electronics

MDPI

Article

Detection of DGA-Generated Domain Names with TF-IDF

Harald Vranken [1,2,*] and Hassan Alizadeh [1]

[1] Department of Computer Science, Open Universiteit, P.O. Box 2960, 6401 DL Heerlen, The Netherlands; hassan77.alizadeh@gmail.com
[2] Institute for Computing and Information Sciences, Radboud University, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
* Correspondence: harald.vranken@ou.nl

Abstract: Botnets often apply domain name generation algorithms (DGAs) to evade detection by generating large numbers of pseudo-random domain names of which only few are registered by cybercriminals. In this paper, we address how DGA-generated domain names can be detected by means of machine learning and deep learning. We first present an extensive literature review on recent prior work in which machine learning and deep learning have been applied for detecting DGA-generated domain names. We observe that a common methodology is still missing, and the use of different datasets causes that experimental results can hardly be compared. We next propose the use of TF-IDF to measure frequencies of the most relevant n-grams in domain names, and use these as features in learning algorithms. We perform experiments with various machine-learning and deep-learning models using TF-IDF features, of which a deep MLP model yields the best results. For comparison, we also apply an LSTM model with embedding layer to convert domain names from a sequence of characters into a vector representation. The performance of our LSTM and MLP models is rather similar, achieving 0.994 and 0.995 AUC, and average F1-scores of 0.907 and 0.891 respectively.

Keywords: DGA; botnet; TF-IDF; machine learning; deep learning

Citation: Vranken, H.; Alizadeh, H. Detection of DGA-Generated Domain Names with TF-IDF. *Electronics* **2022**, *11*, 414. https://doi.org/10.3390/electronics11030414

Academic Editors: Constantinos Kolias, Georgios Kambourakis and Weizhi Meng

Received: 1 December 2021
Accepted: 24 January 2022
Published: 29 January 2022

1. Introduction

Botnets pose a severe threat to the security of systems connected to the Internet and their users. A botnet is composed of a collection of compromised systems ('bots') that receive and respond to commands from a Command and Control (C&C) server. A C&C server acts as rendezvous point between the bots and the botmaster, who controls the botnet. By updating the malware running on the bots, the botmaster can configure the botnet to perform different types of attacks, such as launching DDoS attacks, sending spam, or stealing credentials. This versatility causes that botnets are considered as the Swiss army knife of cybercriminals.

C&C servers and the communication channels between botmaster and bots are critical components of a botnet. By taking down the C&C servers, or by blocking the communication channels, the link between bots and botmaster is broken, which renders the botnet useless. Numerous techniques have been applied to provide stealthy botnet operation and to increase resilience against take-down attempts [1]. An eminent technique to evade detection is the application of domain name generation algorithms (DGAs) in bot malware that generate large numbers of pseudo-random domain names for contacting the C&C server, of which only few are actually registered shortly by the botmaster. Due to the dynamic DGA operation and short-lived domain names, the communication between C&C servers and bots is protected against take-down attempts.

The presence of botnets that use DGAs can be revealed by analysing network traffic. For instance, most of the domain names that are generated by DGAs are not registered and hence DNS lookups for resolving such domain names into IP addresses will result in NXDomain responses from name servers. Hence, by monitoring and analysing NXDomain

responses, the presence of DGA-based botnets can be revealed. In addition, the domain names generated by DGAs typically differ from regular domain names. Regular domain names are usually meant to be interpretable by humans and hence they often are rather short and meaningful, while domain names generated by DGAs typically consist of random strings of letters and digits that humans cannot pronounce or interpret as meaningful. Hence, by analysing the syntax or semantics of domain names, the presence of DGA-based botnets can be revealed.

In recent years, numerous methods applying machine learning, and more recently also deep learning, have been explored for detecting DGA-based botnets. Machine-learning algorithms typically have been used to train models using sample data of network traffic in order to make predictions on whether the traffic contains traces originating from botnets. The key principle is that the models are not programmed explicitly up front to detect botnets, but the models evolve during training by discovering patterns in the sample data. The sample data typically consist of features that are derived from captured network traffic, such as relevant fields in packet headers or payload data. In prior work on detection of DGA-generated domain names with machine learning, a large variety of structural, linguistic, and statistical features have been explored that are derived from domain names in DNS traffic. In deep learning, as an advanced form of machine learning, more complex models are trained that can discover higher-level patterns in the sample data. While machine learning requires to provide sample data by means of selected features during training, deep learning is able to implicitly derive features at multiple levels from the sample data.

Furthermore, in this paper, we apply machine learning and deep learning for detecting DGA-based botnets, and use TF-IDF (term frequency–inverse document frequency) as statistical method to derive features from domain names. TF-IDF originates from information retrieval and automated text analysis, where it is used as a weighting factor to evaluate how relevant a word is to a document in a collection of documents [2]. Key words that appear more often but in a smaller number of documents, have higher TF-IDF. We observed that the distributions of characters and n-grams vary considerably for regular domain names and domain names that are generated by different types of DGAs (as we will show in Section 4.1). Our hypothesis therefore is that TF-IDF of n-grams can be used as features for classifying domain names in learning algorithms.

Our contributions are as follows:

1. We provide an extensive literature review on recent prior work in which machine learning and deep learning have been applied for detecting DGA-based botnets.
2. We explore the usage of TF-IDF for feature selection. Although TF-IDF has been studied and applied extensively for decades in information retrieval, we are the first—to the best of our knowledge—to apply TF-IDF for detecting DGA-based botnets.
3. We provide experimental results using TF-IDF as features with the most popular algorithms for machine learning (Decision Tree, Gradient Boosting, K-Neighbours, Logistic Regression, Multinomial Naive Bayes, Random Forest, and Support Vector Machine) and deep learning (Multi-Layer Perceptron).
4. We compare the results obtained by machine learning, in which TF-IDF scores of n-grams in domain names are used as features, with featureless deep learning (using a long short-term memory (LSTM) classifier) in which domain names are embedded as sequences of input characters.

In the remainder of this paper, we first provide more details about botnets and fluxing methods, DGAs, and TF-IDF in Section 2. In Section 3, we present an extensive literature review on recent prior work in which machine learning and deep learning have been applied for detecting DGA-based botnets. In Section 4, we present our research method, including a description of the datasets we applied in our experiments, the setup of our experiments, and the experimental results with discussion. We conclude the paper in Section 5.

2. Background

This section provides more details on fluxing methods as applied by botnets to evade detection in Section 2.1, on DGAs in Section 2.2, and on TF-IDF in Section 2.3.

2.1. Botnets and Fluxing

The bots in a botnet regularly contact their C&C server. This is the case during the rallying process, when a bot tries to contact its C&C server for the first time to announce its presence, and later on when the bot contacts the C&C server to upload data (such as stolen credentials) or to download malware updates. In order to do so, the bot should know either the IP address or the domain name of the C&C server.

The IP address can be hardcoded in the bot malware. This offers stealthy botnet operation since no DNS lookup is required. However, the IP address can easily be revealed by reverse engineering of the malware. Network administrators can subsequently blacklist the IP address in ACLs at gateways, or apply BGP route announcements to route the IP address to a blackhole where the traffic is dropped.

Alternatively, the domain name can be hardcoded in the bot malware. This is less stealthy since it requires a DNS lookup to resolve the domain name into an IP address. To evade detection, botnets can apply IP/fast flux by using dynamic DNS to provide that the domain name can be resolved into an IP address that changes frequently. These IP addresses refer to proxy bots that relay communication between bots and the C&C server. Bringing down the botnet now requires blacklisting or blackholing the IP addresses of all proxy bots. To further evade detection, botnets can apply double flux, where the concept of flux is also applied to the name server that is responsible for resolving the domain name. The name server, which is under control of the botmaster, will refer to frequently changing authoritative name servers, which in turn will resolve the domain name into frequently changing IP addresses of proxy bots. However, the domain name can be blacklisted, or, by applying DNS sinkholing, the domain name can be resolved into an IP address that is not under control of the botnet. DNS sinkholing allows for instance law enforcement agencies to take over the botnet.

Many botnets therefore do not rely on a single domain name, but apply domain flux by generating a large number of domain names of which only few actually are registered by the botmaster for a short time period. Domain flux renders botnet detection by static domain name blacklists or sinkholing ineffective.

2.2. DGA

Domain flux is implemented in bot malware by a DGA that dynamically generates a large number of pseudo-random domain names from a seed. The seed, which acts as a shared secret between botmaster and bots, can be either static or dynamic [3].

A static seed was for instance applied in early versions of the Kraken botnet, and therefore the same set of domain names is generated at each execution [4]. Early versions of the Torpig botnet applied a deterministic seed that is derived from the current date. Since the domain names derived from such deterministic seeds can be precomputed easily, and botmasters do not register all future domain names in advance, a botnet can be taken over. For instance, a research team was able to preregister some domain names and take over the Torpig botnet for 10 days in 2009 [5]. The Conficker botnet also applied a time-dependent seed based on GMT that is derived from the response of querying a public website [6]. The Conficker.C botnet applied domain flux by generating 50,000 domain names of which bots daily tried up to 500 for contacting the C&C server to receive updates. If the botmaster registered one of these 50,000 domain names, bots have 1% probability per day to contact the C&C server, and hence bots would contact the C&C server once every 100 days on average. While the botmaster had to register only one or a few domain names, law enforcement would have to preregister 50,000 domain names to block the C&C communication.

Domain names derived from dynamic seeds rely on non-deterministic sources. For instance, the Bedep DGA applied a seed that relates to foreign exchange reference rates published daily by the European Central Bank, while the seed in later versions of the Torpig DGA related to trending topics in Twitter [7]. Since domain names from non-deterministic seeds cannot be precomputed in advance, blacklisting and sinkholing or preregistering large numbers of short-lived domain names by law enforcement agencies is a challenging, time-critical task that requires continuous effort. However, also botmasters only have a small time window and should switch continuously to new domain names.

Next to classifying DGAs based on their seeding characteristics, Plohmann et al. classified DGAs into 4 types based on how domain names are constructed [7]. Arithmetic-based DGAs (type DGA-A) are most common. They construct domain names by generating sequences of values that have either an ASCII representation directly or index hardcoded arrays that constitute the DGA alphabet. Hash-based DGAs (type DGA-H) construct domain names from hashing algorithms such as MD5 and SHA256. Wordlist-based DGAs (type DGA-W) construct domain names by concatenating sequences of words from dictionaries that are embedded in the malware or obtained from a publicly accessible source. Permutation-based DGAs (type DGA-P) construct domain names through permutation of an initial domain name.

Domain names generated by DGAs of type DGA-A typically consist of random sequences of characters (letters and digits). Domain names generated by DGAs of type DGA-H represent a hexadecimal number and consist of digits and the letters A–F. Domain names generated by DGAs of type DGA-W are less random and pronounceable, which makes them harder to distinguish from regular domain names. In addition, domain names generated by DGAs of type DGA-P, that are derived by permutation of regular domain names, look similar to regular domain names.

RFC 1035 initially specified the preferred syntax of domain names as a sequence of labels separated by dots [8]. The right-most label conveys the top-level domain. Each label is a sequence of at most 63 characters containing letters (A–Z, a–z), digits (0–9), or the hyphen symbol (-), with the restriction that a label starts with a letter and ends with a letter or digit. Although uppercase and lowercase letters are allowed, no significance is attached to the case. The length of a domain name is at most 255 characters. In later specifications, this has been relaxed to labels that contain the underscore symbol (_), leading or trailing hyphens, other ASCII characters (such as the symbols # and $), and even Unicode characters in internationalized domain names [9].

2.3. TF-IDF

TF-IDF originates from information retrieval and automated text analysis, where it is used as a weighting factor to evaluate how relevant a term is to a document in a collection of documents [2]. For instance, TF-IDF is the most popular weighting scheme in recommender systems for research papers that apply content-based filtering [10].

TF-IDF is composed of multiplying term frequency (TF) [11] and inverse document frequency (IDF) [12], where TF indicates how often a term appears in a document, and IDF indicates the number of documents in a corpus that contain the term.

The simplest way to compute TF is the raw count of appearances of term t_i in document d_j, where t_i is in the set of terms $T = \{t_1, \ldots, t_K\}$ in the corpus of documents $D = \{d_1, \ldots, d_N\}$. This can be normalized by considering for instance the length of the document or the most frequent term in the document. IDF adjusts for the general appearance of terms across documents and is usually defined as $\log(N/n_i)$, where N is the number of documents in the corpus and n_i is the number of documents in which term t_i occurs. IDF is close to 0 when the term appears in many documents, and increases when the term appears in fewer documents. Hence, TF-IDF discriminates key terms that appear more often but in a smaller number of documents.

3. Literature Review

We conducted an extensive literature review on recent prior work in which machine learning (ML) and deep learning (DL) have been applied for detecting DGA-based botnets. Zago et al. [13] previously published a literature review on DGA-based botnet detection that covered literature up to May 2018. We extend their literature review by covering 38 additional scientific papers that were published afterwards from May 2018 to February 2021.

Zago et al. built a taxonomy of approaches for botnet detection considering the applied learning approach (supervised, unsupervised, or semi-supervised) and the type of features adopted (context-aware, context-free, or featureless). Context-aware features are dependent on a specific malware sample execution, such as features extracted from DNS responses that consider timing, origin, or any other environment configuration. Context-free features are related only to domain names, considering structural, statistical or linguistic properties of a domain name. Featureless models, as typically applied in DL, do not require features and use encoded domain names as inputs.

We adopt a slightly different taxonomy by considering the learning method, which is either feature-based ML (see Table 1), featureless DL (see Table 2), or other (see Table 3). Nearly all studies included in our review applied supervised learning algorithms, using either classical ML models such as Decision Tree (DT), Random Forest (RF), k-Nearest Neighbour (kNN), Logistic Regression (LR), Naïve Bayes (NB), Support Vector Machine (SVM), Gradient Boost (GB), and Multi-Layer Perceptron (MLP), or novel DL models such as Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN).

The following subsections provide more details on the ML models in Section 3.1, on the DL models in Section 3.2, on other methods in Section 3.3, and on the datasets used in the reviewed studies in Section 3.4.

Table 1. DGA-based botnet detection using feature-based ML models.

Reference	Year	Model	Dataset (Benign/Malicious)	Number of Features	
				Context-Free	Context-Aware
Chiba et al. [14]	2018	RF	Alexa/hpHosts	-	55
Schüppen et al. [15]	2018	**RF**, SVM	Private/DGArchive (72 DGAs)	21	-
Ashiq et al. [16]	2019	FFNN (2-4 hidden layers)	From [17]	8	-
He et al. [18]	2019	Adaboost, DT, kNN, **RF**	Alexa/various sources	21	153
Li et al. [19]	2019	Adaboost, C4.5, **kNN**, NB	.cn name server/Rustock DGA	1	31
Liu et al. [20]	2019	SVM	Alexa/DGArchive (87 DGAs)	-	18
Selvi et al. [21]	2019	RF	Alexa/26 DGAs	18	-
Yang et al. [22]	2019	DT, ET, NB, SVM, **ensemble (NB,ET,LR)**	Cisco Umbrella/Netlab, synthetic	24	-
Akhila et al. [23]	2020	DT, **GBT**, LR, RF, SVM	Alexa/Bambenek	10	-
Alaeiyan et al. [24]	2020	**RF**, RNN, SVM	Alexa/MasterDGA	18	-
Almashhadani et al. [25]	2020	BT, DT, **kNN**, NB, SVM	Alexa/DGArchive (20 DGAs)	16	-
Anand et al. [26]	2020	**C5.0**, CART, GBM, kNN, RF, SVM	Alexa/Netlab (19 DGAs)	45	-
Hwang et al. [27]	2020	LightGBM	KISA/KISA (20 DGAs)	110	-
Liang et al. [28]	2020	**RF**, SVM, XGBoost	Alexa/various blacklists	5	5
Mao et al. [29]	2020	NB, LSTM, **MLP**, RF, SVM, XGBoost	Alexa/Netlab (40 DGAs)	5	-
Palaniappan et al. [30]	2020	LR	Alexa/various blacklists	4	13
Sivaguru et al. [31]	2020	RF	Alexa, private/DGArchive	26	9
Wu et al. [32]	2020	**MLP**, NB	Alexa/Netlab	4	-
Zhang et al. [33]	2020	DT, LR, NB, RF, SVM, **XGBoost**, Voting	Alexa/UMUDGA (37 DGAs)	18	-
Zago et al. [13]	2020	Adaboost, DT, kNN, **NN**, RF, SVM	Majestic/various sources (16 DGAs)	40	-
Cucchiarelli et al. [34]	2021	**MLP**, RF, SVM	Alexa/Netlab (25 DGAs)	$4n + 5$ (n DGAs)	-
Patsakis et al. [35]	2021	RF	Alexa, unipi/DGArchive, synthetic (13 DGAs)	32	-

Table 2. DGA-based botnet detection using featureless DL models.

Reference	Year	Model	Dataset (Benign/Malicious)
Woodbridge et al. [36]	2016	LSTM	Alexa/Bambenek
Lison and Mavroeidis [37]	2017	RNN	Alexa/DGArchive (63 DGAs), Bambenek (11 DGAs)
Koh and Rhodes [38]	2018	LSTM	OpenDNS/Bader, Abakumov
Tran et al. [39]	2018	LSTM.MI	Alexa/Bambenek (37 DGAs)
Vinayakumar et al. [40]	2018	**LSTM**, GRU, IRNN, RNN, CNN, **hybrid (CNN-LSTM)**	Alexa, OpenDNS/Bambenek, Bader (17 DGAs)
Xu et al. [41]	2018	CNN-based	Alexa/DGArchive (16 DGAs)
Yu et al. [42]	2018	LSTM, BiLSTM, stacked CNN, parallel CNN, hybrid (CNN-LSTM)	Alexa/Bambenek
Akarsh et al. [43]	2019	LSTM	OpenDNS, Alexa/20 public DGAs
Qiao et al. [44]	2019	LSTM	Alexa/Bambenek
Liu et al. [45]	2020	Hybrid (BiLSTM-CNN)	Alexa/Netlab (50 DGAs), Bambenek (30 DGAs)
Ren et al. [46]	2020	CNN, LSTM, CNN-BiLSTM, **ATT-CNN-BiLSTM**, SVM	Alexa/Bambenek, Netlab (19 DGAs)
Sivaguru et al. [31]	2020	hybrid (RF-LSTM.MI)	Alexa, private/DGArchive
Vij et al. [47]	2020	LSTM	Alexa/11 DGAs
Cucchiarelli et al. [34]	2021	BiLSTM, LSTM.MI, hybrid (CNN-BiLSTM)	Alexa/Netlab (25 DGAs)
Highnam et al. [48]	2021	hybrid (CNN-LSTM-ANN)	Alexa/DGArchive (3 DGAs)
Namgung et al. [49]	2021	CNN, LSTM, BiLSTM, **hybrid (CNN-BiLSTM)**	Alexa/Bambenek
Yilmaz et al. [50]	2021	LSTM	Majestic/DGArchive (68 DGAs)

Table 3. DGA-based botnet detection using other approaches.

Reference	Year	Dataset (Benign/Malicious)
Wang et al. [51]	2017	Emulated and real-life network data
Satoh et al. [52]	2020	Alexa/19 DGAs
Sun et al. [53]	2020	Alexa/various
Yan et al. [54]	2020	Passive DNS data/public blacklists
Yin et al. [55]	2020	Alexa/Bader (19 DGAs)

3.1. ML-Based Methods

3.1.1. Models

The studies in our literature review apply a range of ML methods, as shown in Table 1. RF and SVM are applied most often. For each study, the best performing model is shown in bold. It can be observed that there is no single best performing method overall, but RF and MLP (with a single hidden layer) give best results in most cases. Due to differences in the applied hyperparameters, features, datasets of benign and malicious domain names, and evaluation metrics, it is rather unfeasible to compare the experimental results obtained in these studies.

3.1.2. Context-Free Features

Zago et al. [13] focused in their literature review on context-free features. They identified 74 features of 32 types, although we found that their overview is somewhat inaccurate. Most of the studies cited by Zago et al. use string metrics as features in which a domain name is considered as a string of characters or words. Most frequently used are string length (in 69% of the cited works) and entropy (46%). Some features relate to linguistics, such as the pronounceability score (13%) and normality score (13%), while other features capture more complex structural aspects of domain names, such as the Jaccard Index measure (17%) and the Kullback–Leiber divergence (8%). Zago et al. identified features related to length, ratio and sequence of digits, and frequencies and pronounceability of n-grams as most relevant. They observed that arbitrary combinations of features are used in most studies, often with different names or definitions, and hence a common ground for features is still missing.

As a follow-up of the literature review by Zago et al., our literature review covered 22 more recent studies that applied ML-based methods (see Table 1). Nearly all studies (19 out of 22) use context-free features that relate to domain names.

As shown in Table 4, we identified 97 context-free feature types that we categorised as being related to the domain name, subdomains, character-level information, linguistics, and n-grams. Compared to the 32 feature types that were identified by Zago et al. [13] up to early 2018, during the next two years researchers explored 65 additional feature types. We observe however that arbitrary combinations of features have been used. Domain name length, number of subdomains, alphabet cardinality, entropy, and the ratios of digits and vowels are used most frequently.

Table 4. Context-free features.

Feature Type	Usage
Domain name	
Domain name in Unicode format	[29]
Domain name length	[13,15,16,18,23,25–28,31–33]
Domain name contains IP address	[15,18]
Domain name has www prefix	[15,18]
Domain name starts with digit	[31]
Domain name has valid TLD	[15,18]
Characters	
Shanon entropy	[15,18,21,23,25,25,27,28,31,33]
Cardinality (number of different characters)	[15,18,21,31–33]
Ratio of unique characters (Gini value)	[23,26,31]
Minimum character value	[24]
Maximum character value	[24]
Average character value	[24]
Variance of character values	[24]
Median of character values	[24]
Number of non-occurring character values	[24]
Number of odd character values	[24]
Chi-squared test of character values	[24]
Reverse character values	[24]
Ratio of most frequently used letters	[26]
Ratio of least frequently used letters	[26]
Ratio of meaningful characters	[13,31]
Ratio of repeated characters	[15,18,26,31]
Ratio of underscores	[15,18]
Number of hyphens	[31]
Number of underscores and hyphens	[30]
Number of dashes	[24]
Ratio of consonants	[13,21,25,31,33]
Ratio of consecutive consonants	[15,18,31]
Number of consonants	[25,26]
Longest consecutive consonant sequence	[13,25]
Ratio of length of longest consonants sequence	[25]
Contains digits	[15,18]
Ratio of digits	[13,15,18,21,23,31,33]
Ratio of consecutive digits	[15,18,31]
Number of digits	[27,30,31]
Longest consecutive digit sequence	[13]
Ratio of vowels	[13,15,18,23,25,27,28,31–33]
Number of vowels	[25–27]
Longest consecutive vowel sequence	[13,25]
Ratio of length of longest vowels sequence	[25]
Ratio of vowels to consonants	[16,21,25]
Ratio of length of longest consonants sequence to length of longest vowels sequence	[25]
Ratio of vowels and digits	[29]
Contains digits and letters	[27]
Transition frequency of digits and letters	[28]
Ratio of longest sequence lengths of letters or digits	[19,23]
Subdomains	
Number of subdomains	[13,15,18,24,27,28,30]
Subdomain length mean	[15,18]
Contains single-character subdomain	[15,18]
Ratio of subdomains that contain digits only	[15,18]
Ratio of subdomains that contain hexadecimal numbers only	[15,18]
Contains TLD as subdomain	[15,18]
Binary encoded TLD	[31,34]
TLD type	[27,31]
TLD length	[31]
Number of different characters in TLD	[31]
Second-level domain type	[27]
Second-level domain length	[31]
Number of different characters in second-level domain	[31]
Linguistics	
Regularity score (edit distance)	[16]
Levenshtein edit distance	[13,23]
Jaccard index	[23]
Randomness	[16,25]
Number of 'word-like' units	[13]
Ratio of meaningful substrings	[13,16,23]
Length of longest meaningful string	[13]
Frequency score (frequency of words)	[16]
Correlation score (between consecutive words)	[16]
Pronounceability score	[23,24]
Number of syllables	[24]
Illegitimate contents (from word list)	[30]
Domain squatting score	[13]
Language hypothesis	[13]

Table 4. *Cont.*

Feature Type	Usage
n-grams	
n-gram frequency distribution	[13,15,18,24,32]
n-gram mean	[13,21,26,33]
n-gram median	[13,31]
n-gram variance	[13,21,26]
n-gram standard deviation	[13,21,26]
n-gram distance	[29]
n-skip-gram distribution	[24]
Number of most frequently used n-grams	[26,33]
Number of masked n-grams	[26]
Ratio of 4-grams without vowel	[16]
Ratio of n-grams from benign domains	[33]
Ratio of n-grams from malicious domains	[33]
n-gram entropy	[13]
n-gram covariance	[13]
n-gram pronounceability score	[13]
n-gram normality score	[13]
n-gram transition probability	[13]
n-gram probability of appearance	[13]
n-gram index probability	[13]
n-gram Kullback–Leiber divergence	[13,34]
n-gram Jaccard Index measure	[13,34]
n-gram distance-threshold	[13]
n-gram distance–avg. frequency	[13]
n-gram distance–avg. count	[13]

A limited number of studies also used features derived from statistics of n-grams. Next to n-grams, Selvi et al. [21] also used masked n-grams in which every character is substituted by a symbol representing the character type (consonant, vowel, digit, other). Alaeiyan et al. [24] also consider the distribution of n-skip-grams, in which n centre characters are removed in a sequence of adjacent characters (for $n = 1, 2$).

Yang et al. [22] and Patsakis et al. [35] both focus on wordlist-based DGAs and use a large number of features that try to distinguish wordlist-based malicious and benign domain names (excluded from Table 4). Yang et al. use 24 features based on word frequency, part-of-speech frequency, inter-word correlation, and inter-domain correlation. Patsakis et al. use 32 features that consider alphanumeric sequences, statistical and lexical characteristics, and entropy.

Hwang et al. [27] used 10 context-free features and in addition they extracted 100 features using a TextCNN. The TextCNN takes as input a 70×100 matrix for each domain name, constructed by taking 100 characters from the domain name (using truncation for longer domain names and padding for shorter domain names) and one-hot encoding with a dictionary of 70 characters. The TextCNN is composed of two convolutional and max pooling layers with ReLU activation function, three dense layers with ReLU activation function, and dropout.

3.1.3. Context-Aware Features

Of the 22 ML-based studies in our literature review, five studies used a combination of context-free and context-aware features [18,19,28,30,31], and two studies used context-aware features only [14,20], as indicated in Table 1.

Chiba et al. [14] used 55 context-aware features: 20 features reflect how and when a domain name is included in evolving lists of popular and malicious domain names in a certain time window; 18 features consider information from BGP prefixes, ASN, and IP address registration corresponding to the related IP addresses of a domain name; eight features consider relations between domain names of which IP addresses are in the same ASN (so called rDomains). They also use nine features that relate to domain names in rDomains. He et al. [18] used 153 context-aware features: 25 features derived from five feature types that consider DNS information; 128 features obtained from graph embedding to estimate the likelihood of a specific sequence of connected nodes, using a domain relationship graph in which domain names are connected if they are mapped to the same IP addresses using the Jaccard coefficient as weight. They also use 41 context-free features,

which are the same as used by Schüppen et al. [15]. Li et al. [19] focused on the Rustock botnet that applies fast-flux and DGA. They used 31 context-aware features of eight feature types that relate to DNS, and one context-free feature (the ratio of the number of characters in the longest successive string of letters or digits and the total length of the domain name). Liang et al. [28] used five context-aware features that relate to DNS and BGP, and five context-free features. Palaniappan et al. [30] used 13 context-aware features: six DNS-based features and seven web-based features that relate to the web site for which the domain name provides the URL. They also used four context-free features. Sivaguru et al. [31] used nine DNS-based features.

There are two studies, by Schüppen et al. [15] and Liu et al. [20], that focus completely on monitoring of non-existent domain (NXDomain) responses in DNS traffic. Schüppen et al. [15] apply ML for classifying NXDomain responses as originating from benign or malicious sources. Benign NXDomains can originate from either typing errors due to users that misspell existing domain names, misconfigurations due to systems that erroneously try to resolve domain names that do not exist (anymore), and misuse due to non-intended DNS usage such as probing to detect DNS hijacking attempts or anti-virus software performing signature checks. They apply 21 context-free feature types related to domain names. Liu et al. [20] apply filtering to remove benign NXDomain responses using a whitelist, and clustering to group malicious domain names from the same DGA considering the DNS behaviour of hosts. Next, they apply statistical analysis on the clusters considering the distributions of the DNS querying time, count, and domains, from which 18 context-aware featured are derived.

3.2. DL-Based Methods

3.2.1. CNN Models

Xu et al. [41] apply a CNN-based method called n-CBDC (n-gram Character-Based Domain Classification). A sliding window is applied to obtain a sequence with length l of n-grams from a domain name. The sequence of n-grams is represented in a $n \times l$ matrix with one-hot encoding. A CNN layer is used for feature extraction by stacking multiple convolution kernels of different sizes using an inception-like structure. The CNN output is fed into a fully-connected classification network consisting of three layers with dropout. The output is derived from a sigmoid function.

3.2.2. RNN Models

RNN models for detecting DGAs have been applied in several studies. RNN models in general are composed of an embedding layer to transform a domain name into a vector representation, an LSTM layer for implicit feature extraction, and a dense output layer. Dropout is usually applied to prevent overfitting.

Woodbridge et al. [36] and Akarsh et al. [43] apply the Keras embedding layer that learns a 128-dimensional vector representation for each character in the set of valid domain characters. The output of the embedding layer is fed into an LSTM layer with 128 LSTM units for implicit feature extraction. A dropout layer is added to prevent overfitting during training. The final dense layer applies logistic regression and the output is derived from a sigmoid (for binary classification) or softmax (for multi-class classification) activation function.

Lison and Mavroeidis [37] apply one-hot input representation, a layer with 512 GRU or LSTM units, a dense output layer that takes a linear combination, and a sigmoid activation function to generate the output.

Koh and Rhodes [38] apply the pretrained ELMo (Embeddings from Language Models) word embedding layer, a fully-connected layer with 128 rectified linear units (ReLUs), and a logistic regression output layer.

Qiao et al. [44] apply an input layer where a domain name (using a fixed length of 54 characters with padding or truncation) is converted into a matrix of dimension 54×128 using Word2Vec's CBOW model. An LSTM layer with an attention mechanism is used that

gives different attention to different parts of the input domain name, followed by a fully connected layer, dropout, and a softmax classification function.

Vij et al. [47] use an embedding layer where each character is mapped onto a vector with 128 dimensions using a lookup. An LSTM layer with 128 units is added and dropout is used for preventing overfitting.

Yilmaz et al. [50] use an embedding layer where characters are encoded by their ASCII representation. An LSTM layer with two hidden layers is used with dropout to avoid overfitting.

Tran et al. [39] use an embedding layer that projects a padded sequence of input characters of length l to a sequence of vectors with dimension $128 \times l$. LSTM.MI is used, where the original LSTM is adapted to be cost-sensitive for dealing with multi-class imbalance. Sivaguru et al. [31] use a hybrid model where the output of the LSTM.MI model by Tran et al. is used, together with context-free and context-aware features, as input for a B-RF classifier that consists of 100 trees, where each tree is trained using a subset of the feature space.

3.2.3. Hybrid CNN-RNN Models

Several studies have applied hybrid models in which CNN and RNN models are combined.

Vinayakumar et al. [40] use a hybrid CNN-LSTM model. When compared to RNN, LSTM, GRU, I-RNN, and CNN models, best results are obtained with the LSTM and hybrid CNN-LSTM model, achieving over 0.99 accuracy. The accuracy with ML methods (Adaboost, DT, LR, ME, NB, RF) and hand-crafted features is below 0.96.

Yu et al. [42] compared LSTM, BiLSTM, stacked CNN, parallel CNN, and hybrid CNN-LSTM models. All these models performed equally well and obtained over 0.98 accuracy. For comparison, the accuracy of ML methods (RF and MLP) with lexical features is below 0.92.

Liu et al. [45] use a hybrid RCNN-SPP model that combines a bi-directional LSTM network, a CNN, and spatial pyramid pooling.

Highnam et al. [48] use a hybrid CNN-LSTM-ANN model. The output of the embedding layer is passed to separate LSTM and CNN models in parallel. The features extracted by the LSTM and CNN models are sent to a single layer ANN, which is then flattened to produce the output.

Ren et al. [46] compare CNN, LSTM, CNN-BiLSTM, ATT-CNN-BiLSTM, and ML (SVM) models. Best results are obtained with the ATT-CNN-BiLSTM model that is composed of an embedding layer, a CNN layer to extract local parallel features, a BiLSTM layer to extract features that depend on neighbouring characters or on characters that are wider apart, an attention layer, dropout, and an output layer.

Namgung et al. [49] compare CNN, LSTM, BiLSTM, and hybrid CNN-BiLSTM models. In the hybrid model, the output of the embedding layer is sent to a CNN and a BiLSTM with attention in parallel, which subsequently feed into a fully-connected output layer using ReLU and dropout.

Cucchiarelli et al. [34] compare LSTM.MI from Tran et al. [39], BiLSTM from Mac et al. [56], hybrid ATT-CNN-BiLSTM from Ren et al. [46], and ML (MLP, RF, SVM) models. The best accuracy is obtained by a MLP with one single hidden layer composed by 128 units. Although previous studies [40,46] showed that DL methods outperform basic ML methods, Cucchiarelli et al. [34] show that ML methods with careful feature selection and classifier tuning can still outperform DL methods.

3.3. Other Methods

Of the other methods, that are not based on ML or DL, Wang et al. [51] and Yin et al. [55] focused on NXDomain responses. Wang et al. first filter 'normal' NXDomain responses, next cluster hosts that seem compromised by the same DGA-based malware, and finally identify compromised hosts using a supervised statistical algorithm based on query time

and query count distributions. Yin et al. implemented client-side detection by using Threshold Random Walk for sequential hypothesis testing that relies solely on benign domains.

Satoh et al. [52] filter benign domain names using whitelists, select the longest subdomain and split it into words using dictionaries, and estimate the randomness of character strings. To compensate for deficiencies of the dictionaries, they also estimate the randomness of a subdomain by referring to web search results.

Sun et al. [53] applied a graph convolutional network method considering the character distribution of domain names, resources aggregation of attackers, and the query behaviour of clients. The DNS context is modelled as a Heterogeneous Information Network (HIN) of clients, domains, IP addresses, and different types of relations among them. Meta-paths are elaborately extracted to help uncover higher-level semantics hiding in the HIN. A graph convolutional network (GCN) is used that applies an attention mechanism to adaptively learn the meta-paths.

Yan et al. [54] applied graph analysis in a semi-supervised learning scheme. They first extract three types of feature vectors: vectors that represent visiting patterns of domain names in traffic during a fixed time frame are extracted using a CNN-based auto-encoder; vectors that represent the visiting order of domain names are extracted using an embedding scheme where a series of domain names is considered as a series of words in NLP; vectors that represent lexical features of domain names are extracted using an LSTM. These three vectors are combined into a comprehensive feature vector for each domain name. Graph analysis algorithms are used next to group domain names from the same DGA family. By considering thresholds for the number of domain names visited, the most visited domain name, the dispersion of the length of visited domain names, and the dispersion of time intervals in which domain names are visited, it is determined whether a host is infected.

3.4. Datasets

The 42 studies in our literature review as listed in Tables 1–3 used various datasets with benign and malicious domain names to obtain experimental results.

The Alexa dataset https://www.alexa.com/topsites has been used most frequently (in 74% of the reviewed studies) as a source of benign domain names. The Alexa dataset contains the domain names of the most popular web sites in the world. The Alexa dataset with the top 1 million sites was freely available until 2016; afterwards only the top 500 web sites has been freely available. Occasionally other datasets have been used, such as the Majestic dataset https://majestic.com/reports/majestic-million, OpenDNS https://www.opendns.com/, and Cisco Umbrella https://s3-us-west-1.amazonaws.com/umbrella-static/top-1m.csv.zip.

Several sources have been used for DGA-generated domain names. The dataset from Bambenek http://osint.bambenekconsulting.com/feeds/, DGArchive https://dgarchive.caad.fkie.fraunhofer.de, and Netlab https://data.netlab.360.com/feeds/dga/dga.txt have been used most frequently (in 26%, 21%, and 17% of the reviewed studies). The Bambenek dataset is an OSINT DGA domain feed from Bambenek Consulting. DGArchive, as introduced by Plohmann et al. in 2016 [7], originally contained lists of domain names generated by 43 DGA families, and has been extended later on.

4. Research Method

We applied the method as outlined in Figure 1. We trained and tested different multi-class classification models, using both ML and DL, for which we applied TF-IDF features. We also trained and tested an LSTM model without TF-IDF that contains an embedding layer to convert domain names. We consider such LSTM model as state-of-the-art, and hence, by comparing the results obtained with the LSTM model and the TF-IDF based models, we can evaluate the effectiveness of TF-IDF based models.

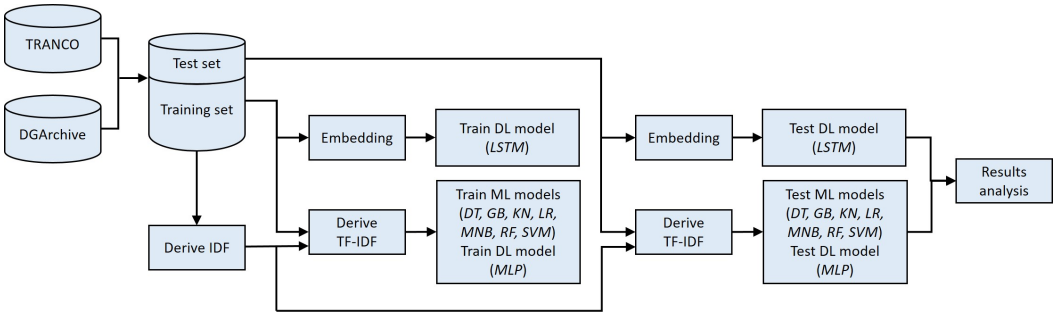


Figure 1. Method.

In the following subsections we provide details on the datasets in Section 4.1, on the usage of TF-IDF in Section 4.2, on the ML and DL models in Section 4.3, on the evaluation metrics in Section 4.4, on the experimental results in Section 4.5, and a discussion on the experimental results in Section 4.6.

4.1. Datasets

We obtained datasets with benign domain names and malicious domain names as generated by DGAs from public sources.

We derived our dataset with benign domain names from the TRANCO list of the top one million most popular domains on the web https://tranco-list.eu. The TRANCO list is based on available rankings from Alexa, Cisco Umbrella, Majestic, and Quantcast, but improves upon each of these rankings by addressing agreement on the set of popular domains, stability over time by averaging the rankings over the past 30 days, popularity and availability of the listed websites, and lack of malicious domains [57]. We performed the following operations:

- We downloaded 10 separate TRANCO lists during 2019 on 8 January, 5 February, 7 May, 4 June, 2 July, 8 August, 3 September, 1 October, 5 November, and 3 December, and took the intersection of these 10 lists.
- We removed domain names that were also in the list of malicious domain names.

Our final dataset contains 583,954 benign domain names. The length of these domain names varies from 4 to 67 characters.

We derived our dataset with malicious domain names from DGArchive, a free service offered by Fraunhofer FKIE [7]. We downloaded the DGArchive dataset at the end of 2018, which contained 110,497,746 malicious domain names from 87 DGA families. We performed the following operations on this dataset:

- We removed duplicate malicious domain names per DGA family and across DGA families.
- We removed domain names that are also in the list of benign domain names.
- We removed the DGA families for which less than 1000 domain names are available.
- We selected all domain names for the DGA families for which between 1000 and 10,000 domain names are available.
- We randomly selected 10,000 domain names from each DGA family for which more than 10,000 domain names are available.

Our final dataset contains 492,800 malicious domain names from 57 DGA families.

Table 5 shows details of the malicious domain names per DGA family. The column 'DGA type' indicates whether the DGA is arithmetic-based (A), hash-based (H), or wordlist-based (W), see Section 2.2. The column 'Count' indicates the number of domain names; the column 'Length' indicates the length (min, max) of the domain names. The last two columns show examples of domain names.

Table 5. Characteristics of DGA families.

DGA Family	DGA Type	Count	Length	Sample 1	Sample 2
banjori	A	10,000	11–30	eihspartbulkyf.com	ochqfordlinnetavox.com
bedep	A	7458	16–22	vhljakiutpq7.com	csejdvmqgmqj.com
chinad	A	10,000	19–21	3vainry4stex8arf.cn	vfuupsix5ki5omg0.cn
conficker	A	10,000	8–16	qzvwnnije.biz	dovcujbpg.biz
corebot	A	10,000	15–32	kr105hivgrqvo8e8ijqh1bc.ws	i472uvy6qjyvgh18mhw4k85.ws
cryptolocker	A	10,000	15–21	leojfthetfvk.com	thtatcpfomflk.com
dnschanger	A	10,000	14–14	xxxfuhkjzu.com	viwnolcsqf.com
ebury	A	2000	17–18	r2g1v3mau7h4k.info	k1i5q3w5r1x4i.net
emotet	A	10,000	19–19	iqpucsfnnijdnbii.eu	olahnvuhbiitauve.eu
fobber	A	2000	14–21	phtatognxg.com	vzuopketsrtaqttgk.net
gameover	A	10,000	18–37	iz6tx9jwre387brksimxpkcp.net	d2u8ds1aif9oryzft8f1u052m5.org
locky	A	10,000	8–23	viuoabuc.fr	rkwaoicjullpc.click
murofet	A	10,000	13–21	prkwwoswewwkfzuy.com	udumozptkqqpo.info
murofetweekly	A	10,000	35–51	jyi35d10gwgqlrmrhupudxdqoyc69n40d20dq.ru	buiuj26gvhxk57pvmrk17d50bwfzlxa17hrls.ru
necurs	A	10,000	10–28	yaatqhjjgicemhoeiu.nf	inlclnelid.ug
nymaim	A	10,000	8–16	xhhtaldw.net	uckvk.net
oderoor	A	3833	10–16	uyftputndw.cc	mdnaizofvm.cc
padcrypt	A	10,000	19–24	fkaokkbfaalfbdeb.info	menccfmdkcmaemfk.de
proslikefan	A	10,000	9–17	zrimegy.in	vnmwww.co
pushdo	A	10,000	11–16	katcetutyx.kz	lakeotux.kz
pushdotid	A	6000	13–14	gxmdgfmjcx.com	opgrexsbif.net
pykspa	A	10,000	10–17	rldbwwarp.net	myhmexr.net
pykspa2	A	10,000	10–19	iugzosiugkeq.net	wkuglwiugkeq.biz
pykspa2s	A	9957	10–19	pkpycifox.com	wudmdgeoya.biz
qadars	A	10,000	16–16	ysmoq4esi0q0.org	gt6b8tirkh2r.net
qakbot	A	10,000	12–30	xvvluuabuftqilmnynimpipb.info	tugfpmprjspprbwxdzi.biz
ramdo	A	6000	20–20	skuqesksmewsckwg.org	iqgieiyuigamowca.org
ramnit	A	10,000	11–25	ixrghbaytyaksgug.com	bwqkmskfwpvljd.com
ranbyus	A	10,000	17–21	ndgpkwlmftaryloae.cc	gttfhnegjtmegkhrt.cc
rovnix	A	10,000	21–22	jaitc336ybcds71ykg.cn	oar7juqajea1wnyopo.cn
shifu	A	2331	10–12	vhqrdfg.info	xxuissv.info
simda	A	10,000	8–14	rynezev.info	qebol.eu
sisron	A	8800	16–17	mjcwmziwmtqa.net	mjmwotiwmtga.net
sphinx	A	10,000	20–20	libuybegcrlrfyof.com	oixwkitoiqseltry.com
sutra	A	9882	19–29	gweqifjejtoaemgw.info	hpwazeehjwpfwgaj.ru
symmi	A	10,000	17–24	oqmievkeedloovm.ddns.net	esitkoelmei.ddns.net
szribi	A	10,000	12–12	ddpuuddd.com	grawspwe.com
tempedrevetdd	A	1380	12–14	gbuxwrwx.org	crwhchuda.org
tinba	A	10,000	10–23	bcjwxxumttmh.net	rwtopxoocwtt.cc
tofsee	A	3140	10–11	drndrng.biz	drodroi.biz
torpig	A	10,000	11–13	bfcmulj.net	bhksvgrpa.com
urlzone	A	10,000	8–19	ehw5jdkwkv.com	rc5iycl4suf.com
vawtrak	A	2700	10–15	dmzqvyn.top	misohnatl.com
vidro	A	10,000	11–23	prjbemepgzkp.com	rakrfxs.com
virut	A	10,000	10–10	yzraho.com	ehuquf.com
xxhex	A	4400	12–13	xxa5c1b019.sg	xx3603da38.sg
bamital	H	10,000	36–38	43f3d094f08dd1a2df2869352e2a9712.cz.cc	f0b79a9253cf7c58f0e1f54426f45bf4.cz.cc
dyre	H	10,000	37–37	rdf36ed41339f9abd57a5a1c9f2143f513.ws	u28c43d53bb3ecafbdfd29fa34a47dae09.to
ekforward	H	2919	8–11	80a118c7.eu	9356c774.eu
infy	H	10,000	12–14	1e60c5f5.space	a56bc6c6.top
pandabanker	H	10,000	16–17	52efedef74d4.com	0b16dca48547.com
tinynuke	H	10,000	36–36	ec893776679264b90cfff916cc5f0eaf.com	84b4a55d8ac046a9816dda8b866893b7.top
wd	H	10,000	36–38	wd679ab775d15bbee733b8545f20452504.win	a0e433f4c96c6b8f3ece607d791d6546.pro
gozi	W	10,000	15–29	formsworkfreeall.com	allowdisalloallow.me
matsnu	W	10,000	16–28	bitpersuadebutton.com	structuresurvey.com
nymaim2	W	10,000	11–33	sculpturenegative.net	shuttlefatty.it
suppobox	W	10,000	11–30	senseinto.ru	threeslept.net

In total, our dataset contains 1,076,754 domain names. For our experiments, we divided the dataset into two disjoint subsets: 70% is included in the training dataset for training the models (containing 753,727 domain names), and 30% is included in the test dataset for evaluating the trained models (containing 323,027 domain names).

The distribution of the average frequency of unigram occurrences in malicious domain names per DGA family as well as in benign domain names from the topsites, is shown in Figure 2. We derived these distributions from the domain names in the training dataset as follows: Since domain names are case insensitive, we first lowercased all domain names. The resulting domain names contain characters from a dictionary of 39 characters $\{a, \ldots, z, 0, \ldots, 9, -, ., \_\}$ (i.e., letters, digits, hyphen, dot, and underscore). Next, we computed the relative frequency of each character, which is the number of occurrences of the character divided by the length of the domain name. Finally, we computed the average frequency of each character for each category of domain names (which is either a DGA family or topsites).

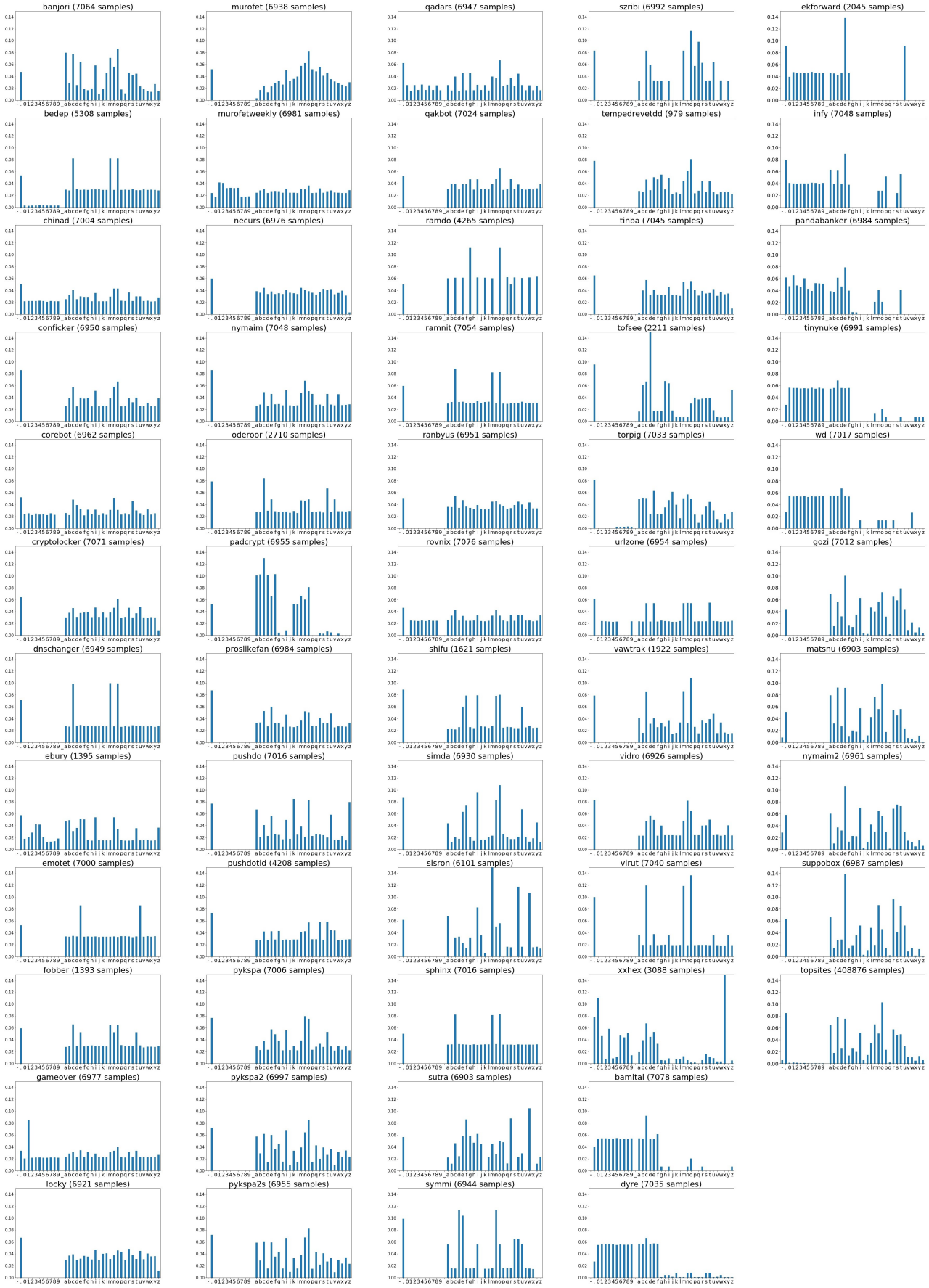


Figure 2. Unigram distributions.

The figure shows that different types of DGA families have different distributions. Of the DGA-A families (banjori to xxhex), some have a rather uniform distribution (e.g., chinad and gameover), but in most distributions no digits are present. Since the second-level domain names of DGA-H families (bamital to wd) are hexadecimal numbers, the distributions mainly contain characters $\{a, \ldots, f, 0, \ldots, 9\}$; other characters are rare and occur in the top-level domain name. Since the domain names of DGA-W families (gozi to suppobox) contain words, the distributions are similar to the distribution of the domain names of topsites. However, this also holds for some of the distributions of DGA-A families (e.g., banjori and pykspa2).

The differences in distributions show that these can be used as a basis for multi-class or binary classification of domain names. Hence, classifiers in which features are applied that rely on n-gram distributions (such as TF-IDF as we propose in this paper), look promising.

4.2. TF-IDF

We use TF-IDF (as explained in Section 2.3) as the single feature type in various classifiers. In general, TF-IDF yields a weighting factor for each term which indicates how relevant the term is to a document in a collection of documents. We apply TF-IDF to obtain weighting factors to evaluate how relevant n-grams are to a domain name in a set of domain names.

We applied the code as shown in Listing 1 to derive the TF-IDF for the top 5000 n-grams in the training dataset, using the class TfidfVectorizer from scikit-learn. We first determine the top 5000 of n-grams (for $n \in \{1, 2, 3\}$) that occur most often in the training dataset, and learn the IDF for each of these n-grams (i.e., the inverse frequency of each n-gram in the domain names in the training dataset). The training dataset is transformed from a set of domain names into a set of vectors with dimension 5000 representing the TF-IDF of the top 5000 n-grams. Next, the TF-IDF is determined for these 5000 n-grams in the test dataset. Hence, also the test dataset is transformed from a set of domain names into a set of vectors with dimension 5000 representing the TF-IDF of the top 5000 n-grams, using the vocabulary and IDF of n-grams derived from the training dataset. TF-IDF for n-gram x in domain name y is computed by Equation (1), where $\text{TF}_{x,y}$ denotes the frequency of x in y, DF_x the number of domain names containing x, and N the total number of domain names.

$$\text{TF-IDF}_{x,y} = (1 + log(\text{TF}_{x,y})) \times log(\frac{N}{\text{DF}_x}) \quad (1)$$

Listing 1. Deriving TF-IDF.

```
tfidf = TfidfVectorizer(sublinear_tf=True, lowercase=True, analyzer='char', ngram_range=(1,3), dtype=np.float32, max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

4.3. ML and DL Models

As indicated in our literature review in Section 3.1 (see in Table 1), different ML models have been applied in prior studies for DGA-based botnet detection, including Decision Trees, K-Nearest Neighbours, Logistic Regression, Naive Bayes, Neural Networks, Support Vector Machine, and ensemble approaches constituted by boosting (such as AdaBoost, BT, C5.0, GB, GBM, GBT, XGBoost) or bagging (such as CART, ET, RF). RF and SVM are applied most often. In line with these prior studies, we considered 7 ML models, all with TF-IDF feature vectors: Decision Tree (DT), Gradient Boosting (GB), K-Neighbours (KN), Logistic Regression (LR), Multinomial Naive Bayes (MNB), Random Forest (RF), and Support Vector Machine (SVM). All models are multi-class classifiers with 58 outputs (corresponding to the 57 DGA families and topsites).

We applied TensorFlow with Keras and Scikit-learn in our experiments. We tuned the hyperparameters for each ML model using RandomizedSearchCV on the training set with 2-fold cross-validation and 20 iterations. The hyperparameters involved are shown in

Table 6. In each iteration, a different hyperparameter setting was tried. The hyperparameter settings that yielded the best classification results (by means of the mean F1-score) are shown in bold.

Table 6. Values of hyperparameters tried in randomized search.

Classifier	Parameter	Candidate Values
DT	min_samples_split	**2**, 5, 10
	min_samples_leaf	1, 2, **4**
	max_features	auto, sqrt, log2, **None**
	max_depth	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, **110**, None
	criterion	**gini**, entropy
GB	subsample	0.5, 0.7, **1.0**
	n_estimators	10, 100, **1000**
	max_depth	**3**, 7, 9
	learning_rate	0.001, **0.01**, 0.1
KN	n_neighbours	1, 3, **5**, 7
	metric	euclidean, manhattan, **minkowski**
	weights	uniform, **distance**
LR	solver	newton-cg, lbfgs, liblinear, sag, **saga**
	penalty	l1, **l2**, elasticnet, none
	C	100, **10**, 1.0, 0.1, 0.01
	class_weight	balanced, **None**
MNB	alpha	1, 0.1, **0.01**, 0.001, 0.0001, 1e-05, 1e-06, 1e-07, 1e-08, 1e-09, 0
	fit_prior	**True**, False
RF	n_estimators	10, 132, 255, 377, **500**
	min_samples_split	2, **5**, 10, 15, 20
	min_samples_leaf	1, **2**, 5, 10, 15
	max_features	**auto**, sqrt, log2
	max_depth	10, 20, 30, 40, 50, 60, **70**, 80, 90, 100, 110, None
	criterion	**gini**, entropy
	bootstrap	True, **False**
SVM	kernel	poly, **rbf**, sigmoid
	gamma	**scale**, auto
	C	50, **10**, 1.0, 0.1, 0.01

We also considered two DL models: Multi-Layer Perceptron (MLP) with TF-IDF feature vectors, and Long Short-Term Memory (LSTM) with embedding (without TF-IDF). For training the DL models, we used 'Adam' as optimizer, 'categorical_crossentropy' as loss function, and 'accuracy' as metric. The 'validation_split' is set to 0.3, which means that 70% of the training set is used for training and 30% for validation. We trained the DL models for 20 epochs with a batch size of 512. We used an early stopping mechanism, configured by EarlyStopping(monitor = 'val_categorical_crossentropy', patience = 5) which means that training is stopped after five epochs without improvement.

We tried MLP models using two to five dense layers, with and without dropout. All models are multi-class with 58 outputs (corresponding to the 57 DGA families and topsites). Best results (by means of mean F1-score) were obtained using the model as shown in Table 7.

Table 7. Details of MLP model.

Layer	Type	Output Shape	#Params
dense_10	Dense (ReLU activation, L2 regularization 0.0001)	352	1,760,352
dropout_2	Dropout (0.2)	352	0
dense_11	Dense (ReLU activation, L2 regularization 0.0001)	352	124,256
dropout_3	Dropout (0.2)	352	0
dense_12	Dense (ReLU activation, L2 regularization 0.0001)	352	124,256
dropout_4	Dropout (0.2)	352	0
dense_13	Dense	58	20,474
Total params	2,029,338		
Trainable params	2,029,338		
Non-trainable params	0		

For comparison, we also applied an LSTM model without TF-IDF. We used an embedding layer to convert a domain name into a vector representation. We derived a dictionary of 39 characters and assigned a unique id to each character. We lowercased and tokenized each domain name into a sequence of characters, which we next transformed into a numeric vector by assigning the character id's from the dictionary. We used vectors with a fixed length of 67. For domain names containing less than 67 characters, we padded the corresponding vector with 0 values. For domain names that contained more than 67 characters, we discarded the extra characters. In addition, our LSTM model is multi-class with 58 outputs (corresponding to the 57 DGA families and topsites). Details of the model are shown in Table 8.

Table 8. Details of LSTM model.

Layer	Type	Output Shape	#Params
embedding_2	Embedding	67,128	5120
lstm_2	LSTM	128	131,584
dropout_2	Dropout (0.1)	128	0
dense_2	Dense	58	7482
activation_2	Activation (softmax)	58	0
Total params	144,186		
Trainable params	144,186		
Non-trainable params	0		

4.4. Metrics

We evaluated the accuracy of our models by considering how well the models are able to correctly classify domain names. Consider for instance the actual positive class of domain names that are generated by the banjori DGA, and the actual negative class of all other domain names (including benign domain names and domain names generated by other DGAs). We can express the classification results, i.e., the predictions as output by a model, by considering the positive predictions (i.e., domain names that are classified as being generated by the banjori DGA) and the negative predictions (i.e., domain names that are classified as not being generated by the banjori DGA), and whether the predicted class matches the actual class. A classification result is either a true positive (TP) in case of a correct positive prediction, a false positive (FP) in case of an incorrect positive prediction, a true negative (TN) in case of a correct negative prediction, or a false negative (FN) in case of an incorrect negative prediction. The ratio between these classifications can be expressed by different metrics. Common metrics are precision, which is the fraction of correct positive predictions among all correct and incorrect positive predictions ($TP/(TP+FP)$) and recall, which is the fraction of correct positive predictions among all elements in the actual positive class ($TP/(TP+FN)$). Precision and recall may not be particularly useful when used in isolation, since deviations in FP and FN may lead to considerable differences between recall and precision. This is addressed by the F1-score, which takes the harmonic mean of precision and recall:

$$\text{F1-score} = \frac{2}{\text{precision}^{-1} + \text{recall}^{-1}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})} \tag{2}$$

We evaluated all our DL and ML models using the held-out test set and computed the F1-score. For further analysis of the results, we also considered confusion matrix, precision-recall curve, and ROC-curve of each model as metrics. In a confusion matrix, each row represents an actual class, while each column represents a predicted class. A precision-recall curve shows the trade-off between precision and recall for different settings of the classification threshold. An ROC-curve shows the true positive rate against the false positive rate at various settings of the classification threshold.

4.5. Results

Table 9 shows the F1-score for different ML and DL models. The corresponding results for precision and recall are shown in Tables A1 and A2 in Appendix A. The last two columns show the number of vectors in the training set and the test set for each class. The top rows show the results for each DGA family and for the benign domain names ('topsites'). The bottom rows show the aggregated results for all the 57 DGA families, for the arithmetic-based DGAs (DGA-A: banjori to xxhex), the hash-based DGAs (DGA-H: bamital to wd), and the wordlist-based DGAs (DGA-W: gozi to suppobox), and the overall results (for all malicious and benign domain names). The aggregated results are expressed as the average results ('macro average'), the weighted average results (where the weights correspond to the number of vectors in the test set; 'micro-average'), and the corresponding standard deviations.

Table 9 shows that best results overall are obtained with the LSTM model, yielding 90.69% weighted average F1-score. The MLP model performed second best (89.08%), closely followed by the SVM model (88.08%). The LR model (86.30%) performed better than the GB (81.02%), DT (80.52%), MNB (80.01%), and RF (78.54%) models, while the KN model (63.00%) performed worst.

Although the LSTM model yielded best results, there are some notable exceptions. The best results for the DGA-W families were obtained with the SVM model, with the MLP model as second best. For the topsites, best results were obtained with the MLP model, with the SVM model as second best.

The aggregated results in Table 9 also show that the best performing models in terms of the highest average F1-score also have the smallest standard deviation and hence the smallest spread in F1-scores. Nevertheless, the standard deviations of 14.68 for the LSTM model and 17.17 for the MLP model still are rather large.

The spread in F1-scores is also shown in the boxplots in Figure 3. The LSTM model clearly performs best, with the MLP model as second best, when considering all malicious and benign domain names ('Total'), all malicious domain names ('DGA-all'), and the domain names generated by arithmetic-based DGAs ('DGA-A'). All models, except the KN model, perform rather well for domain names generated by hash-based DGAs ('DGA-H'). The SVM model performs best for domain names generated by wordlist-based DGAs ('DGA-W').

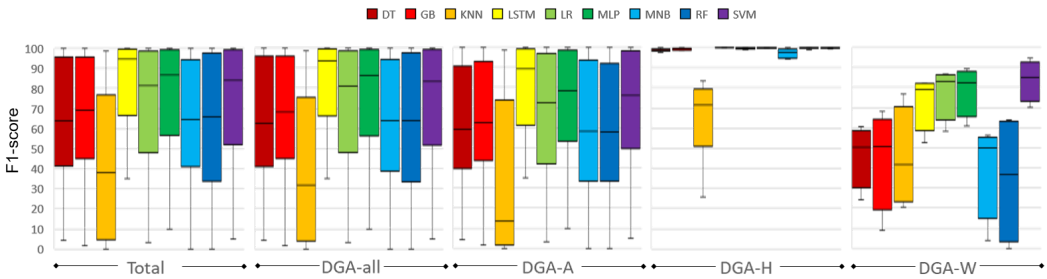
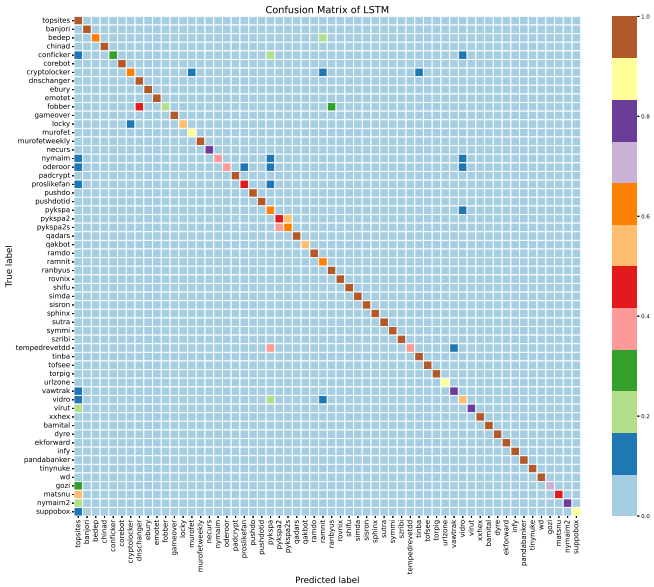


Figure 3. Boxplots of F1-score.

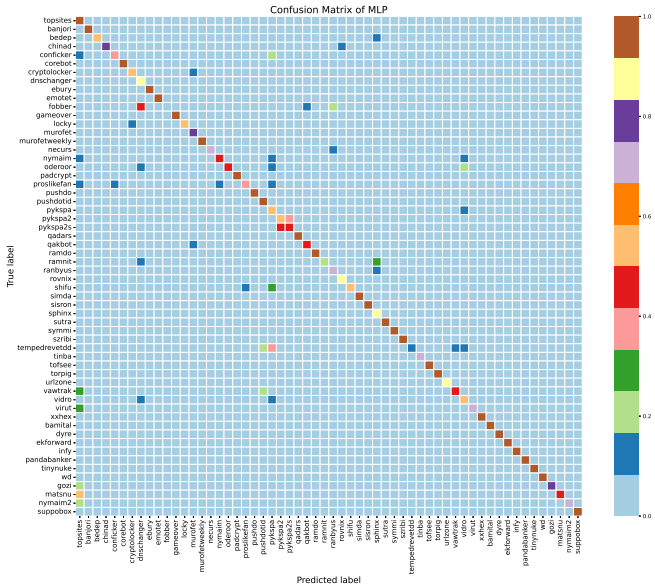
Table 9. F1-score (expressed as percentage).

Class			DT	GB	KN	LSTM	LR	MLP	MNB	RF	SVM	Test	Training
banjori			94.44	98.36	98.39	99.61	99.52	99.81	95.34	99.14	**99.83**	2936	7064
bedep			39.81	49.20	0.37	**73.56**	51.59	61.79	47.06	33.79	60.99	2150	5308
chinad			69.43	67.35	0.73	**99.20**	71.48	84.81	50.73	67.83	79.01	2996	7004
conficker			22.91	29.42	5.63	**39.41**	29.80	38.44	19.28	22.92	35.91	3050	6950
corebot			85.97	88.34	51.26	**99.79**	99.31	99.47	98.38	90.95	99.32	3038	6962
cryptolocker			40.58	45.69	1.59	**67.25**	39.70	56.77	32.42	38.24	48.80	2929	7071
dnschanger			46.70	44.18	1.16	**83.09**	34.69	69.13	20.80	37.50	52.53	3051	6949
ebury			54.86	61.85	91.60	**99.17**	92.09	93.23	94.32	57.27	93.57	605	1395
emotet			96.75	94.16	51.11	**99.60**	94.11	97.84	86.82	95.06	96.54	3000	7000
fobber			22.29	20.51	0	**35.16**	3.26	9.78	0	15.87	4.77	607	1393
gameover			90.66	94.25	0.33	**98.91**	88.04	97.61	65.28	95.75	96.10	3023	6977
locky			41.64	49.96	10.43	**65.12**	52.86	55.95	45.33	44.60	53.10	3079	6921
murofet			58.64	63.57	0.96	**80.46**	60.17	72.03	48.46	59.36	67.62	3062	6938
murofetweekly			95.67	99.14	87.27	99.90	99.83	99.87	99.33	99.68	**100**	3019	6981
necurs			64.92	71.43	0.65	**87.60**	77.42	82.01	70.44	58.88	80.34	3024	6976
nymaim			31.88	46.78	8.10	**51.44**	41.05	48.65	24.51	43.32	45.77	2952	7048
oderoor			26.07	33.13	**79.78**	48.40	73.84	51.78	75.71	23.55	73.45	1123	2710
padcrypt			90.59	92.51	66.46	**99.93**	99.71	99.72	93.37	97.97	99.71	3045	6955
proslikefan			34.18	41.66	5.12	**53.09**	42.31	48.61	27.03	33.32	46.23	3016	6984
pushdo			80.58	88.14	72.87	95.51	**96.15**	96.03	91.92	83.93	95.92	2984	7016
pushdotid			39.80	42.21	11.45	**97.32**	80.68	86.51	75.07	30.41	83.38	1792	4208
pykspa			24.12	29.96	6.90	**50.19**	30.23	43.36	21.26	31.00	39.96	2994	7006
pykspa2			47.77	45.34	45.67	50.44	47.84	**53.63**	46.29	49.77	49.67	3003	6997
pykspa2s			44.57	52.60	44.39	**56.05**	47.83	48.88	43.20	51.82	49.28	3002	6955
qadars			74.80	80.75	76.80	**99.35**	98.74	98.58	98.09	81.19	99.26	3053	6947
qakbot			45.03	50.22	0.33	**62.91**	39.97	55.45	33.84	46.59	45.53	2976	7024
ramdo			98.13	95.46	31.20	**100**	96.57	99.40	85.40	97.47	98.08	1735	4265
ramnit			25.57	18.12	1.43	**52.11**	27.71	29.54	27.71	9.86	28.12	2946	7054
ranbyus			65.72	70.86	2.70	**86.24**	65.21	74.88	48.60	69.00	71.71	3049	6951
rovnix			77.11	79.17	0.48	**99.42**	76.26	87.69	62.85	80.88	82.10	2924	7076
shifu			38.26	39.55	6.04	**79.83**	25.93	52.68	5.41	21.90	31.80	710	1621
simda			75.92	73.49	71.36	**97.54**	95.67	96.08	93.67	72.26	96.77	3070	6930
sisron			99.93	**100**	98.65	**100**	99.98	99.96	99.96	**100**	99.98	2699	6101
sphinx			59.85	58.30	5.06	**91.31**	59.20	71.04	45.82	54.77	63.85	2984	7016
sutra			96.18	97.32	92.36	**99.75**	98.56	98.66	94.30	97.44	99.10	2979	6903
symmi			99.79	99.69	94.94	**100**	99.92	99.98	98.92	99.80	99.98	3056	6944
szribi			93.45	77.81	84.22	**98.11**	96.37	97.86	91.95	90.92	98.01	3008	6992
tempedrevetdd			4.32	1.63	0.88	**48.41**	5.33	15.75	0	0	9.50	401	979
tinba			53.51	44.48	15.51	**82.83**	61.59	74.44	53.45	33.14	68.07	2955	7045
tofsee			97.92	99.57	63.91	**99.09**	98.67	99.30	97.15	97.93	98.93	929	2211
torpig			80.26	94.29	85.99	98.58	98.31	98.71	94.24	87.46	**98.85**	2967	7033
urlzone			62.70	70.60	2.44	**93.68**	81.70	87.25	63.93	68.05	85.03	3046	6954
vawtrak			16.20	17.40	20.43	**76.49**	57.21	57.91	34.07	9.30	56.29	778	1922
vidro			44.19	48.07	50.30	**50.93**	46.57	48.43	47.96	48.83	50.64	3074	6926
virut			42.56	38.00	6.39	**77.88**	40.47	69.43	23.14	17.21	49.93	2960	7040
xxhex			99.05	99.81	96.47	99.73	**100**	99.96	98.68	**100**	**100**	1312	3088
bamital			99.47	99.66	60.29	**99.98**	**99.98**	99.91	96.61	**99.98**	**99.98**	2922	7078
dyre			98.86	99.75	25.73	**100**	99.98	99.98	94.65	**100**	**100**	2965	7035
ekforward			97.60	98.73	79.34	**99.94**	99.83	**99.94**	99.32	99.66	99.71	874	2045
infy			98.40	96.95	71.62	**99.90**	98.91	99.63	97.75	99.12	99.37	2952	7048
pandabanker			98.81	99.16	74.40	**99.93**	99.32	99.78	98.35	99.62	99.62	3016	6984
tinynuke			99.27	99.67	51.10	**99.98**	99.88	99.97	94.50	99.95	99.97	3009	6991
wd			99.93	**100**	83.63	**100**	99.97	99.98	99.92	**100**	**100**	2983	7017
gozi			52.23	68.12	51.84	76.58	85.19	84.32	52.08	64.01	**87.60**	2988	7012
matsnu			24.07	8.81	31.71	52.62	58.17	60.77	4.04	0.06	**70.30**	3097	6903
nymaim2			48.09	49.45	20.46	81.86	81.28	80.34	47.64	12.92	**82.33**	3039	6961
suppobox			60.63	52.11	76.83	82.17	86.83	89.37	56.42	60.76	**94.61**	3013	6987
topsites			92.34	91.70	82.76	96.78	96.58	**97.24**	93.12	89.74	96.94	175,078	408,876
DGA	macro	average	64.43	66.26	39.95	**82.76**	72.51	77.59	63.38	62.84	75.73	147,949	344,851
		stddev	28.23	28.25	35.50	**20.13**	27.88	24.10	31.71	32.30	26.02		
	micro	average	66.54	68.38	39.62	**83.49**	74.14	79.42	64.51	65.29	77.60		
		stddev	26.32	26.28	35.03	**19.36**	25.63	21.72	29.79	30.46	23.45		
DGA-A	macro	average	60.77	63.14	35.87	**80.97**	67.90	74.10	60.25	59.69	71.38	117,091	272,790
		stddev	27.35	27.06	36.93	**20.82**	28.55	24.93	31.22	30.58	26.82		
	micro	average	63.46	65.95	35.48	**81.93**	69.68	76.24	61.86	63.02	73.42		
		stddev	25.19	24.59	36.68	**20.04**	26.28	22.51	28.88	28.16	24.21		
DGA-H	macro	average	98.91	99.13	63.73	**99.96**	99.70	99.88	97.30	99.76	99.81	18,721	44,198
		stddev	0.70	0.97	18.64	**0.04**	0.39	0.12	1.99	0.30	0.23		
	micro	average	99.05	99.18	62.01	**99.96**	99.68	99.88	97.07	99.77	99.82		
		stddev	0.58	1.01	18.86	**0.04**	0.40	0.13	1.97	0.32	0.24		
DGA-W	macro	average	46.26	44.62	45.21	73.31	77.87	78.70	40.05	34.44	**83.71**	12,137	27,863
		stddev	13.58	21.87	21.44	12.15	11.55	10.84	21.02	28.34	**8.88**		
	micro	average	46.09	44.34	45.05	73.18	77.72	78.57	39.79	34.09	**83.61**		
		stddev	13.66	21.97	21.42	12.23	11.62	10.90	21.15	28.37	**8.92**		
Total	macro	average	64.91	66.70	40.69	**83.00**	72.92	77.93	63.89	63.31	76.10	323,027	753,727
		stddev	28.22	28.20	35.63	**20.04**	27.81	24.02	31.68	32.21	25.94		
	micro	average	80.52	81.02	63.00	**90.69**	86.30	89.08	80.01	78.54	88.08		
		stddev	21.97	21.24	32.00	**14.68**	20.64	17.17	24.69	23.94	18.57		

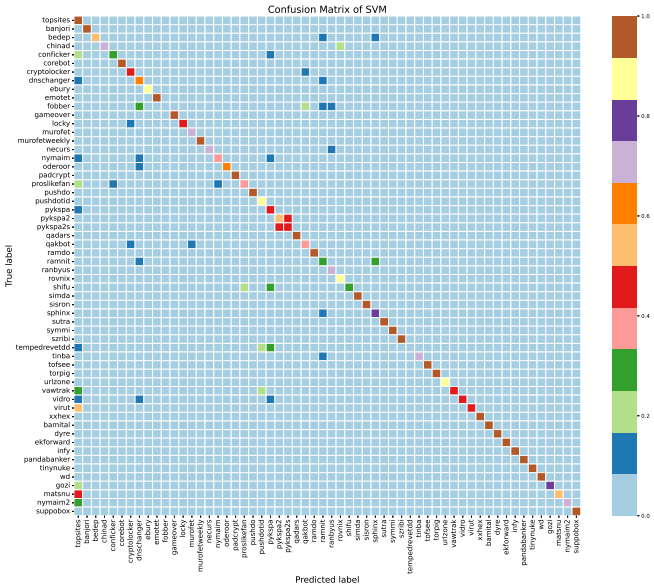
Figure 4 shows the confusion matrices of the LSTM, MLP, SVM, and LR models. All matrices show that false classifications of DGA-W domain names are classified as topsites. As observed in Section 4.1, the DGA-W domain names closely resemble benign domain names. In the LSTM and MLP models, false classifications are often classified as topsites, pykspa and vidro, while in the SVM and LR models, false classifications are often classified as topsites, dnschanger, and ramnit.



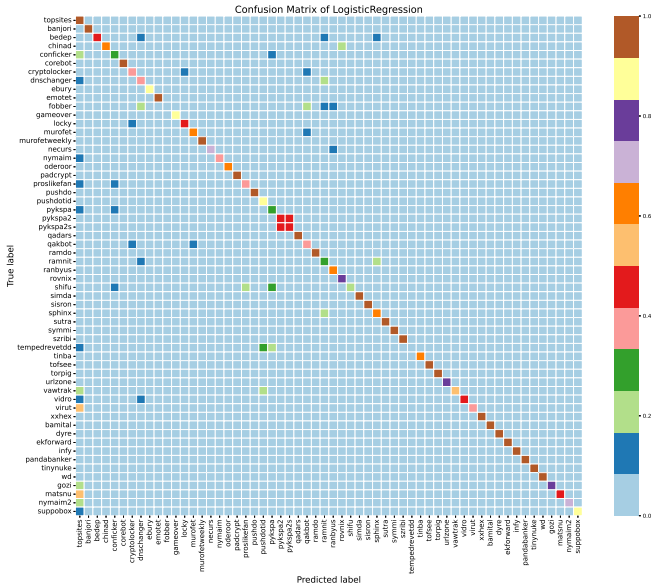
(**a**) Confusion matrix for LSTM model



(**b**) Confusion matrix for MLP model



(**c**) Confusion matrix for SVM model



(**d**) Confusion matrix for LR model

Figure 4. Confusion matrices.

Figure 5 shows the precision-recall curves for the micro-average of all classes. This figure confirms that the LSTM model performs best overall, yielding 0.974 area under the curve, closely followed by the MLP model, yielding 0.965 area under the curve. A high area under the curve represents high precision and high recall, which indicates low false positive rates and low false negative rates.

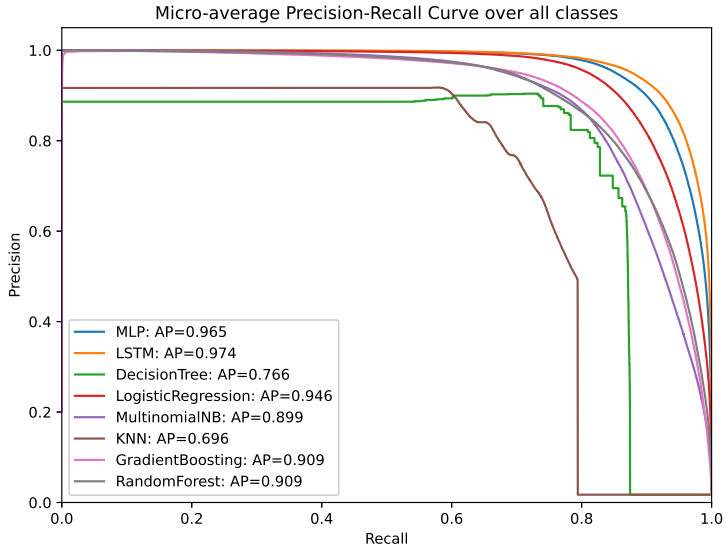
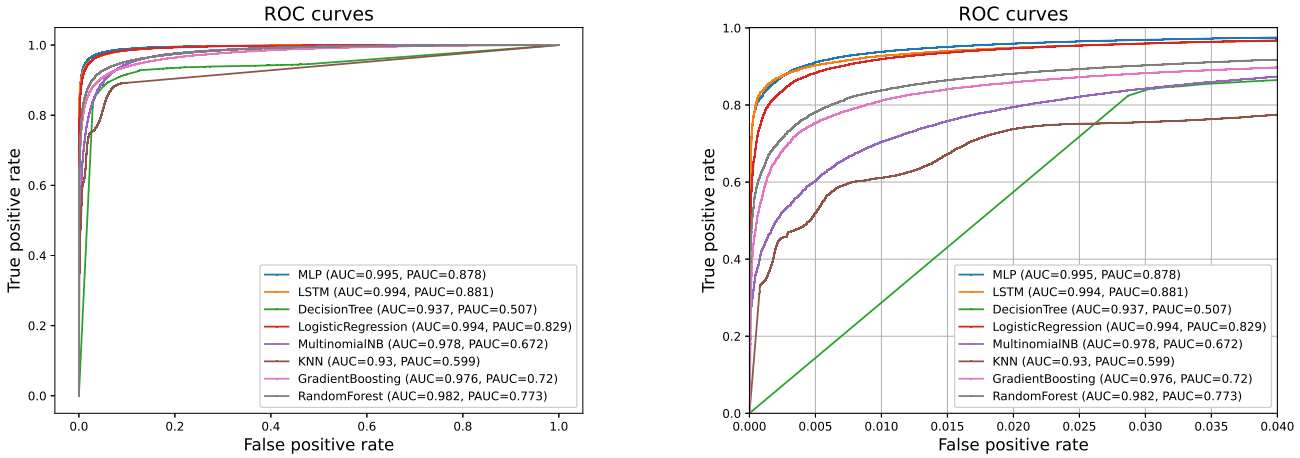


Figure 5. Precision-recall curves.

Figure 6 shows the ROC-curves. We aggregated the outputs of our multi-class classification models into binary classification by aggregating the 57 DGA outputs into class_1 and taking the topsites as class_0. The figure shows that the MLP model performs best, yielding 0.995 area under the ROC-curve (AUC), closely followed by the LSTM model and the LR model that both yield 0.994 AUC. The partial AUC (PAUC) for the false positive rate in the range from 0 to 0.04 is 0.881 for the LSTM model, 0.878 for the MLP model, and 0.829 for the LR model.



(**a**) ROC-curves (**b**) ROC-curves zoomed-in

Figure 6. ROC-curves.

Table 10 shows the optimum true positive and false positive rate for each model. The MLP model yields best results (96.54% TPR and 2.53% FPR) and performs slightly better than the LR model (96.11% TPR and 3.13% FPR) and the LSTM model (95.67% TPR and 2.68% FPR).

Table 10. Optimum true positive and false positive rates.

Model	True Positive Rate (%)	False Positive Rate (%)
DT	88.88	5.58
GB	90.81	4.96
KN	88.22	7.41
LR	96.11	3.13
LSTM	95.67	2.68
MLP	96.54	2.53
MNB	92.61	7.39
RF	92.20	4.39

4.6. Discussion

In our experiments, the DL models clearly yielded better results than the ML models in multi-class classification, as shown by the F1-scores. This is as expected, since DL models are more advanced and allow deeper analysis of the input data either by applying several hidden layers (in the MLP model) or feedback (in the LSTM model). This is in line with previous studies [40,46], which also showed that DL methods tend to outperform basic ML methods. Cucchiarelli et al. [34] on the other hand showed that ML methods with careful feature selection and classifier tuning can still outperform DL methods. Although our experiments show that ML methods perform worse than DL methods with multi-class classification, the AUC results with binary classification for the DL models and the best performing ML models (SVM and LR) are similar, which confirms the importance of feature selection (as demonstrated by our TF-IDF feature selection) and classifier tuning.

We used TF-IDF features with the MLP model, and with all ML models. We argued that TF-IDF features are promising due to the differences in the n-gram distributions of benign domain names and malicious domain names from different DGA families. Our experiments indicate that models using TF-IDF features indeed perform well. The results obtained with TF-IDF in the MLP model are comparable with the results obtained with the LSTM model in which we applied standard embedding.

Our experiments also show that there are notable differences among domain names from different DGA types. DGA-H domain names are distinguishable since they represent hexadecimal numbers, and hence they are easier to classify. This also shows in our results, as the highest aggregated F1-scores are for DGA-H (up to 99.96% with the LSTM model). DGA-W domain names are close to regular domain names, and hence they are more difficult to classify. This also shows in our results, as the F1-scores for DGA-W are much lower than for DGA-A and DGA-H. For DGA-W, both the LSTM and MLP model do not perform very well (73.18% and 78.57% micro average F1-score) and the SVM model shows better results (83.61%).

Unfortunately it is not straightforward to compare our results to prior results as published in scientific literature. This is mainly due to differences in the datasets used in experiments. As discussed in Section 3, nearly all studies used different datasets of benign and malicious domain names. Even if the same datasets are used as sources, the datasets originate from different time periods and also different subsets are taken with different numbers and types of DGA families. As discussed above, the mix of DGA families included in the dataset has a large impact.

Comparison with prior work therefore would require to implement the models from prior work and evaluate them with our datasets. There is a vast amount of prior work, and criteria would be needed for deciding which models from prior work to consider. Furthermore, often details are missing in literature that hinder reproducing models, such as the values of all hyperparameters and default settings, and the configurations of software tools. And even when reproducing the models is feasible, training and evaluating the models takes considerable effort.

5. Conclusions

We presented the results of an extensive literature review on the application of ML and DL for detection of DGA-generated domain names. We observed that this is an active research field to which numerous groups all over the world are contributing. We also observed that there is no common methodology for performing experiments and reporting on results. Different ML and DL models are being used. Arbitrary combinations of features for ML models are being used and a common ground for features is missing. Different datasets from various sources are being used, with different subsets of DGA families. These differences cause that it is hard to compare experimental results.

We proposed the usage of TF-IDF as single feature type. We apply TF-IDF for evaluating how relevant n-grams are to a domain name in a set of domain names. We used TF-IDF of the 5000 most popular n-grams (for $n = 1, 2, 3$) as features for popular ML and DL models. For comparison, we also used an LSTM model with embedding layer to convert domain names from a sequence of characters into a vector representation. Our results show that the DL models outperform ML models. The LSTM and MLP model provide the highest overall F1-scores (micro-averages of 90.69% and 89.08%), the highest area under the precision-recall curve (micro-averages of 0.974 and 0.965), the highest area under the ROC-curve (0.994 and 0.995), and the highest true positive rates (95.67% and 96.54%) with the lowest false positive rates (2.68% and 2.53%). Hence, the performance of the MLP model with TF-IDF features and the LSTM model with embedding is rather similar.

A limitation of any approach that relies on a single feature type, is that an adversary can tune the DGA such that the feature values of malicious domain names match those of benign domain names. This is also the case for our TF-IDF based approach: an adversary may tune a DGA to generate domain names for which the n-gram distributions match those of benign domain names, although this might be difficult to implement due to the large number of n-grams. We also observe that results differ for different DGA types. Our LSTM and MLP models perform well for classifying arithmetic-based and hash-based DGAs, but less for wordlist-based DGAs where domain names resemble regular domain names.

In our future work we intend to look at the effectiveness of features for ML models as mentioned in scientific literature. We observed that a large variety and rather arbitrary combinations of features have been applied, and it is not clear yet which features are effective in what cases. We also plan to look into hybrid learning models, where different types of models are combined. We observed that different models perform best for classifying domain names from different types of DGAs. We also intend to explore more advanced deep-learning models to derive features and to classify DGA-generated domain names.

Author Contributions: H.V.: literature review, paper writing, project lead, funding acquisition; H.A.: TF-IDF methodology, model creation, experiments, figures. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Open Universiteit and SIDN Labs in the Dagobert-project. The APC was funded by Open Universiteit.

Acknowledgments: We kindly thank IT and Facility Services at Open Universiteit and SURF for providing the compute servers for performing our experiments. We kindly thank Daniel Plohmann at Fraunhofer FKIE for providing access to DGArchive.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A. Precision and Recall

Table A1. Precision (expressed as percentage).

Class	DT	GB	KNN	LSTM	LR	MLP	MNB	RF	SVM
banjori	93.38	99.51	98.76	99.36	99.49	99.80	99.41	**100**	99.73
bedep	36.44	61.67	12.12	**85.80**	55.75	70.84	71.29	58.40	69.29
chinad	67.76	76.24	32.35	**99.63**	85.75	91.80	57.37	84.51	95.68
conficker	21.70	47.01	10.59	**55.58**	34.05	41.55	26.70	49.40	42.16
corebot	85.93	97.57	97.33	**99.74**	99.34	99.28	97.11	98.43	99.44
cryptolocker	37.83	51.12	29.27	**68.73**	40.90	57.19	28.86	57.25	48.16
dnschanger	43.47	42.87	8.09	**74.41**	33.34	58.66	22.64	42.63	44.10
ebury	58.40	74.13	94.08	**99.67**	98.31	93.08	98.38	91.64	95.84
emotet	95.67	89.61	96.48	**99.34**	89.19	95.96	79.82	90.85	93.45
fobber	22.03	46.24	0	**83.85**	9.23	48.53	0	63.95	25.00
gameover	90.45	94.96	71.43	**99.90**	89.73	98.58	56.44	95.67	98.71
locky	42.42	72.54	53.45	**83.45**	56.79	59.73	53.92	75.94	61.41
murofet	58.57	64.84	28.30	**76.99**	57.73	65.60	37.81	62.12	65.12
murofetweekly	95.84	99.37	**100**	99.87	99.97	99.87	98.66	99.90	**100**
necurs	66.96	93.38	29.41	**96.28**	86.08	92.89	76.43	94.05	94.74
nymaim	32.16	64.00	18.55	**67.31**	44.17	55.02	31.75	63.18	54.09
oderoor	27.77	52.83	77.50	75.09	83.82	67.19	**94.53**	56.61	92.08
padcrypt	89.49	95.98	97.77	**99.90**	99.48	99.67	87.60	99.39	99.44
proslikefan	36.19	63.77	14.35	64.94	51.83	**69.42**	47.89	69.43	62.81
pushdo	82.34	93.07	94.01	**98.02**	96.43	96.39	93.15	94.73	96.92
pushdotid	41.13	57.57	41.61	**95.34**	76.08	78.55	62.97	69.38	79.31
pykspa	24.96	32.50	15.04	**40.58**	29.95	36.41	22.98	34.42	36.56
pykspa2	46.92	53.46	45.05	54.01	47.93	51.53	49.18	**54.66**	49.01
pykspa2s	46.73	52.71	45.49	52.71	48.10	51.77	49.46	**54.53**	49.80
qadars	78.06	93.07	90.42	99.25	98.69	**99.43**	97.02	88.77	99.28
qakbot	48.94	65.05	6.25	72.19	39.55	63.58	33.98	**75.11**	50.22
ramdo	97.77	93.97	90.58	**100**	93.43	98.92	74.55	96.13	96.23
ramnit	26.27	31.05	17.19	46.95	25.19	38.85	23.24	**50.47**	26.54
ranbyus	67.11	72.00	70.00	**81.09**	64.83	75.82	45.66	72.91	71.59
rovnix	78.04	75.96	31.82	**99.76**	73.2	85.68	55.40	74.96	77.50
shifu	38.08	37.07	10.60	**69.10**	32.55	52.68	31.34	42.17	36.50
simda	76.00	70.64	61.85	**96.17**	94.28	94.40	93.79	66.47	95.44
sisron	99.93	**100**	97.33	**100**	99.96	99.93	99.93	**100**	99.96
sphinx	59.19	49.86	78.79	**85.16**	53.92	60.02	39.52	48.13	54.62
sutra	95.71	96.38	97.82	**99.70**	97.32	97.45	89.43	95.31	98.22
symmi	99.87	99.54	90.53	**100**	99.84	99.97	97.85	99.77	99.97
szribi	93.42	78.53	72.81	**96.41**	93.63	96.39	88.16	90.67	96.19
tempedrevetdd	5.67	4.40	3.57	**83.03**	14.94	64.29	0	0	51.22
tinba	53.86	57.89	67.27	**73.67**	60.29	75.91	45.23	57.27	69.03
tofsee	97.24	99.89	46.97	98.20	97.48	**98.93**	97.20	96.45	97.99
torpig	84.44	95.16	78.84	98.07	97.42	**98.20**	97.41	86.54	98.04
urlzone	66.56	78.40	25.49	**98.44**	81.76	90.83	64.65	73.40	86.00
vawtrak	25.70	71.96	51.83	75.25	65.94	75.31	78.97	**97.44**	76.26
vidro	48.86	61.01	**82.21**	50.19	49.98	43.10	72.03	71.71	59.39
virut	47.46	51.41	8.13	**77.79**	44.61	70.33	34.12	68.96	59.71
xxhex	99.23	99.62	93.18	99.47	**100**	99.92	97.40	**100**	**100**
bamital	99.39	99.62	71.36	**99.97**	**99.97**	99.83	93.44	**99.97**	**99.97**
dyre	98.59	99.73	63.73	**100**	99.97	99.97	97.98	**100**	**100**
ekforward	97.49	99.53	81.20	**99.89**	**99.89**	**99.89**	98.87	99.66	99.66
infy	98.05	97.60	58.21	**99.80**	99.15	99.53	97.63	98.66	99.42
pandabanker	98.33	99.07	61.18	**99.87**	98.66	99.57	96.91	99.28	99.24
tinynuke	99.37	99.57	65.78	**99.97**	99.87	**99.97**	89.84	99.90	99.93
wd	99.93	**100**	83.16	**100**	99.93	99.97	99.83	**100**	**100**
gozi	53.51	88.72	93.62	83.69	91.87	92.49	94.30	96.37	**97.46**
matsnu	25.18	66.67	70.29	59.75	75.75	81.17	92.75	**100**	91.96
nymaim2	52.05	87.13	82.37	84.51	89.61	88.51	88.86	**99.06**	95.60
suppobox	60.46	72.12	85.98	79.47	87.00	86.42	92.51	**96.67**	93.74
topsites	91.49	85.60	70.89	95.94	95.13	**96.20**	90.49	81.67	95.02

Table A1. *Cont.*

Class			DT	GB	KNN	LSTM	LR	MLP	MNB	RF	SVM
DGA	macro	average	65.16	74.90	57.92	**85.57**	74.28	80.82	69.20	79.00	79.47
		stddev	27.66	22.74	31.94	**16.81**	26.65	20.02	29.22	22.16	23.13
	micro	average	67.17	76.55	59.30	**85.16**	75.54	81.25	70.00	80.25	80.25
		stddev	25.91	20.81	30.99	**17.50**	24.88	20.15	27.26	19.89	22.38
DGA-A	macro	average	61.56	70.87	54.01	**84.14**	69.40	77.38	63.07	74.21	75.06
		stddev	26.87	22.91	34.00	**17.36**	27.25	20.61	29.34	22.11	23.67
	micro	average	64.12	72.70	55.48	**83.67**	70.61	77.67	63.55	75.30	75.64
		stddev	24.90	20.97	33.32	**18.21**	25.49	20.98	27.13	19.55	22.99
DGA-H	macro	average	98.74	99.30	69.23	**99.93**	99.63	99.82	96.36	99.64	99.75
		stddev	0.80	0.74	9.03	**0.07**	0.48	0.18	3.25	0.47	0.29
	micro	average	98.88	99.28	67.88	**99.93**	99.60	99.81	96.07	99.64	99.75
		stddev	0.72	0.78	8.53	**0.07**	0.50	0.19	3.32	0.49	0.30
DGA-W	macro	average	47.80	78.66	83.07	76.86	86.06	87.15	92.11	**98.03**	94.69
		stddev	13.44	9.48	8.42	10.06	6.20	4.08	2.00	**1.54**	2.05
	micro	average	47.64	78.57	82.95	76.74	85.98	87.10	92.10	**98.04**	94.67
		stddev	13.52	9.50	8.44	10.13	6.23	4.09	1.99	**1.55**	2.05
Total	macro	average	65.62	75.09	58.14	**85.75**	74.64	81.08	69.56	79.05	79.74
		stddev	27.64	22.59	31.71	**16.72**	26.56	19.95	29.10	21.97	23.02
	micro	average	80.35	81.45	65.59	**91.00**	86.15	89.35	81.11	81.02	88.26
		stddev	21.32	14.79	21.75	**13.00**	19.46	15.54	21.09	13.48	16.84

Table A2. Recall (expressed as percentage).

Class	DT	GB	KNN	LSTM	LR	MLP	MNB	RF	SVM
banjori	95.54	97.24	98.02	99.86	99.56	99.83	91.59	98.30	**99.93**
bedep	43.86	40.93	0.19	**64.37**	48.00	54.79	35.12	23.77	54.47
chinad	71.19	60.31	0.37	**98.77**	61.28	78.81	45.46	56.64	67.29
conficker	24.26	21.41	3.84	30.52	26.49	**35.77**	15.08	14.92	31.28
corebot	86.01	80.71	34.79	**99.84**	99.28	99.67	99.67	84.53	99.21
cryptolocker	43.77	41.31	0.82	**65.82**	38.58	56.37	36.98	28.71	49.47
dnschanger	50.44	45.56	0.62	**94.07**	36.15	84.14	19.24	33.46	64.93
ebury	51.74	53.06	89.26	**98.68**	86.61	93.39	90.58	41.65	91.40
emotet	97.87	99.20	34.77	**99.87**	99.60	99.80	95.17	99.67	99.83
fobber	**22.57**	13.18	0	22.24	1.98	5.44	0	9.06	2.64
gameover	90.87	93.55	0.17	**97.95**	86.40	96.66	77.41	95.83	93.62
locky	40.89	38.10	5.78	**53.39**	49.43	52.61	39.10	31.57	46.77
murofet	58.72	62.34	0.49	**84.26**	62.83	79.85	67.47	56.83	70.31
murofetweekly	95.50	98.91	77.41	99.93	99.70	99.87	**100**	99.47	**100**
necurs	63.00	57.84	0.33	**80.36**	70.34	73.41	65.31	42.86	69.74
nymaim	31.61	36.86	5.18	41.63	38.35	**43.60**	19.95	32.96	39.67
oderoor	24.58	24.13	82.19	35.71	**65.98**	42.12	63.13	14.87	61.09
padcrypt	91.72	89.29	50.34	**99.97**	99.93	99.77	**99.97**	96.58	**99.97**
proslikefan	32.39	30.94	3.12	**44.89**	35.74	37.40	18.83	21.92	36.57
pushdo	78.89	83.71	59.48	93.13	**95.88**	95.68	90.72	75.34	94.94
pushdotid	38.56	33.31	6.64	**99.39**	85.88	96.26	92.91	19.48	87.89
pykspa	23.35	27.79	4.48	**65.76**	30.53	53.57	19.77	28.19	44.05
pykspa2	48.65	39.36	46.32	47.32	47.75	**55.91**	43.72	45.69	50.35
pykspa2s	42.60	52.50	43.34	**59.83**	47.57	46.30	38.34	49.37	48.77
qadars	71.80	71.31	66.75	**99.44**	98.79	97.74	99.18	74.81	99.25
qakbot	41.70	40.89	0.17	**55.75**	40.39	49.16	33.70	33.77	41.63
ramdo	98.50	97.00	18.85	**100**	99.94	99.88	99.94	98.85	**100**
ramnit	24.92	12.80	0.75	**58.55**	30.79	23.83	34.32	5.47	29.90
ranbyus	64.38	69.76	1.38	**92.10**	65.60	73.96	51.95	65.50	71.83
rovnix	76.20	82.66	0.24	**99.08**	79.58	89.81	72.61	87.82	87.28
shifu	38.45	42.39	4.23	**94.51**	21.55	52.68	2.96	14.79	28.17
simda	75.83	76.58	84.33	**98.96**	97.10	97.82	93.55	79.15	98.14
sisron	99.93	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
sphinx	60.52	70.17	2.61	**98.42**	65.62	87.03	54.49	63.54	76.84
sutra	96.64	98.29	87.48	99.80	99.83	99.90	99.73	99.66	**100**
symmi	99.71	99.84	99.80	**100**	**100**	**100**	**100**	99.84	**100**
szribi	93.48	77.09	99.87	99.87	99.27	99.37	96.08	91.16	**99.90**
tempedrevetdd	3.49	1.00	0.50	**34.16**	3.24	8.98	0	0	5.24
tinba	53.16	36.11	8.76	**94.59**	62.94	73.03	65.31	23.32	67.14
tofsee	98.60	99.25	**100**	**100**	99.89	99.68	97.09	99.46	99.89

Table A2. *Cont.*

Class			DT	GB	KNN	LSTM	LR	MLP	MNB	RF	SVM
torpig			76.47	93.43	94.57	99.09	99.22	99.22	91.27	88.41	**99.66**
urlzone			59.26	64.22	1.28	**89.36**	81.65	83.95	63.23	63.43	84.08
vawtrak			11.83	9.90	12.72	**77.76**	50.51	47.04	21.72	4.88	44.60
vidro			40.34	39.66	36.24	51.69	43.59	**55.27**	35.95	37.02	44.14
virut			38.58	30.14	5.27	**77.97**	37.03	68.55	17.50	9.83	42.91
xxhex			98.86	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
bamital			99.56	99.69	52.19	**100**	**100**	**100**	**100**	**100**	**100**
dyre			99.12	99.76	16.12	**100**	**100**	**100**	91.53	**100**	**100**
ekforward			97.71	97.94	77.57	**100**	99.77	**100**	99.77	99.66	99.77
infy			98.75	96.31	93.06	**100**	98.68	99.73	97.87	99.59	99.32
pandabanker			99.30	99.24	94.89	**100**	**100**	**100**	99.83	99.97	**100**
tinynuke			99.17	99.77	41.77	**100**	99.90	99.97	99.67	**100**	**100**
wd			99.93	**100**	84.11	**100**	**100**	**100**	**100**	**100**	**100**
gozi			51.00	55.29	35.84	70.58	79.42	77.48	35.98	47.93	**79.55**
matsnu			23.05	4.71	20.47	47.01	47.21	48.56	2.07	0.03	**56.89**
nymaim2			44.69	34.52	11.68	**79.37**	74.37	73.54	32.54	6.91	72.29
suppobox			60.80	40.79	69.43	85.06	86.66	92.53	40.59	44.31	**95.49**
topsites			93.20	98.75	99.40	97.63	98.07	98.31	95.91	**99.58**	98.94
DGA	macro	average	63.94	62.49	38.09	**82.12**	71.52	76.84	63.09	58.61	74.18
		stddev	28.67	30.87	38.05	**22.94**	28.90	26.01	34.19	35.42	27.54
	micro	average	66.07	64.58	37.23	**83.14**	73.33	78.97	64.37	61.15	76.28
		stddev	26.71	29.12	37.19	**21.35**	26.60	23.23	32.36	33.74	24.82
DGA-A	macro	average	60.24	59.44	34.21	**80.41**	67.18	73.65	60.78	55.27	70.10
		stddev	27.73	29.46	38.79	**23.89**	29.59	26.91	33.53	33.83	28.39
	micro	average	62.99	62.28	33.20	**81.77**	69.26	76.25	62.77	58.75	72.55
		stddev	25.47	27.22	37.82	**22.13**	27.25	23.96	31.17	31.71	25.60
DGA-H	macro	average	99.08	98.96	65.67	**100**	99.76	99.96	98.38	99.89	99.87
		stddev	0.65	1.26	27.50	**0**	0.45	0.09	2.88	0.17	0.24
	micro	average	99.23	99.08	64.40	**100**	99.77	99.95	98.23	99.91	99.88
		stddev	0.49	1.27	28.71	**0**	0.47	0.10	3.00	0.16	0.25
DGA-W	macro	average	44.89	33.83	34.36	70.51	71.92	73.03	27.80	24.80	**76.06**
		stddev	**13.85**	18.42	22.02	14.51	14.92	15.80	15.12	21.50	13.89
	micro	average	44.72	33.58	34.21	70.36	71.73	72.85	27.61	24.54	**75.91**
		stddev	**13.92**	18.49	22.00	14.60	15.00	15.88	15.22	21.51	13.93
Total	macro	average	64.44	63.12	39.14	**82.38**	71.97	77.21	63.65	59.32	74.60
		stddev	28.68	30.96	38.55	**22.83**	28.86	25.94	34.16	35.52	27.49
	micro	average	80.78	83.10	70.93	**90.99**	86.74	89.45	81.47	81.98	88.57
		stddev	22.57	26.04	39.91	**16.15**	21.82	18.44	26.95	29.80	20.24

References

1. Khattak, S.; Ramay, N.R.; Khan, K.R.; Syed, A.A.; Khayam, S.A. A Taxonomy of Botnet Behavior, Detection, and Defense. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 898–924. [CrossRef]
2. Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* **1988**, *24*, 513–523. [CrossRef]
3. Barabosch, T.; Wichmann, A.; Leder, F.; Gerhards-Padilla, E. Automatic extraction of domain name generation algorithms from current malware. In Proceedings of the STO Information Systems and Technology Panel (IST) Symposium: Information Assurance and Cyber Defense, Koblenz, Germany, 24–25 September 2012 ; NATO: Brussels, Belgium, 2012; pp. 2.1–2.13.
4. Royal, P. *Analysis of the Kraken Botnet*; Technical Report; Damballa, Inc.: Atlanta, GA, USA, 2008.
5. Stone-Gross, B.; Cova, M.; Cavallaro, L.; Gilbert, B.; Szydlowski, M.; Kemmerer, R.; Kruegel, C.; Vigna, G. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In Proceedings of the 16th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 9–13 November 2009; ACM: New York, NY, USA , 2009; pp. 635–647. [CrossRef]
6. Leder, F.; Werner, T. Know Your Enemy: Containing Conficker. To Tame A Malware; 7 April 2009 (rev2). Available online: https://www.honeynet.org/papers/kye-kyt/know-your-enemy-containing-conficker (accessed on 1 July 2021).
7. Plohmann, D.; Yakdan, K.; Klatt, M.; Bader, J.; Gerhards-Padilla, E. A Comprehensive Measurement Study of Domain Generating Malware. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; USENIX Association: Austin, TX, USA, 2016; pp. 263–278.
8. Mockapetris, P. *RFC 1035 Domain Names Implementation and Specification*; IETF Request for Comments: 1035; IETF: Fremont, CA, USA, 1987.
9. Klensin, J. *RFC 5890 Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework*; IETF Request for Comments: 5890; IETF: Fremont, CA, USA, 2010.

10. Beel, J.; Gipp, B.; Langer, S.; Breitinger, C. Research-paper recommender systems: A literature survey. *Int. J. Digit. Libr.* **2016**, *17*, 305–338. [CrossRef]
11. Luhn, H. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM J. Res. Dev.* **1957**, *1*, 309–317. [CrossRef]
12. Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *J. Doc.* **1972**, *28*, 11–21. [CrossRef]
13. Zago, M.; Pérez, M.G.; Pérez, G.M. Scalable detection of botnets based on DGA. *Soft Comput.* **2020**, *24*, 5517–5537. [CrossRef]
14. Chiba, D.; Yagi, T.; Akiyama, M.; Shibahara, T.; Mori, T.; Goto, S. DomainProfiler: toward accurate and early discovery of domain names abused in future. *Int. J. Inf. Secur.* **2018**, *17*, 661–680. [CrossRef]
15. Schüppen, S.; Teubert, D.; Herrmann, P.; Meyer, U. FANCI: Feature-based Automated NXDomain Classification and Intelligence. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018 ; USENIX Association: Austin, TX, USA, 2018; pp. 1165–1181.
16. Ashiq, M.I.; Bhowmick, P.; Hossain, M.S.; Narman, H.S. Domain Flux-based DGA Botnet Detection Using Feedforward Neural Network. In Proceedings of the MILCOM 2019—2019 IEEE Military Communications Conference (MILCOM), Norfolk, VA, USA, 12–14 November 2019 ; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6. [CrossRef]
17. Fu, Y.; Yu, L.; Hambolu, O.; Ozcelik, I.; Husain, B.; Sun, J.; Sapra, K.; Du, D.; Beasley, C.T.; Brooks, R.R. Stealthy Domain Generation Algorithms. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1430–1443. [CrossRef]
18. He, W.; Gou, G.; Kang, C.; Liu, C.; Li, Z.; Xiong, G. Malicious Domain Detection via Domain Relationship and Graph Models. In Proceedings of the 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC), London, UK, 29–31 October 2019 ; IEEE: Piscataway, NJ, USA, 2019; pp. 1–8. [CrossRef]
19. Li, W.; Jin, J.; Lee, J. Analysis of Botnet Domain Names for IoT Cybersecurity. *IEEE Access* **2019**, *7*, 94658–94665. [CrossRef]
20. Liu, Z.; Yun, X.; Zhang, Y.; Wang, Y. CCGA: Clustering and Capturing Group Activities for DGA-Based Botnets Detection. In Proceedings of the 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, 5–8 August 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 136–143. [CrossRef]
21. Selvi, J.; Rodríguez, R.J.; Soria-Olivas, E. Detection of algorithmically generated malicious domain names using masked N-grams. *Expert Syst. Appl.* **2019**, *124*, 156–163. [CrossRef]
22. Yang, L.; Zhai, J.; Liu, W.; Ji, X.; Bai, H.; Liu, G.; Dai, Y. Detecting Word-Based Algorithmically Generated Domains Using Semantic Analysis. *Symmetry* **2019**, *11*, 176. [CrossRef]
23. Akhila, G.P.; Gayathri, R.; Keerthana, S.; Gladston, A. A machine learning framework for domain generating algorithm based malware detection. *Secur. Priv.* **2020**, *3*, e127. [CrossRef]
24. Alaeiyan, M.; Parsa, S.; Vinod, P.; Conti, M. Detection of algorithmically-generated domains: An adversarial machine learning approach. *Comput. Commun.* **2020**, *160*, 661–673. [CrossRef]
25. Almashhadani, A.O.; Kaiiali, M.; Carlin, D.; Sezer, S. MaldomDetector: A system for detecting algorithmically generated domain names with machine learning. *Comput. Secur.* **2020**, *93*, 101787. [CrossRef]
26. Anand, P.M.; Kumar, T.G.; Charan, P.S. An Ensemble Approach For Algorithmically Generated Domain Name Detection Using Statistical And Lexical Analysis. *Procedia Comput. Sci.* **2020**, *171*, 1129–1136. [CrossRef]
27. Hwang, C.; Kim, H.; Lee, H.; Lee, T. Effective DGA-Domain Detection and Classification with TextCNN and Additional Features. *Electronics* **2020**, *9*, 1070. [CrossRef]
28. Liang, Z.; Zang, T.; Zeng, Y. MalPortrait: Sketch Malicious Domain Portraits Based on Passive DNS Data. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea, 25–28 May 2020 ; IEEE: Piscataway, NJ, USA, 2020; pp. 1–8. [CrossRef]
29. Mao, J.; Zhang, J.; Tang, Z.; Gu, Z. DNS anti-attack machine learning model for DGA domain name detection. *Phys. Commun.* **2020**, *40*, 101069. [CrossRef]
30. Palaniappan, G.; Sangeetha, S.; Rajendran, B.; Goyal, S.; Bindhumadhava, B.S. Malicious Domain Detection Using Machine Learning On Domain Name Features, Host-Based Features and Web-Based Features. *Procedia Comput. Sci.* **2020**, *171*, 654–661. [CrossRef]
31. Sivaguru, R.; Peck, J.; Olumofin, F.; Nascimento, A.; De Cock, M. Inline Detection of DGA Domains Using Side Information. *IEEE Access* **2020**, *8*, 141910–141922. [CrossRef]
32. Wu, J. Artificial Neural Network Based DGA Botnet Detection. *J. Phys. Conf. Ser.* **2020**, *1578*, 012074. [CrossRef]
33. Zhang, Y. A Ensemble Learning method for Domain Generation Algorithm Detection. *Acad. J. Comput. Inf. Sci.* **2020**, *3*, 31–40. [CrossRef]
34. Cucchiarelli, A.; Morbidoni, C.; Spalazzi, L.; Baldi, M. Algorithmically generated malicious domain names detection based on n-grams features. *Expert Syst. Appl.* **2021**, *170*, 114551. [CrossRef]
35. Patsakis, C.; Casino, F. Exploiting statistical and structural features for the detection of Domain Generation Algorithms. *J. Inf. Secur. Appl.* **2021**, *58*, 102725. [CrossRef]
36. Woodbridge, J.; Anderson, H.S.; Ahuja, A.; Grant, D. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. *arXiv* **2016**, arXiv:1611.00791.
37. Lison, P.; Mavroeidis, V. Automatic Detection of Malware-Generated Domains with Recurrent Neural Models. *arXiv* **2017**, arXiv:1709.07102.

38. Koh, J.J.; Rhodes, B. Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings. In Proceedings of the 2018 IEEE International Conference on Big Data, Seattle, WA, USA, 10–13 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 2966–2971. [CrossRef]
39. Tran, D.; Mac, H.; Tong, V.; Tran, H.A.; Nguyen, L.G. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing* **2018**, *275*, 2401–2413. [CrossRef]
40. Vinayakumara, R.; Somana, K.; Poornachandranb, P.; Kumara, S.S. Evaluating Deep Learning Approaches to Characterize and Classify the DGAs at Scale. *J. Intell. Fuzzy Syst.* **2018**, *34*, 1265–1276. [CrossRef]
41. Xu, C.; Shen, J.; Du, X. Detection method of domain names generated by DGAs based on semantic representation and deep neural network. *Comput. Secur.* **2019**, *85*, 77–88. [CrossRef]
42. Yu, B.; Pan, J.; Hu, J.; Nascimento, A.; De Cock, M. Character Level based Detection of DGA Domain Names. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–8. [CrossRef]
43. Akarsh, S.; Sriram, S.; Poornachandran, P.; Menon, V.K.; Soman, K.P. Deep Learning Framework for Domain Generation Algorithms Prediction Using Long Short-term Memory. In Proceedings of the 5th International Conference on Advanced Computing Communication Systems (ICACCS), Coimbatore, India, 15–16 March 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 666–671. [CrossRef]
44. Qiao, Y.; Zhang, B.; Zhang, W.; Sangaiah, A.K.; Wu, H. DGA Domain Name Classification Method Based on Long Short-Term Memory with Attention Mechanism. *Appl. Sci.* **2019**, *9*, 4205. [CrossRef]
45. Liu, Z.; Zhang, Y.; Chen, Y.; Fan, X.; Dong, C. Detection of Algorithmically Generated Domain Names Using the Recurrent Convolutional Neural Network with Spatial Pyramid Pooling. *Entropy* **2020**, *22*, 1058. [CrossRef]
46. Ren, F.; Jiang, Z.; Wang, X.; Liu, J. A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network. *Cybersecurity* **2020**, *3*, 4. [CrossRef]
47. Vij, P.; Nikam, S.; Bhatia, A. Detection of Algorithmically Generated Domain Names using LSTM. In Proceedings of the International Conference on COMmunication Systems and NETworkS (COMSNETS), Bengaluru, India, 7–11 January 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6. [CrossRef]
48. Highnam, K.; Puzio, D.; Luo, S.; Jennings, N.R. Real-Time Detection of Dictionary DGA Network Traffic Using Deep Learning. *SN Comput. Sci.* **2021**, *2*, 110. [CrossRef]
49. Namgung, J.; Son, S.; Moon, Y.S. Efficient Deep Learning Models for DGA Domain Detection. *Secur. Commun. Netw.* **2021**, *2021*, 8887881. [CrossRef]
50. Yilmaz, I.; Siraj, A.; Ulybyshev, D. Improving DGA-Based Malicious Domain Classifiers for Malware Defense with Adversarial Machine Learning. In Proceedings of the IEEE 4th Conference on Information and Communication Technology (CICT), Chennai, India, 3–5 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6. [CrossRef]
51. Wang, T.S.; Lin, H.T.; Cheng, W.T.; Chen, C.Y. DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis. *Comput. Secur.* **2017**, *64*, 1–15. [CrossRef]
52. Satoh, A.; Fukuda, Y.; Hayashi, T.; Kitagata, G. A Superficial Analysis Approach for Identifying Malicious Domain Names Generated by DGA Malware. *IEEE Open J. Commun. Soc.* **2020**, *1*, 1837–1849. [CrossRef]
53. Sun, X.; Yang, J.; Wang, Z.; Liu, H. HGDom: Heterogeneous Graph Convolutional Networks for Malicious Domain Detection. In Proceedings of the NOMS 2020—2020 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 20–24 April 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–9. [CrossRef]
54. Yan, F.; Liu, J.; Gu, L.; Chen, Z. A Semi-Supervised Learning Scheme to Detect Unknown DGA Domain Names Based on Graph Analysis. In Proceedings of the IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December 2020–1 January 2021; IEEE: Piscataway, NJ, USA, 2020; pp. 1578–1583. [CrossRef]
55. Yin, L.; Luo, X.; Zhu, C.; Wang, L.; Xu, Z.; Lu, H. ConnSpoiler: Disrupting C&C Communication of IoT-Based Botnet Through Fast Detection of Anomalous Domain Queries. *IEEE Trans. Ind. Inform.* **2020**, *16*, 1373–1384. [CrossRef]
56. Mac, H.; Tran, D.; Tong, V.; Nguyen, L.; Tran, H. DGA botnet detection using supervised learning methods. In Proceedings of the Eighth International Symposium on Information and Communication Technology (SoICT), Nha Trang, Vietnam, 7–8 December 2017; ACM: New York, NY, USA, 2017; pp. 211–218.
57. Pochat, V.L.; Goethem, T.V.; Tajalizadehkhoob, S.; Maciej Korczyński, W.J. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. *arXiv* **2018**, arXiv:1806.01156v3.