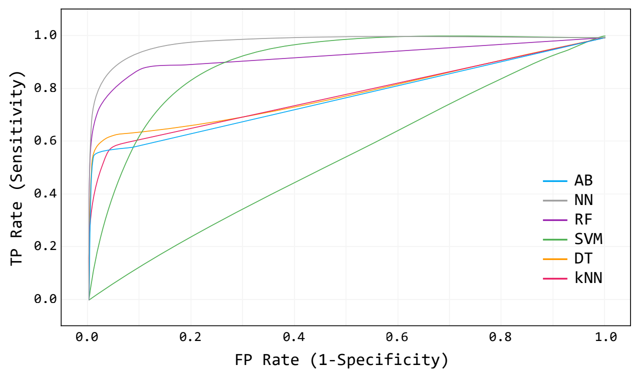


(a) Average ROC curves

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.981	0.982	0.981	0.770	0.981
NN	0.989	0.988	0.989	0.973	0.988
RF	0.989	0.988	0.989	0.914	0.989
SVM	0.403	0.403	0.965	0.403	0.556
DT	0.980	0.961	0.980	0.500	0.971
kNN	0.986	0.984	0.986	0.781	0.983

(b) Classifiers performances

Figure 5: Results for the Binary experiment (Exp. 1)



(a) Average ROC curves for the legitimate class

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.665	0.666	0.665	0.829	0.666
NN	0.772	0.775	0.772	0.993	0.769
RF	0.710	0.706	0.710	0.974	0.704
SVM	0.420	0.432	0.420	0.945	0.385
DT	0.672	0.671	0.672	0.850	0.672
kNN	0.312	0.297	0.312	0.765	0.297

(b) Classifiers performances

Figure 6: Results for the Multiclass experiment (Exp. 2)

Moreover, the features selected in the literature are rarely mathematically defined, let alone implemented and made publicly available (*i.e.*, Property 2.5, `EXTS`). Once again, without having a structural formalism of the performed data processing that led to those results, comparing them is somewhat difficult.

As expected, the Binary experiment (Exp. 1) performs much better than the Multiclass one (Exp. 2). And, to further explore these average results, a sample of a class-specific analysis is proposed in Section 4.1 and Section 4.2.

Figure 7 presents the confusion matrix for the Random Forest classifier in the Multiclass experiment as an heatmap chart. In the figure, the darker the colour, the better is the classifier precision. From the picture, it appears clear that although the classifier achieves excellent results in most of the classes, some clusters of classes appears to be difficult to separate, ultimately causing the degraded overall performances presented in Figure 6b. In Figure 7, actual percentages are omitted for clarity, the complete report is available at [15].

To further explore these classes clusters, we picked two clusters of variants, namely the one formed by `Alureon` and `Fobber (2nd version)` and the one composed by `Bedep`, `DirCrypt` and `Ramnit`. The results of these analysis are presented in Section 4.1 and Section 4.2 respectively. The experiments have been performed with amount compatible with the highest tier available for each malware variant. As indicated in both section, a potential solution for this issue might be represented by the double detection process implemented by [51]. To be more precise, by considering the elements of the cluster as a single unique entity, it is possible to flag as "suspicious" a DNS query made by some user (*i.e.*, the first, high-level detector). A dedicated analyser (*i.e.*, the second, fine-granularity detector) might then take care of performing a more precise, accurate and time consuming analysis of the identified user.

To further explore the potential researches that might spring from the usage of this dataset, it is worth mentioning the challenges related to ML identification of DGA-based botnets. For example, it is unclear whether is possible to define a common signature for AGDs as a group (Exp. 1) or if from a sample analysis is possible to incontrovertibly identify the malware family or even the precise variant (Exp. 2). Further researches are also required in terms of algorithms application to explore the data, for example, literature suggests that both deep learning and clustering solutions might prove useful in identifying known and unknown malwares, respectively.

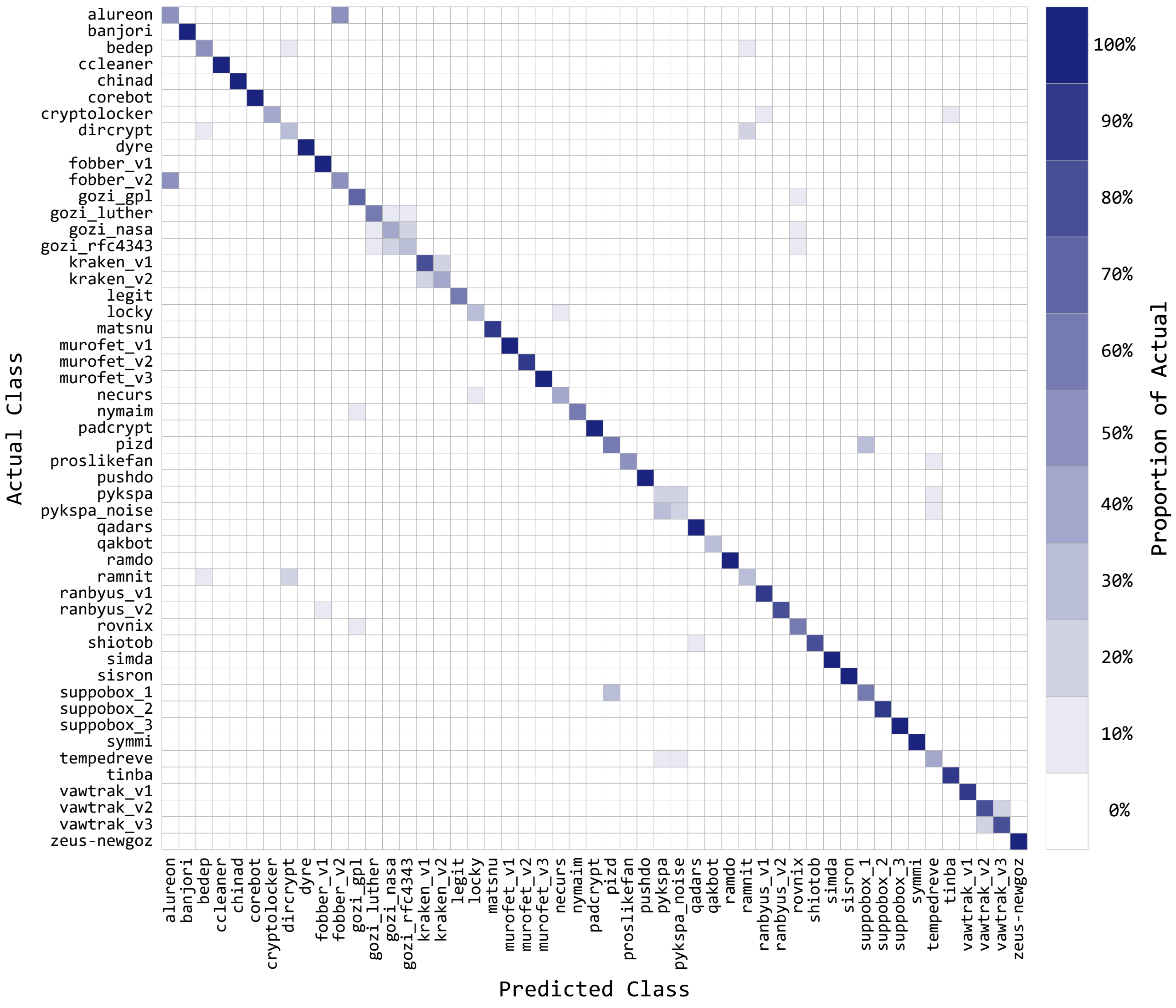
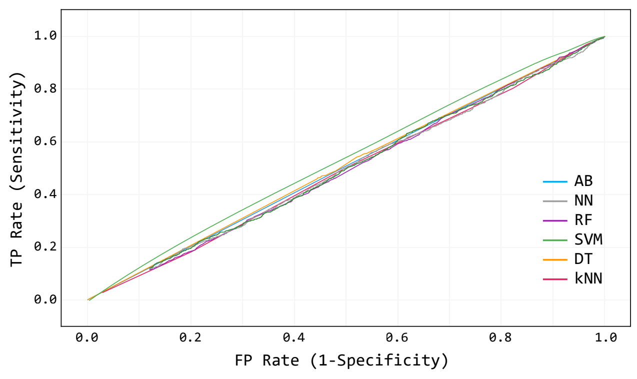


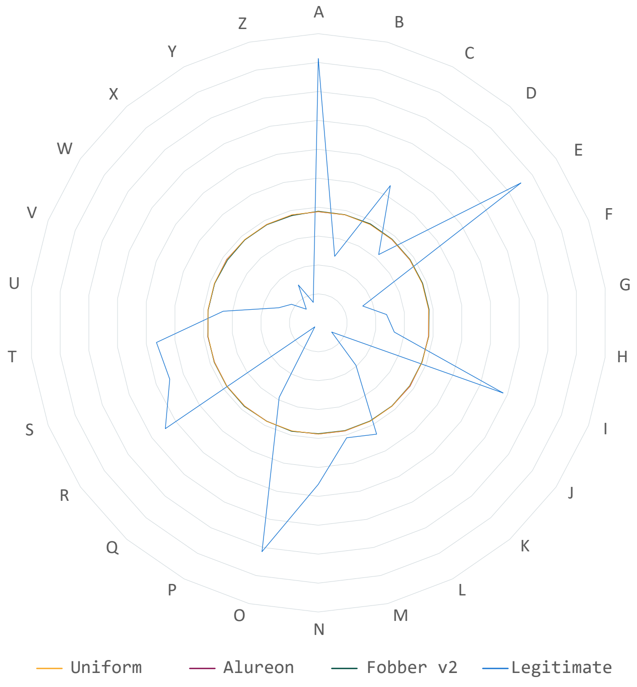
Figure 7: Confusion Matrix for the Multiclass experiment (Exp. 2, Random Forest)



(a) ROC curve

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.504	0.504	0.504	0.504	0.504
NN	0.494	0.494	0.494	0.493	0.494
RF	0.503	0.503	0.503	0.505	0.503
SVM	0.501	0.501	0.501	0.499	0.500
DT	0.500	0.500	0.500	0.500	0.500
kNN	0.502	0.501	0.501	0.502	0.501

(b) Classifiers performances



(c) Distributions comparison

Distribution	Expected		
Observed	Uniform	Alureon	Fobber v2
Alureon	1.000	–	0.979
Fobber v2	0.983	0.979	–

(d) Pearson chi-squared test (χ^2)

Figure 8: Comparative analysis of Fobber (2nd variant) and Alureon

4.1. Fobber (2nd version) versus Alureon

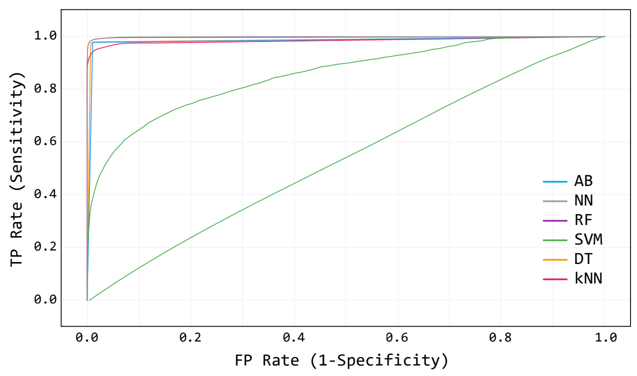
The first cluster of errors belongs to `Fobber (2nd version)` and `Alureon`. Both variants DGAs are publicly available at [15]. The first step is to execute a specific ML experiment oriented toward those two classes, which results are reported in Figure 8a (ROC curve) and Figure 8b (classifiers performances). From these, it is clear that the feature set chosen, in combination with the data and the algorithms configurations do not permit to separate the two classes.

To further explore the two classes, we have printed their 1Gram distributions in Figure 8c together with the English distribution and a uniform one. As suggested by [52], we performed a Pearson's ChiSquare Test to compare them, and the results are reported in Figure 8d. The test does not reject the hypothesis that the two variants belong to the same uniform distribution. Follows that both malware variants have achieved to generate AGDs within a uniform distribution, and thus impossible to distinguish with the current feature set.

Further analysis including, but not limited to, context-aware features might result in an effective way to distinguish the two classes. For example, a double cycle detection as proposed by [51] might be applied in such case. That is to say, both `Fobber (2nd version)` and `Alureon` can be considered as a single class, and then deeply analysed with a specific component once a detection event occurs. In fact, as shown in both Figure 9a and Figure 9b, almost all classifiers achieve extremely high performances while oriented toward distinguishing legitimate domain names from these two joined classes.

4.2. Bedep versus DirCrypt versus Ramnit

The second cluster of classification errors belongs to three classes, namely `Bedep`, `DirCrypt` and `Ramnit`. However, as shown in Figure 10, it appears that this group is driven by the identical distributions of `DirCrypt` and `Ramnit`, which are then very likely, but not identical to `Bedep`.



(a) ROC curve for the `legit` class while compared to `Fobber (2nd version)` and `Alureon`

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.986	0.979	0.978	0.984	0.979
NN	0.990	0.986	0.983	0.999	0.985
RF	0.989	0.986	0.983	0.998	0.984
SVM	0.701	0.736	0.669	0.855	0.701
DT	0.979	0.981	0.977	0.987	0.979
kNN	0.958	0.991	0.927	0.985	0.958

(b) Classifiers performances for the `legit` class while compared to `Fobber (2nd version)` and `Alureon`

Figure 9: Comparative analysis of `Fobber (2nd variant)` and `Alureon` as a single class against the `Legit` one.

The classifiers output for the three malware variants can be found in Figure 10b, and together with the ROC curve analysis (Figure 10a) prove that, within this data, the ML algorithms only take educated guesses over the class, without being capable of actually memorising the class characteristics.

The Pearson's ChiSquare test, presented in Figure 11b, suggests another subgroup formed by `DirCrypt` and `Ramnit`. To analyse and eventually confirm this hypothesis, we conducted a Binary experiment over those two classes, proving that also in this case it is not possible to separate the two classes (Figure 10c and Figure 10d).

However, in Figure 11a it is possible to notice that both `DirCrypt` and `Ramnit` differ from the uniform distribution in a few cases. To be more precise, our implementation of `Ramnit` DGA does not ever produce the letter "`z`" in any domain. This leads to a missing value in the distribution value, which can be mapped as zero. Having one zero in the distribution causes the ChiSquare test to fail for not being defined at zero. This condition, together with the fact that we do not have a specific feature for each character (including "`z`" and "`c`", as highlighted in Figure 11a) permits to not consider the character during this test. The results are available in Figure 11b and, as expected, both three variants achieve to be uniform (or almost uniform).

As suggested in Section 3, a double cycle detection [51] might take advantage of other class-specific characteristics to perform a deeper analysis. For instance, Figure 12 reports the classifier results and the ROC curve for the six classifiers defined earlier in Section 4. It is clear that most classifiers can achieve extremely good performances, and thus can act as a filter before a more deep and accurate analysis is performed.

4.3. Discussion

As previously mentioned in Section 4, only a few clusters have been analysed and reported here. The same analysis however has been carried out for all classification errors. To be more precise, we decided to aggregate the classes that have at least 20% of misclassification between each other. For example, `Fobber (2nd version) and Alureon` share around 50%.

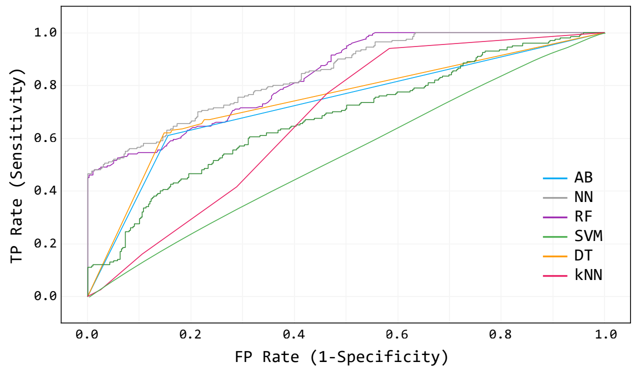
Following what previously suggested throughout this section, an approach like the one defined in [51] might be appropriate to develop a full-fledged detection solution for tackling DGA-based botnets. To validate this hint, Experiment 2 have been executed on a tweaked dataset that considers clusters of classes as classification target instead of malware variants. Figure 13 summarises the results for this scenario and to be more precise, Figure 13b reports the classifiers performances while Figure 13a presents a comparison between the previously shown Multiclass experiment's classifiers' F1 scores (Figure 6b) and the ones obtained by the classifiers in this scenario.

As depicted in Figure 13a there is a general improvement across all the classifiers, which is also clearly visible in the related confusion matrix available in Figure 14.

Once a FQDN is flagged as suspicious, for example by classifying it as in the new aggregated class `Alu-Fobv2` (which comprehends both `Fobber (2nd version) and Alureon`, further analysis can be deployed to perform deep inspection that might allow to pinpoint the exact malware variant, thus enabling the deployment of appropriate countermeasures.

Future researches are, however, required to establish whether an improved *context-free* set of features may be able to distinguish between the clusters of classes or if a deeper analysis based on *context-aware* features is required. Nevertheless, this result, along with the one reported in Figure 5, validates the concept of having a first, high-level filter for detecting AGDs, followed by a malware-specific intrusive inspection technique.

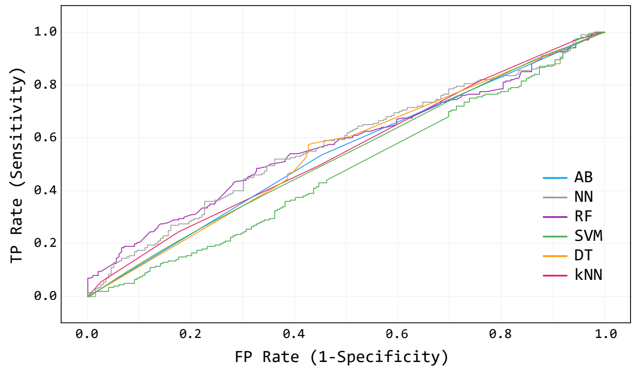
One could argue that this approach, does not rely on studying and developing a ML approach suitable for solving the previously detailed Multiclass experiment (Exp. 2). That would be correct to claim if the target of this article would have included the proposal for a ML-powered detection framework. Nevertheless, in this context, where the proposal is a dataset to enable such analysis and comparison, the above mentioned claim



(a) ROC curve for `Bedep` vs `DirCrypt` vs `Ramnit`

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.525	0.533	0.525	0.644	0.528
NN	0.518	0.525	0.518	0.735	0.521
RF	0.497	0.499	0.497	0.722	0.499
SVM	0.400	0.389	0.400	0.615	0.389
DT	0.522	0.519	0.522	0.656	0.519
kNN	0.425	0.410	0.425	0.605	0.410

(b) Classifiers performances for `Bedep` vs `DirCrypt` vs `Ramnit`

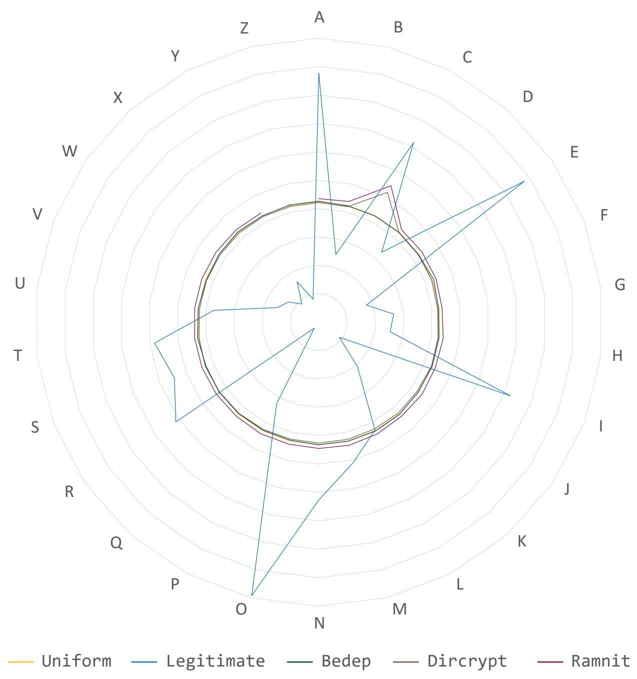


(c) ROC curve for `DirCrypt` vs `Ramnit`

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.525	0.525	0.525	0.549	0.525
NN	0.552	0.553	0.552	0.548	0.552
RF	0.515	0.515	0.515	0.478	0.514
SVM	0.555	0.555	0.555	0.576	0.555
DT	0.557	0.558	0.557	0.572	0.557
kNN	0.542	0.543	0.542	0.542	0.542

(d) Classifiers performances for `DirCrypt` vs `Ramnit`

Figure 10: Comparative analysis of `Bedep`, `DirCrypt` and `Ramnit` — Part 1

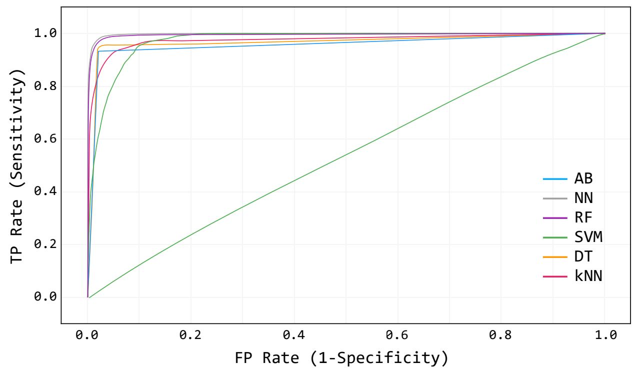


(a) Distributions comparison

Distribution	Expected			
Observed	Uniform	Bedep	DirCrypt	Ramnit[1]
Bedep	1.000	–	0.000	0.000
DirCrypt	0.972	0.993	–	1.000
Ramnit[1]	0.929	0.991	1.000	–

(b) Pearson chi-squared test (χ^2), where [1] indicates that the tests have been conducted excluding the character "`z`"

Figure 11: Comparative analysis of `Bedep`, `DirCrypt` and `Ramnit` — Part 2

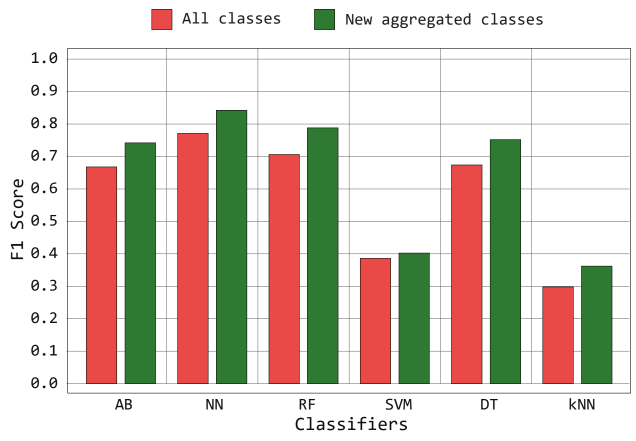


(a) ROC curve for the `legit` class while compared to `Bedep`, `DirCrypt` and `Ramnit`

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.968	0.938	0.932	0.956	0.935
NN	0.980	0.956	0.965	0.998	0.961
RF	0.977	0.944	0.964	0.994	0.954
SVM	0.838	0.834	0.438	0.973	0.575
DT	0.970	0.933	0.947	0.965	0.940
kNN	0.946	0.887	0.898	0.974	0.892

(b) Classifiers performances for the `legit` class while compared to `Bedep`, `DirCrypt` and `Ramnit`

Figure 12: Comparative analysis of `Bedep`, `DirCrypt` and `Ramnit` versus the legitimate class



(a) F1 Score comparison between the results proposed in Figure 6b and the ones proposed in Figure 13b.

Method	Acc.	Prec.	Rec.	AUC	F1
AB	0.739	0.741	0.739	0.866	0.740
NN	0.847	0.845	0.847	0.995	0.840
RF	0.794	0.788	0.794	0.979	0.788
SVM	0.445	0.456	0.445	0.945	0.456
DT	0.751	0.750	0.751	0.886	0.750
kNN	0.382	0.358	0.382	0.792	0.358

(b) Classifiers performances

Figure 13: Results for the Multiclass experiment (Exp. 2) after remapping the classes as described in Section 4.3.

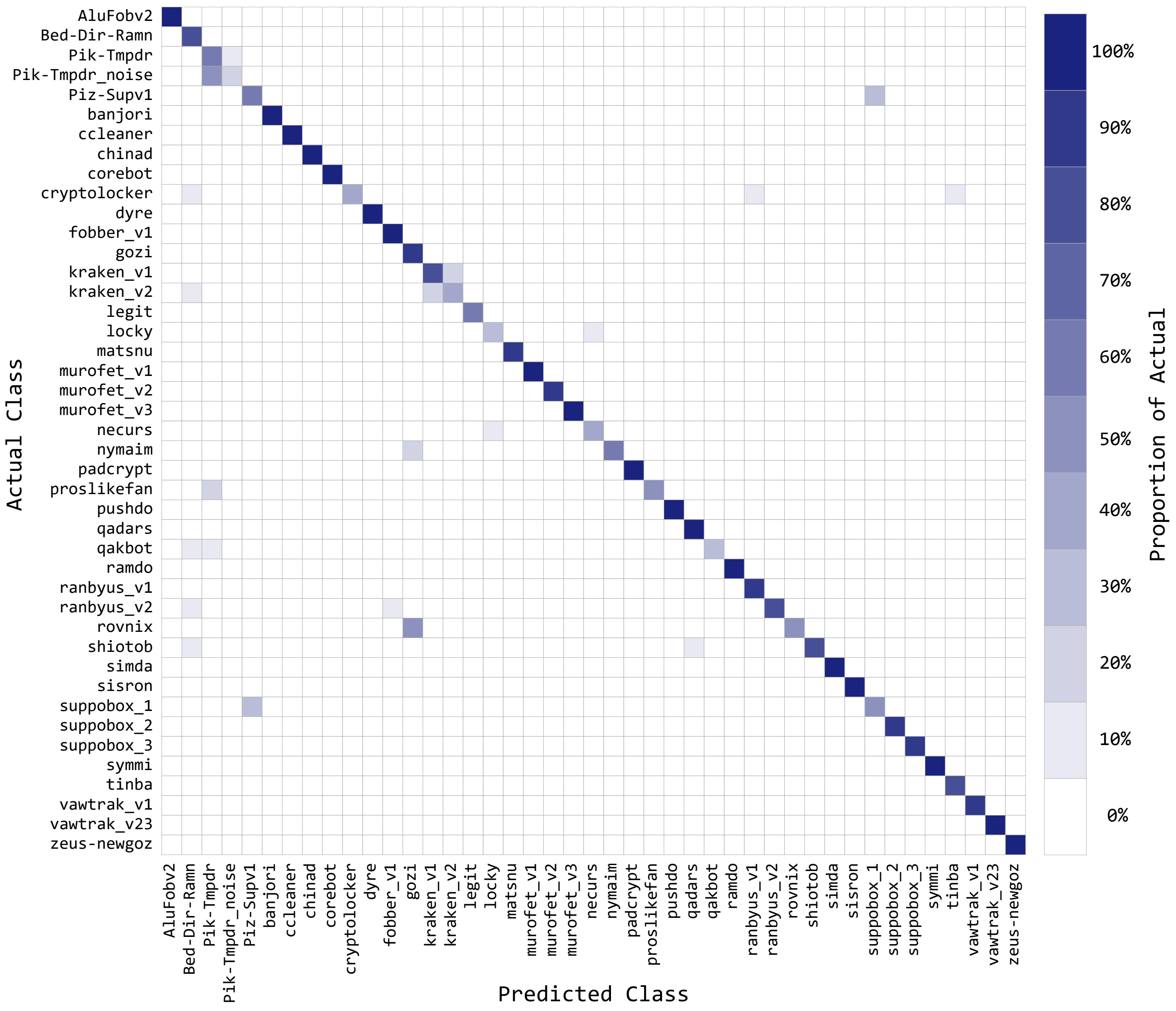


Figure 14: Confusion Matrix for the Multiclass experiment (Exp. 2, Random Forest) with tweaked data as described in Section 4.3.

does not hold. Once again, one of the main contributions of this article is to propose a publicly available dataset that can be used to research, study and deploy new comparable ML-based solutions that do not require harming the users' privacy.

As pinpointed previously by the authors [10], the task of detecting DGA-based botnets without privacy breaches remains an open challenge, especially with the approach of encrypted DNS [53]. Even if some recent studies suggest that Deep Learning (DL) techniques might provide some advantages [54, 55, 56], however, a comparison between these solutions is far to be achievable due to the lack of structured and publicly accessible data.

As previously suggested, with `UMUDGA` we aim to provide data to overcome the lack of ML-ready and publicly available datasets. However, we acknowledge that our proposal of a context-free dataset is just one side of the problem [10] and the study of the state-of-the-art in terms of context-aware features might result in potential game-changing applications. Despite the unquestionable benefits that supplying fresh data to the scientific community produces, there are several areas at which potential future researchers might look. To cite one, ML techniques have been only used to scratch the surface of the problem [10] and further analysis might lead to innovative products.

Finally, we once again would like to remark that, generally, literature solutions are not replicable, let alone deployable in a real world environment [10]. To reach such remarkable objective, future works must:

1. publish the data, a necessary condition to enable the reproducibility of the experiments but also to enable third party future researches to not start from scratch;

2. precisely discuss about initial configurations and subsequent optimisation of applied techniques, including ML algorithms;

3. identify the architecture, the workflow, the environment, the experiments configurations and, in general, provide all the information required to independently redeploy the scenarios and verify the results; and, finally

4. compare the obtained results with the state-of-the-art techniques using reproducible means based, at least, on comparable, if not identical, data sources.

5. Conclusions

Recent technical reports suggest an increasing interest in ML solutions for cybersecurity, and, although their are sold as all-comprehensive panacea, their applications are non-specialist at best. Literature researches show a plethora of shady solutions that claim to achieve almost perfect performances without providing enough means to validate, let alone reproduce, the results. The first and foremost key issue regarding this problem is attributable to the data sources, which are not properly organised or carefully reviewed. In fact, most of the publicly available datasets suffer from important shortcomings that prevents to achieve the required rigorousness, reproducibility and credibility of the research. To the best of our knowledge, Section 2, summarised in Table 1, highlights the well-known properties of the current state-of-the-art in terms of data sources, providing a clear categorisation that may prove useful to future researchers.

As a consequence, we propose our dataset, the University of Murcia Domain Generation Algorithm Dataset (`UMUDGA`) available at Mendeley Data [15], that ultimately achieves all these established properties alongside with a formal and rigorous mathematical data definition [17].

At the same time, it holds that several challenges are yet to be solved. Further researches are, in fact, required to address the problem of DGA-based botnets. Unlike the related works analysed in Section 2, `UMUDGA` aims to address the first of the shortcomings of comparable ML results, *i.e.*, the data source.

Finally, the exploratory analysis shows that data manipulation can easily lead to significant improvements in the performances of any ML solutions, and thus should be strictly documented and justified. Moreover, scientists and future researches should transparently adhere to an experiment protocol that follows predetermined and well-established guidelines, which, to the best of our knowledge, nowadays do not exist. Our proposed guidelines aim to serve as catalyst for creating a standard protocol for ML solutions in network cybersecurity.

Declaration of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work has been supported by a predoctoral and a postdoctoral INCIBE (Spanish National Cybersecurity Institute) grants within the "Ayudas para la Excelencia de los Equipos de Investigación Avanzada en Ciberseguridad" ("Grants for the Excellence of Advanced Cybersecurity Research Teams") Program, with codes INCIBEI-2015-27353 and INCIBEI-2015-27352, respectively.

References

[1] B. O'Gorman, C. Wueest, D. O'Brien, G. Cleary, H. Lau, J.-P. Power, M. Corpin, O. Cox, P. Wood, S. Wallace, Internet security threat report, Tech. rep., Symantec Corporation (2019).
URL `https://www.symantec.com/security-center/threat-report`

[2] A. Kujawa, W. Zamora, J. Umawing, J. Segura, W. Tsing, P. Arntz, C. Boyd, 2019 state of malware, Tech. rep., Malwarebytes LABS (2019).
URL `https://resources.malwarebytes.com/resource/2019-state-malware-malwarebytes-labs-report/`

[3] A. Eremin, What are botnets downloading? Statistics for the past year on files downloaded by botnets, Tech. rep., Kaspersky Labs (2018).
URL `https://securelist.com/what-are-botnets-downloading/87658/`

[4] A. Brandt, B. Cove, C. Yu, C. Wisniewski, G. Szappanos, J. Chandraiah, J. Zhang, J. Levy, P. Kohli, P. MacKenzie, R. Cohen, R. Yu, S. Shevchenko, T. Easton, Sophoslabs 2019 treath report, Tech. rep., SOPHOS Ltd (2018).
URL `https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophoslabs-2019-threat-report.pdf`

[5] Spamhaus Project, Spamhaus botnet threat report 2019, Tech. rep., Spamhaus Project (2019).
URL `https://www.spamhaustech.com/botnet-threat-report-2019/`

[6] Spamhaus Project, Spamhaus botnet threat update: Q1-2019, Tech. rep., Spamhaus Project (2019).
URL `https://www.spamhaus.org/news/article/784/spamhaus-botnet-threat-update-q1-2019`

[7] Fireeye Mandiant Services, M-Trends 2019 special report, Tech. rep., FireEye, Inc. (2019).
URL `https://www.fireeye.com/current-threats/annual-threat-report.html`

[8] N. Etaher, G. R. S. Weir, M. Alazab, From ZeuS to Zitmo: trends in banking malware, in: 2015 IEEE Trustcom/BigDataSE/ISPA, 2015, pp. 1386–1391. `doi:10.1109/Trustcom.2015.535`.

[9] G. Vormayr, T. Zseby, J. Fabini, Botnet communication patterns, IEEE Communications Surveys & Tutorials 19 (4) (2017) 2768–2796. `doi:10.1109/COMST.2017.2749442`.

[10] M. Zago, M. Gil Pérez, G. Martínez Pérez, Scalable detection of botnets based on DGA: efficient feature discovery process in machine learning techniques, Soft Computing, In Press `doi:10.1007/s00500-018-03703-8`.

[11] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, E. Gerhards-Padilla, A comprehensive measurement study of domain generating malware, in: 25th USENIX Conference on Security Symposium, 2016, pp. 263–278.
URL `http://dl.acm.org/citation.cfm?id=3241094.3241115`

[12] C. G. J. Putman, Abhishta, L. J. M. Nieuwenhuis, Business model of a botnet, in: 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing, 2018, pp. 441–445. `doi:10.1109/PDP2018.2018.00077`.

[13] B. Hammi, S. Zeadally, R. Khatoun, An empirical investigation of botnet as a service for cyberattacks, Transactions on Emerging Telecommunications Technologies 30 (3) (2019) 1–11. `doi:10.1002/ett.3537`.

[14] M. Kührer, C. Rossow, T. Holz, Paint it black: evaluating the effectiveness of malware blacklists, in: Research in Attacks, Intrusions and Defenses, 2014, pp. 1–21. `doi:10.1007/978-3-319-11379-1_1`.

[15] M. Zago, M. Gil Pérez, G. Martínez Pérez, UMUDGA - University of Murcia Domain Generation Algorithm Dataset, Mendeley Data (2020). `doi:10.17632/y8ph45msv8`.

[16] M. Zago, M. Gil Pérez, G. Martínez Pérez, UMUDGA - University of Murcia Domain Generation Algorithm Dataset (2020). `doi:10.5281/zenodo.3618221`.
URL `https://github.com/Cyberdefence-Lab-Murcia/UMUDGA`

[17] M. Zago, M. Gil Pérez, G. Martínez Pérez, UMUDGA: A dataset for profiling algorithmically generated domain names in botnet detection, Data in Brief `doi:10.1016/j.dib.2020.105400`.

[18] A. Shiravi, H. Shiravi, M. Tavallaee, A. A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Computers & Security 31 (3) (2012) 357–374. `doi:10.1016/j.cose.2011.12.012`.

[19] S. García, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, Computers & Security 45 (2014) 100–123. `doi:10.1016/j.cose.2014.05.011`.

[20] M. Parkour, Contagio Malware Dump - Collection of Pcap files from malware analysis (2015).
URL `http://contagiodump.blogspot.com/2013/04/collection-of-pcap-files-from-malware.html`

[21] A. D. Kent, Cybersecurity data sources for dynamic network research, in: Dynamic Networks in Cybersecurity, Imperial College Press, 2015, pp. 27–65. `doi:10.1142/9781786340757_0002`.

[22] A. D. Kent, Comprehensive, multi-source cyber-security events, Los Alamos National Laboratory (2015). `doi:10.17021/1179829`.

[23] A. Alenazi, I. Traore, K. Ganame, I. Woungang, Holistic model for HTTP botnet detection based on DNS traffic analysis, in: 1st International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments, 2017, pp. 1–18. `doi:10.1007/978-3-319-69155-8_1`.

[24] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, D. Garant, Botnet detection based on traffic behavior analysis and flow intervals, Computers & Security 39 (2013) 2–16. `doi:10.1016/j.cose.2013.04.007`.

[25] E. B. Beigi, H. H. Jazi, N. Stakhanova, A. A. Ghorbani, Towards effective feature selection in machine learning-based botnet detection approaches, in: 2014 IEEE Conference on Communications and Network Security, 2014, pp. 247–255. `doi:10.1109/CNS.2014.6997492`.

[26] R. Sharma, R. K. Singla, A. Guleria, A new labelled flow-based DNS dataset for anomaly detection: PUF dataset, Procedia Computer Science 132 (2018) 1458–1466. `doi:10.1016/J.PROCS.2018.05.079`.

[27] B. Wylie, SuperCowPowers - data hacking (2013).
URL `https://github.com/SuperCowPowers/data_hacking/tree/master/dga_detection`

[28] A. Abakumov, Andrewaeva/DGA (2014).
URL `https://github.com/andrewaeva/DGA`

[29] P. Chaignon, Pchaigno/DGA_Collection (2015).
URL `https://github.com/pchaigno/dga-collection`

[30] J. Bader, BaderJ - Domain generation algorithm.
URL `https://github.com/baderj/domain_generation_algorithms`

[31] R. Vinayakumar, K. P. Soman, P. Poornachandran, Detecting malicious domain names using deep learning approaches at scale, Journal of Intelligent & Fuzzy Systems 34 (3) (2018) 1355–1367. `doi:10.3233/JIFS-169431`.

[32] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6. `doi:10.1109/CISDA.2009.5356528`.

[33] J. Ma, L. K. Saul, S. Savage, G. M. Voelker, Identifying suspicious URLs: an application of large-scale online learning, in: 26th Annual International Conference on Machine Learning, 2009, pp. 681–688. `doi:10.1145/1553374.1553462`.

[34] M. H. Bhuyan, D. K. Bhattacharyya, J. K. Kalita, Towards generating real-life datasets for network intrusion detection, I. J. Network Security 17 (2015) 683–701. `doi:10.6633/IJNS.201511.17(6).05`.

[35] N. Moustafa, Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic, Ph.D. thesis, The University of New South Wales (2017).

[36] D. Berman, A. Buczak, J. Chavis, C. Corbett, A survey of deep learning methods for cyber security, Information 10 (4) (2019) 1–35. `doi:10.3390/info10040122`.

[37] The Guava Authors, Google Guava - InternetDomainName class (2009).
URL `https://github.com/google/guava/wiki/InternetDomainNameExplained`

[38] Gerald Combs, Wireshark (1998).
URL `https://www.wireshark.org/`

[39] Alexa Internet Inc., Alexa Top Domains.
URL `https://www.alexa.com/topsites`

[40] The Apache Software Foundation, Apache Commons Validator (Feb 2017).
URL `https://commons.apache.org/proper/commons-validator/`

[41] P. Mockapetris, Domain names - concepts and facilities, STD 13, RFC Editor (November 1987). `doi:10.17487/RFC1034`.

[42] R. Braden, Requirements for internet hosts - application and support, STD 3, RFC Editor (October 1989). `doi:10.17487/RFC1123`.

[43] D. Goldhahn, T. Eckart, U. Quasthoff, Building large monolingual dictionaries at the leipzig corpora collection: from 100 to 200 languages, in: 8th International Conference on Language Resources and Evaluation, European Languages Resources Association (ELRA), 2012, pp. 759–765.

[44] Netlab 360, DGA families.
URL `http://data.netlab.360.com/dga/`

[45] Malware domain list (2009).
URL `https://www.malwaredomainlist.com/mdl.php`

[46] OSINT, OSINT DGA list.
URL `http://osint.bambenekconsulting.com/feeds/`

[47] Risk Analytics, DNS-BH - Malware domain blocklist (2007).
URL `http://www.malwaredomains.com`

[48] Majestic-12 Ltd, The Majestic Million.
URL `https://majestic.com/reports/majestic-million`

[49] R. Elz, R. Bush, Clarifications to the DNS specification, RFC 2181, RFC Editor (July 1997). `doi:10.17487/RFC2181`.

[50] J. Demšar, T. Curk, A. Erjavec, Č. Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, M. Štajdohar, L. Umek, L. Žagar, J. Žbontar, M. Žitnik, B. Zupan, Orange: data mining toolbox in python, Journal of Machine Learning Research 14 (2013) 2349–2353.
URL `http://jmlr.org/papers/v14/demsar13a.html`

[51] M. Gil Pérez, A. Huertas Celdrán, F. Ippoliti, P. G. Giardina, G. Bernini, R. M. Alaez, E. Chirivella-Perez, F. J. García Clemente, G. Martínez Pérez, E. Kraja, G. Carrozzo, J. M. Alcaraz Calero, Q. Wang, Dynamic reconfiguration in 5G mobile networks to proactively detect and mitigate botnets, IEEE Internet Computing 21 (5) (2017) 28–36. `doi:10.1109/MIC.2017.3481345`.

[52] D. E. Knuth, The art of computer programming: seminumerical algorithms, 3rd Edition, Vol. 2, Addison-Wesley Longman Publishing Co., Inc., 1997. `doi:10.1137/1012065`.

[53] C. Patsakis, F. Casino, V. Katos, Encrypted and covert DNS queries for botnets: Challenges and countermeasures, Computers & Security 88 (2020) 101614. `doi:https://doi.org/10.1016/j.cose.2019.101614`.

[54] Y. Qiao, B. Zhang, W. Zhang, A. K. Sangaiah, H. Wu, DGA Domain Name Classification Method Based on Long Short-Term Memory with Attention Mechanism, Applied Sciences 9 (20) (2019) 4205. `doi:10.3390/app9204205`.

[55] R. Vinayakumar, K. P. Soman, P. Poornachandran, S. Akarsh, M. Elhoseny, Improved DGA Domain Names Detection and Categorization Using Deep Learning Architectures with Classical Machine Learning Algorithms, Springer International Publishing, 2019, Ch. 8, pp. 161–192. `doi:10.1007/978-3-030-16837-7_8`.

[56] Y. Liang, X. Yan, Using Deep Learning to Detect Malicious URLs, in: 2019 IEEE International Conference on Energy Internet (ICEI), 2019, pp. 487–492. `doi:10.1109/ICEI.2019.00092`.