Detecting DGA-managed Thingbots Using DNS Traffic

Viet Hung Nguyen*, Nathan Shone**

**Le Quy Don Technical University, Hanoi, Vietnam (e-mail:hungnv@lqdtu.edu.vn).*
***Liverpool John-Moore University, L35AH, UK (e-mail:N.Shone@ljmu.ac.uk)*

Abstract: Many Internet of Things devices have weak security by default, which is often exploited by malware to recruit such devices into Thingbots (a botnet comprised of Internet of Things devices). The concerning capabilities of Thingbots have been demonstrated by Mirai, which powered the largest Distributed Denial of Service (DDoS) attack ever recorded. Thingbots rely upon the number of devices, rather than their computational capacity. Hence, as more devices become Internet-enabled, the severity of this problem will only increase. However, the coordination of such a large and distributed system poses a challenge. One commonly utilised mechanism is Domain Generation Algorithms (DGAs), which can help attackers communicate with compromised devices whilst evading detection. Current detection systems are unsuitable for analysing high volumes of network traffic as they struggle to balance accuracy with the expected levels of privacy-preservation and performance. Many effective yet complex techniques such as reverse engineering, are considered too costly in terms of time, resources and privacy. Other existing techniques tend to focus on post-event analysis such as aggregation of non-existent domains or analysis of basic characteristics but these often yield unacceptable delays or high false detection rates. As a solution to this problem, we present our novel technique for detecting DGA-managed Thingbots. We propose a solution that hybridises deep learning, shallow learning and morphological observations, to monitor Domain Name System (DNS) traffic. Using real-world data, we demonstrate that the proposed approach can accurately identify DNS traffic symptomatic of DGA-based Thingbots, with low false detection rates.

Keywords: DGA, DNS, deep learning, morphologic, Thingbots, IoT.

1. INTRODUCTION

Too many Internet of Things (IoT) devices lack a basic level of security, e.g., static default passwords, no brute force protection, no transport layer security (TLS) support. When combined with complacent end-users, they have become easy targets for malware infection and ultimately their recruitment into Thingbots. Hence, both Thingbots and botnets remain a persistent global security problem. Their continued prominence is mainly due to their appeal to criminals, as they are cheap to deploy, easy to use and manage, and difficult to uncover. Hence, botnets are involved in an ever-expanding array of cybercrime, including phishing (Naaz., 2021), spear phishing (Eftimie et al., 2022), malware propagation (Zhu et al., 2023), ransomware management (Beaman et al., 2021), identity theft, fraud (Raman et al., 2023) and Distributed Denial of Service (DDoS) attacks (Kumari and Jain, 2023). A side-effect of this is the increasing social and economic impact of botnets; for example, related ad fraud was estimated to cost $84bn in 2023 and will reach $170bn over the next five years (Bureau, 2023). Increasingly sophisticated methods utilised by Thingbot herders (the human controllers) ensure that a diverse range of OSs and platforms can now be enlisted. So far, the IoT concept has encouraged the instatement of Internet connectivity to a multitude of appliances and devices. Additionally, there is malware in circulation that specifically targets ARM platforms (Aurangzeb, 2023), the architecture commonly used by embedded and IoT devices in smart-cities. The vast number of Internet-enabled devices resulting from the IoT concept has provided significant incentives and motivation for botnet herders to target them. Furthermore, the recent drastic increase in the number of top level domains (TLDs) (from around 1500 in 2022 to over 2000 in 2024 (Marketing, 2024)), provides a significantly larger pool of potential domain name combinations, providing increased scope for Domain Generation Algorithm (DGA)-based botnet operation. Unfortunately, this problem is only going to increase in size, complexity and significance.

DGA-based Thingbots make use of domain fluxing – a notable technique proving popular and effective in operation control (Ghafir and Prenosil ,2014). This technique involves regularly changing the domain names that actively point to the herders' Command and Control (C&C) server. This constant changing (or fluxing) removes predictability, making it much more difficult to implement adequate defences. DGAs are used to choreograph connectivity by generating a pool of candidate domain names (using various different approaches) for a specific period. Bots can then attempt to contact the C&C server through these domains, before the DGA recalculates a fresh pool of domains. The effectiveness of this approach stems from its dynamic, flexible and rapidly deployable infrastructure. This presents numerous difficulties for administrators, law enforcement and security researchers attempting to shut them down. The success of DGAs has resulted in them being utilized in various botnets and adopted by other forms of malware. Unfortunately, DGA-based Thingbots will remain prominent due to the lack of accurate and suitable detection techniques. Our literature review has identified four main limitations with existing approaches, which are as follows:

• Performance: Many effective techniques such as deep packet inspection and reverse engineering (Deri and Fusco, 2021)

incur significant time and resource overheads. These prove particularly problematic when considering the scalability required to monitor large-scale networks.

• Privacy: Payload inspection provides a highly effective form of botnet detection (providing packets are unencrypted) (Ozdel et al., 2022). However, this technique is fraught with concerns over privacy violation. This is why multiple existing works are overlooked as being too intrusive, and explains the trending usage of basic high-level observations.

• Scope: The heavy reliance on the detection of reoccurring patterns (e.g. frequency of Non-Existent Domain (NXDomain) replies) is a significant limitation. Herders actively improve their techniques to evade detection e.g. the progression from a random character DGA to a dictionary DGA. It is therefore imperative that the scope used in the detection method is able to account for such progression. Otherwise, the approach can become quickly outdated or ineffective.

• Timeliness: The use of pattern-based techniques can often result in a detection delay. For example, multiple NXDomains need to be observed before being identified as anomalous. In modern interconnected networks, real time threat identification and remediation is a necessity to prevent the spread of threats.

The looming threat of Thingbots, combined with the existing shortcomings previously outlined, has provided the motivation for the development of the novel solution in this paper. We believe that combining deep learning and linguistic observations of domains can provide such a solution. In linguistics, morphology is considered to be the study of the form of words and the relationship between other words of the same language (Anderson and R., 2011).

Morphological analysis can provide measures for us to ascertain the genuineness of a domain. In this paper, we propose a novel solution to detect DGA domains by observing Domain Name System (DNS) traffic. Our approach is a hybridized detection algorithm that combines the power of deep learning (long short-term memory), shallow learning (random forest) and morphological observations. This solution aspires to overcome the four limitations outlined previously, whilst providing rapid and accurate detection of DGA-based domains.

It is important to note that our solution has been developed and implemented for domains written in English using ASCII characters. We firmly believe that the approach would be applicable to other languages, particularly those that are Latin alphabet-based. However, this application and evaluation will form part of our future work. To the best of our knowledge, the main novel contributions of this paper are:

• Hybrid DGA detection algorithm (HDDA): We have devised a novel approach for detecting DGA-generated domains. This approach combines the analytical power of deep learning, shallow learning and statistical observations to provide a comprehensive detection strategy using the linguistic structure of domain names.

• Holistic perspective: The unique combination of features utilized creates a holistic view of the domain name, rather than relying on specific and narrowly-focused characteristics. This means that the decision process is more resilient to characteristic changes. The increased scope offered through this perspective means that classifiers have a greater diversity of information upon which to base a more-accurate decision.

• Privacy-preserving solution: Our proposed approach does not require complex or time-consuming analysis, so it is able to scale more efficiently with reduced overheads. Furthermore, our focus on linguistic characteristics offers a solution that is able to circumvent privacy concerns.

The remainder of this paper is structured as follows. Section 2 provides background information on the core topics involved. Section 3 analyses the most relevant work in the field. Section 4 presents an overview of our proposed system and details our approach. The evaluation is provided in Section 5, Discussion in Section 6, whilst our conclusions and future work are outlined in Section 7.

2. BACKGROUND

In this section, we provide a brief introduction to both the DNS and DGA concepts, which are required to understand the topics addressed in this paper.

2.1 DNS

DNS is an integral part of most modern networks. It provides a decentralised and fault-tolerant worldwide directory, enabling the identification of services and devices across networks. Its primary function is to serve as a client-server model to translate memorable, user-friendly domain names into numerical IP addresses.

Client devices operate a DNS cache, whereby DNS records are stored for fast retrieval if the same site is accessed in the future. However, these DNS records are issued with a Time To Live (TTL) value, which dictates how long the cached records can remain valid. Different sites use different TTL values, based on their setup and configuration. However, a typical TTL value is considered to be around 1 hour or more. If the client does not possess a valid version of the record, then the query is passed to their recursive DNS server. This will either issue the information based on its own valid cached records, or it will query the authoritative DNS hierarchy on the client's behalf. A simplified version of the DNS hierarchy is illustrated in Figure 1.

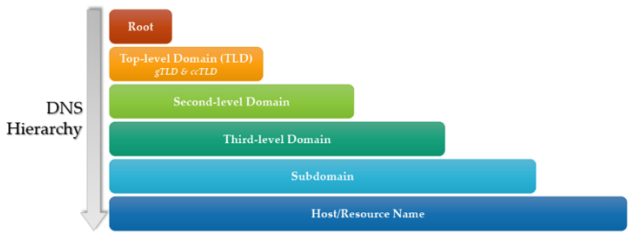

Fig. 1. Typical DNS Hierarchy.

Each level of the DNS hierarchy, shown in Figure 1, has an authoritative server that can answer DNS queries. The lower the level of the DNS hierarchy, the more potential records there are. To explain the hierarchy illustrated, we will use the example of:

'webmail.a-university.ac.uk'

It is important to note that domain names are read in reverse order. This is primarily for performance, as the higher elements of the DNS hierarchy have fewer options to traverse before finding the correct authoritative nameserver. There are strict rules governing the formulation of domain names, and we will only discuss those related to our work here.

Table 1. Explanation of DNS levels.

Label	Description
Root	At the top of the hierarchical tree are the root nameservers. Every domain has an implied "." at the end of the address, which denotes the root.
Top-level Domain (TLD)	The TLD can be broken down into two main categories, generic TLD (gTLD) and country code TLD (ccTLD). There are currently around 1205 TLDs. In example (1), "uk" is the ccTLD
Second-level Domain (SLD)	This is a dual purpose label. For domains with traditional TLDs (e.g. ".com"), this label is used to refer to the organisation that registered the domain. However, when ccTLDs are used, this label functions as an extension of the TLD and refers to categories under the TLD. So, in example (1), this refers to the academic network "ac".
Third-level Domain	This is usually unused by domains using traditional TLDs. However, when the SLD acts as an extension to the TLD (e.g. when a gTLD or ccTLD is used) it refers to the organisation that registered the domain. In example (1), this refers to the "a-university" organisation.
Subdomain	This refers to a subsection of the organisation's network. In example (1), this refers to "webmail".

Domain names are split into distinct parts (concatenated using a dot '.'), which are called labels. Each label cannot be longer than 63 characters in textual length and the full domain name cannot be more than 253 characters. A domain may be broken into as many as 127 different levels. In addition, domain names can only be formed using letters a-z or A-Z, digits 0-9 and a hyphen, also known as the LDH (Letters, Digits, Hyphens) rule. However, labels may not start or end with hyphens. This structure is detailed and explained in Table 1 using example defined previously.

2.2 DGAs

An efficient implementation of domain fluxing relies on a large pool of domain names through which bots and C&C servers can attempt connection. This technique reduces the probability of detection, and enables the botnet to continue even if one or more of the domains are blacklisted by authorities. Traditionally, a finite number of domains were defined in a static list embedded within the malware binary. However, by reverse engineering and analysing the malware binary, it became straightforward to extract these string lists, blacklist their contents, and for authorities to take down the C&C servers. To overcome this weakness, DGAs were created and are utilised by a growing number of types of malware, particularly those related to botnets. One of the first DGA-based malware families is Conficker, and since then the technique has evolved and diversified to other malware (Shin et al., 2012).

DGAs are algorithms that use custom methodologies to generate a large number of random candidate domains on the-fly. However, for the sake of synchronicity and limiting visibility, DGA methods usually operate around dates, times or seed values. Botnet herders register several of these candidate domains per connection schedule. When a connection is required, a defined number of these candidate domains are tested (to avoid raising suspicion) to verify whether the C&C server can be contacted. For example, the Confiker.D worm tests 500 domains from a candidate list of 50,000. These candidate domains are generated on a regular basis, with most algorithms adopting a daily schedule, although some use more frequent schedules (e.g. every hour). The benefit of this approach for herders is they only have to register several of the thousands of random candidate domains generated per day to succeed, whereas authorities would have to register or blacklist thousands.

DGAs build upon the flaws of their static-list-based predecessors and are designed to evade blacklisting and domain reputation techniques. They provide a lightweight and easily configurable infrastructure that permits the dissemination of malware without compromising C&C infrastructures or identities. The technique also ensures minimum operational visibility, as the window of C&C accessibility is short (e.g. 1 day), providing enough time for the bots to connect, before the C&C is relocated.

There are four main generation strategies used by DGA algorithms, which are outlined below (Ayub et al., 2021).

• Arithmetic: Mathematical calculations are used to compute or randomly select ASCII codes or characters from an array.

• Wordlist: Random words are selected from either a static wordlist (embedded in the malware binary) or a public static text (e.g. laws).

• Hashing: The digest output of a hashing algorithm (e.g. MD5 or SHA) is used.

• Permutation: Characters or words in a domain are reordered. Example outputs from these techniques are shown in Table 2.

The dynamic and obscured nature of DGA-based methods makes them extremely difficult and time-consuming to mitigate. It requires the malware and DGA algorithm to be reverse engineered to understand the calculation method and implement measures to prevent further communications. One disadvantage of DGA-based botnets is the large number of observable NXDomain DNS responses that can result from failed C&C connections. This is currently a characteristic that the majority of existing detection techniques focus upon. However, as techniques continue to evolve, this will be a key area that herders will seek to improve upon.

Table 2. Example DGA-based Domains.

Generation Strategy	Examples
Arithmetic	yoxxuihfbffhp5.com
Wordlist	toformidablerepeated.com
Hashing	1887ed498f5239844d2550ba83f1dadc.xyz
Permutation	circumstanceandreduces.cc

3. RELATED WORKS

Botnet and DGA-domain detection is an active research area. Some of the most pertinent existing research in the area is discussed in this section.

A common approach for botnet detection is the analysis of NXDomain DNS answers, and there are several notable examples of this approach. For example, GOTS (Nguyen et al., 2015) is a Big Data orientated large-scale DGA-based botnet detection system. It detects DGA-based botnets by analysing DNS traffic logs and NXDomain records. Based on the similarity of domain name characteristic distribution, they combine clustering and classification algorithms to remove noise and group similar domains into clusters. Another example is in (Manasrah et al., 2022), which is a DGA-based botnet identification system that analyses streams of NXDomain DNS responses. It adopts a similar approach of using classifiers to label various statistical features extracted from NXDomains and uses a hidden Markov model to identify domains created using NewDGA-vX based on the sequence of characters. Similarly, Tu et al. (Tu et al., 2015) also adopt a NXDomain-based analysis, but focus on both time series analysis and domain pattern similarity. Other similar approaches also include work by Sharifnya (Sharifnya et.al., 2013) and Guerid (Guerid et al., 2013).

An alternative approach commonly featured in existing literature is the statistical evaluation of key metrics and observations. In (Bilge et al., 2014), the authors present EXPOSURE, which is a passive DNS analysis tool for detecting botnets. The authors claim that it does not focus on specific classes of malware, instead it provides a generic and widely applicable service. It uses classifiers on various observations to determine whether a machine is a bot. Similarly, Grill et al. (Grill et al., 2015) propose a method based on the statistical analysis of DNS traffic patterns, specifically the relationship between the DNS server and hosts (e.g. the number of requests and number of newly visited domains).

Reputation is an emerging approach within botnet and DGA-domain detection. Current notable examples include the work by Zou et al. (Zou et al., 2015), which proposes a method of establishing reputation based on belief propagation using DNS relationships modelled using graphs. Also, Sharifnya et al. (Sharifnya et.al., 2013) and Abadi (Abadi, 2013) proposed a reputation scoring technique based upon suspicious failure matrices (which incidentally uses NXDomain data). The existing work most similar to that proposed in this paper, is by Yadav et al. (S. Yadav, 2010). They propose statistical measures of the linguistic characteristics of domain names. They utilise techniques such as Kullback-Leibler divergence, Jaccard index, and Levenshtein edit distance for classifying domains to identify whether they are DGA-based or not. However, we present an alternative method that focuses upon different features, reduces complexity and is therefore more resource and time efficient, as will be detailed in the next section.

Recently, many researchers have focused on using deep learning in DGA detection. F. Ren et.al. proposed a deep learning framework called ATT-CNN-BiLSTM designed to detect DGA domains (F. Ren, 2020). The authors used Convolutional Neural Network (CNN) and a bidirectional Long Short-Term Memory (BiLSTM) neural network layer to extract features from domain sequence information, then fed them to the classifier to detect DGA. Shahzad et al. proposed a method that used recurrent neural network (RNN) architecture to detect the DGA domain (Shahzad et al., 2021). The authors assessed various RNN-based architectures by testing them with a dataset comprising over 2 million domain names. X. Sun et al. proposed adding Whois features and the N-gram features and then using a deep learning model based on BiLSTM, Attention, and CNN constructed for DGA detection (Sun X, 2023).

4. PROPOSED SOLUTION

Humans have greater difficulty in recalling numerical sequences (in this case IP addresses) than letter sequences. This is why DNS was created, to provide a human-friendly method of locating and accessing services and resources. Hence, domain names are usually carefully chosen to concisely reflect organisation names or services.

However, domains created by DGAs are designed to be suitably obscure, so as to avoid clashes with genuine domains. Typically, this involves the random generation of alphanumeric strings or the merging of several dictionary words. Some DGAs generate in the region of 50,000 domains per run, so it is important that the algorithm can create suitably unique domains to avoid pattern detection mechanisms. An increasing number of advanced DGAs are becoming more linguistically compliant (Zhao et al., 2023) (e.g. using common prefixes and suffixes). However, complete compliance is difficult to accomplish on the scale required for DGA operation, without revealing trends or patterns. We firmly believe that the morphological characteristics of domain names can be used as the basis of a novel approach to accurately distinguish between DGA and human-created domains. Furthermore, morphological characteristics are not subject to privacy concerns.

For our proposed approach, we are interested in the TLD and second/third level domain because this is main part of the domain name freely created by a human (i.e. not defined by convention). We say the second/third level domain, as many countries use a ccTLD, which subsequently means that the second level domain is actually used as additional form of TLD. For instance, in the UK, many sites use 'co.uk', meaning that the unique and most significant part of the domain is usually in the third level label. To avoid confusion, we will refer to this unique part of the domain as the Significant Label (SL), so "example" would be the SL of www.example.co.uk.

1. Sequential character-level feature extraction: LSTM RNNs have the ability to learn sequential patterns (S. Hochreiter, 1997). So, we use LSTM RNNs to extract character-level features from each domain name sequence.

2. Feature combining: We then combine the LSTM-extracted features with the morphological observation features described in this section. This combining creates a new feature that offers a more holistic perspective to domain name analysis. Using this approach, it is possible for some feature overlap to occur (i.e. some mandatory morphological features may also be extracted by the LSTM layers). In such an eventuality, this would have the effect of endorsing the correct extractions by the LSTM layers when classification occurs. Likewise, if no overlap occurs, this ensures that the critical features are still considered during the classification.

3. Classification: The new combined feature will then undergo training and binary classification (i.e. benign or malicious).

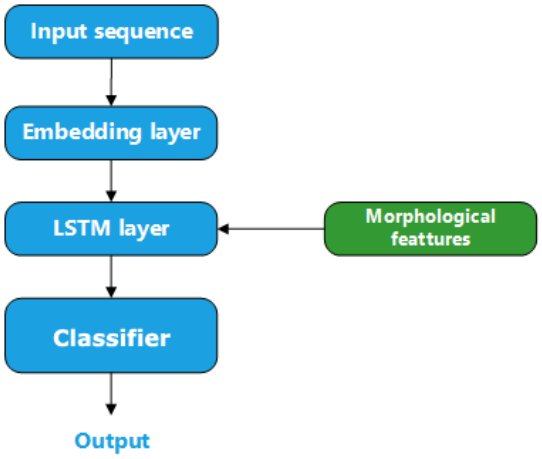

Fig. 2. Hybrid DGA detection model.

An illustrative overview of our proposed model is given in Figure 2.

To the best of our knowledge, the proposed method is the first time that a mixed feature between sequential characteristics and linguistic characteristics has been used to detect DGA-created domains.

4.1 LSTM

A Recurrent Neural Network (RNN) is a dynamic model that has been used for many years to learn the time dependencies across input data. The output depends not only on current input, but also on the previous states of networks. RNN scan be trained using a backpropagation algorithm but error signals tend to diminish over time, and traditional RNN can't hold the dependency through many iterations. In order to overcome this "vanishing problem", a LSTM cell has been used instead of a normal neural cell (S. Hochreiter, 1997). A LSTM cell is formed by a memory block which consists of a memory cell and three sigmoid gates including an input gate, output gate, and forget gate. This special structure enables LSTM recurrent networks to learn long-term dependencies.

The idea of using LSTM RNNs for DGA detection has been used successfully by Woodbridge et al. (Woodbridge et al., 2016). The input for the RNN is a sequence of characters from the domain name S. The embedding layer transforms the L-length sequences of input characters to a sequence of vectors R^{dxl}, where l is the maximum length determined from the training set. This way, the LSTM RNN model accepts variable-length character sequences as input, so there is no need for data preprocessing for normalization of input. After embedding, the vectors are fed into a LSTM layer that contains n LSTM cells. This layer will process the input vectors one by one and update a hidden state vector at each step. Outputs of these n LSTM cells in the LSTM layer will establish our character-level features.

4.2. Morphological Features

In our proposed approach, we also utilise specifically selected features to provide measurable traits as to the legitimacy of a domain name. Our approach focuses on the linguistic structure of the domain name rather than any literary meaning. There are existing solutions that utilise linguistic features, but none offer such a holistic and comprehensive approach as that proposed in this paper. These features are listed in this subsection, and a brief explanation and justification is given for each. It must be noted that several features discussed can be present in both legitimate and DGA-based domains. However, when combined with the presence of another feature they can provide a more reliable indication. This therefore emphasises the importance of using the proposed holistic approach.

• Domain length: This is the alphanumerical length of the SL. Typically, legitimate domain lengths are relatively short for memorability purposes, whereas those generated by DGAs are often longer to avoid potential clashes with existing domains. Therefore, the excessive length can be indicative of a DGA-generated domain. For example, the average length of the SLs in our benign dataset is 7.01 characters, whereas the average SL length of our DGA-based SLs is 14.43 characters.

• Vowel-Consonant ratio: This measures the balance between the number of vowels and consonants in a SL. Generally, in English, the number of consonants is greater than the number of vowels. An uneven distribution of consonants to vowels can often be indicative of a domain created by a random-based DGA. The calculation of this ratio is shown in equation 1.

$$R = \frac{|V|}{|C|} \quad (1)$$

Here, R is the decimal ratio value, |V| is the number of vowels and |C| is the number of consonants.

• Maximum consecutive consonant count: The maximum number of consecutive consonants can provide an indication

as to the unusual morphology of a SL. In English, typically it is rare to see more than 7 consecutive consonants in a single word. DGAs that use either randomisation or partial word concatenation can often produce such results. Therefore, examining the maximum number of consecutive consonants can provide a useful insight. For example, "xzkprst" contains a long consonant streak, contrasting with a legitimate domain like "strengths.

• Number count: The number of numerical characters in the specified domain label. Typically, SLs tend to utilise characters rather than numbers (but there are numerous notable exceptions). A high count of numbers, particularly in a longer SL is a notable characteristic of DGA-based domains.

• Hyphen count: From our research, we have identified that currently DGAs seldom use hyphens in their domain names. So (for now at least) this can provide an indicative measure in determining legitimacy. For example, "my-blog" includes a hyphen typical of legitimate use, while "xyz123" does not.

• Repeated consonants: This provides an aggregated count of the number of repeated consonants in the SL, weighted by its length (implemented by dividing it by the label length). Our analysis indicated that the number of repeated consonants is recognisably higher in random-based DGA-created domains, e.g., "xkkppz" versus "apple".

• Repeated vowels: This provides an aggregated count of the number of repeated vowels in the SL, weighted by its length (implemented by dividing it by the label length). Our analysis indicated that the number of repeated vowels is recognisably higher in dictionary-based DGA-created domains, such as "goooogle" versus "google".

Table 3. Letter frequency weighting scores.

Letter	Score	Letter	Score	Letter	Score
a	0.08	j	0	s	0.06
b	0.02	k	0.01	t	0.09
c	0.03	l	0.04	u	0.03
d	0.04	m	0.02	v	0.01
i	0.07	r	0.06		

• Letter score: This is a quantifiable measure of how common the letter composition of a SL can be considered. The idea behind this is to identify highly unusual or random letter combinations. Each of the 26 alpha characters in English is assigned a score based upon its usage frequency within the SLs of the current Alexa dataset, the results of which are shown in Table 3.

We then construct an aggregated score, weighted by the length of the label. This process is shown in equation 2.

$$\frac{1}{n}\sum_{i=1}^{n}\left(S_{c_i}\right) \tag{2}$$

Here, n is the character length of the label, S is the score, c is an individual character and i is an iterative value.

• Dictionary count: The number of in-line (i.e. no anagrams) dictionary words that can be identified within the SL. The dictionary used also includes commonly-used abbreviations and acronyms of web terminology (e.g. cdn, img). Randomly generated strings often contain a considerably lower number of dictionary words for their length. Similarly, some DGAs that utilise whole dictionary words, have the opposite effect in that they have an abnormally large number of words.

• Average dictionary length: The average character length of the dictionary words identified. Using our benign dataset, we have calculated the average length to be 3.1 characters. Drastic deviations from this average can indicate unusual characteristics.

• Maximum dictionary length: The maximum character length of the dictionary words identified. Long dictionary words can be an indication of dictionary DGA-created domains.

• Scaled n-gram score: We calculate an n-gram score to express the probability of a genuine morphological composition of a SL. This n-gram model is constructed using the training data. The score is calculated using the probability of a character-set of size n occurring when given a starting character-set of the same size. Here, we used the sliding window approach to calculating this score. Based on our evaluations, we have found that using a scalable value for n yielded far superior results than using a fixed value. Furthermore, we found that by scaling the value of n with the length of the label, we could better detect the aggregated word techniques used by dictionary-based DGAs. The value of n is scaled to compliment the size of the SL length. In this instance, we have used the square root of the SL length as the value of n as this offered the most stable result. This process is shown in equations 3 and 4. Firstly, in equation 5 we will create the character-sets using the supplied label. The size of these labels is determined by the square root of the label length.

$$\forall j \in \left\{0, \dots, |L| - \sqrt{|L|}\right\}. h_j = \left(L_j, L_{j+1}, .., L_{j+\sqrt{|L|}}\right) \tag{3}$$

Here, L is the domain label, $|L|$ refers to the length of the domain label, h is the character-set and j is an iterative locator for each character position.

$$P\left(h_i|h_{i-1}\right) = \frac{count(h_{i-1},h_i)}{count(h_{i-1},*n)} \tag{4}$$

Here, P is the probability, h is the character-set, count is a function that returns the number of occurrences of a specified character-set sequence within the n-gram model, $*n$ is a wildcard for any character-set of length n, and i is an iterative value.

• Entropy: Information entropy is used to measure the degree of randomness of the character composition in the domain label.

$$H = -\sum_{i=1}^{|L|} P(L_i).\ln(P(L_i)) \tag{5}$$

Here, L is the domain label, $|L|$ is the length of the label, P is the probability mass and i is an iterative value.

5. EVALUATION

5.1 Datasets

In this paper, we will be using two different datasets for benign and DGA-based domain names. For the benign domain dataset, we are using domain names provided by Alexa's Top 1 Million Sites. For the DGA-based domain dataset, we use

domain names obtained from the opensource DGA domain repository (Aeva, 2023) and the Shadowserver feed (Shadowserver, 2023). This dataset currently contains 812,622 domains and offers a comprehensive encapsulation of the main DGA strategies. All domains are real world samples from various malware, including Cryptolocker, Zeus, Pushdo, Rovnix, Tinba, Conficker and Matsnu.. The number of DGA domain types in the dataset is presented in Table 4.

In total, there are 295432 domains (both benign and malicious) in the dataset. Of which, 70% is the training data set, and 30% is the testing dataset.

5.2 Experimental results

Initially, a case study was conducted to verify whether our idea of basing DGA-based domain detection on morphological features was viable. So we used real-world data to construct a training a small dataset of 4000 entries and test dataset of 1000 entries, thus adhering to the recommended 80:20 split. We used the features outlined in the previous section to evaluate the performance of seven of the most widely accepted classifiers, repeating each experiment 30 times (which is the accepted standard level of repetition). The seven classifiers used were Random Forest (RF), k-Nearest neighbour (k-NN), Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Naive Bayes and Polynomial.

All evaluations have been conducted using the Python version of GPU-enabled Tensorflow and have been run on an Nvidia GTX 1080Ti graphics card. The performance will be measured using:

• True Positive Rate (TPR): measures the proportion of positives that are correctly identified (formula 6)

• False Alarm Rate (FAR): measures the proportion of negatives incorrectly identified as positive (formula 7).

$$TPR = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{6}$$

$$FAR = \frac{\mathrm{FP}}{\mathrm{FP+TN}} \tag{7}$$

Here, TP is the number of true positive domains, TN is the number of true negative domains, FP is the number of false positive domains, FN is the number of false negative domains. The selection of optimal hyperparameters for the LSTM models plays a crucial role in the final performance. In this study, we employed a grid search process combined with 5-fold cross-validation on the training set to determine the most suitable set of hyperparameters. The search spaces included the number of units in LSTM layers (testing values 64, 96, 128, 144), dropout rate (0.2, 0.3, 0.5). The final set of hyperparameters that yielded the best validation performance was selected and used for all performance evaluations presented in this paper.

The results obtained are presented in Table 6. As can be seen in Table 5, the polynomial classifier results have been categorised as 'DNF'. In this instance, the classifier failed to complete within an acceptable time-frame. This is significant as the aim of the idea is to provide real-time processing of the domain names. The results in Table 6 also show that the LDA and RF classifier offered the best performance in TPR and FAR, consequently of those evaluated. However, the LDA classifier has one major disadvantage, which is that it uses an inflexible decision boundary (i.e. linear). This poses a significant disadvantage if the data shape is unknown in advance, or if the data does not conform. In our situation, both of these cases will apply, so we will be adopting the Random Forest (RF) classifier as a classifier of our proposed model.

We will be comparing four different models as part of our evaluation: our proposed HDDA model, LSTM with no morphological features (LSTM), Bigrams with SVM and Recurrent Neural Networks (RNN) which are commonly used technique in linguistic analysis). In addition to TPR and FAR, we will use Area Under the Curve (AUC) to measure the performance of the model across various classification thresholds. AUC is the probability that a randomly chosen positive instance is ranked above a randomly chosen negative instance. This is considered as a measure of classification accuracy. The results obtained are presented in the following Tables 6. As can be seen from our results, our HDDA approach has achieved higher levels of accuracy and lower levels of false alarm than LSTM, Bigram and RNN.

Although this improvement appears slight, when considering the number of domains it could potentially be analysing, such an improvement could have significant benefits.

Table 4. DGA domains in the evaluated dataset.

DGA	Number	DGA	Number	DGA	Number	DGA	Number
Cryptolocker	10000	Necurs	8192	Mydoom	10000	Pykspa	9983
Pushdo	9990	Simda	10000	Nymaim	9984	Qadars	9600
Tinba	10000	Lockyv2	9999	Padcrypt2	9984	Symmi	9984
Matsnu	10000	Ramnit	10000	Prolikefan	10000		

Table 5. Classifier comparison.

	RF	k-NN	SVM	LDA	QDA	Naive Bayes	Polynomial
TPR	0.925	0.936	0.828	0.973	1	0.907	DNF
FAR	0.115	0.143	0.130	0.138	1	0.136	DNF

Table 6. DGA detection model comparison.

	Bigram	RNN	LSTM	HDDA
AUC	0.936	0.956	0.969	**0.973**
TPR	0.951	0.945	0.977	**0.980**
FAR	0.079	0.032	0.040	**0.031**

6. DISCUSSION

While the evaluation of HDDA was conducted on static datasets to ensure controlled and reproducible results, we recognize the absence of real-time, large-scale deployment testing as a limitation in assessing its practical scalability. The design of HDDA, however, is inherently suited for handling large-scale DNS traffic due to its reliance on lightweight Random Forest classifiers and the parallelizable nature of LSTM feature extraction. For example, the morphological analysis can be precomputed and cached, reducing runtime overhead, while the model's avoidance of deep packet inspection minimizes latency in high-throughput environments. Theoretically, HDDA could process millions of DNS queries per day on modern hardware, though this claim awaits empirical validation. Future studies will prioritize deploying HDDA in a live DNS infrastructure to quantify its performance under real-world conditions, such as fluctuating traffic volumes and adversarial attempts to evade detection.

Monitoring DNS traffic to detect Thingbots raises ethical considerations, particularly regarding user privacy and data misuse. Unlike traditional methods that rely on invasive deep packet inspection, HDDA mitigates these concerns by analyzing only domain name metadata—such as length, character distribution, and linguistic patterns—without accessing payload content. This design aligns with privacy-preserving principles and reduces the risk of exposing sensitive user information. However, even metadata analysis could potentially be exploited for profiling or surveillance if not governed by strict policies. We advocate for the deployment of HDDA within frameworks that ensure transparency, such as anonymizing query origins and adhering to regulations like GDPR. By prioritizing non-intrusive detection, our approach strikes a balance between cybersecurity needs and ethical obligations, though broader discussions on DNS monitoring policies remain essential

7. CONCLUSIONS

In this paper, we have presented a novel hybrid features extraction using morphological-based and deep learning solution for detecting DGA-based Thingbots through the analysis of DNS traffic. To accomplish this, we have focused on the morphological aspects of domain names. The paper has outlined our key morphological features and our LSTM-based character-level feature extraction method. These features provide a holistic, adaptable and privacy-friendly approach DGA-domain detection.

It is clear from our evaluation results that our proposed approach has proved to be both feasible and accurate. This is evident from the consistently low FARs and high TPRs. It has demonstrated that it can also offer improvements over a standalone LSTM approach. The use of real-world datasets in our evaluations has added an extra dimension of relevance to our results and subsequent claims.

We believe that our solution and its novel contributions build upon the main limitations of existing work, as outlined in Section 3. Our solution can offer increased performance and scalability due to the lack of complexity of the approach. The morphological and observational features used are not invasive, therefore, we are able to preserve the privacy of end-users. The solution covers a wide-range of different features, which helps to ensure a more holistic view and that no single characteristic is dependent upon. Finally, we have demonstrated the performance capabilities of the solution to analyse DNS response packets.

For our immediate future work, we are looking to perform a large-scale evaluation in collaboration with an ISP to further refine and validate our solution. Another aspect of our future work is to investigate and expand our solution's applicability to other languages, particularly those using non-Latin characters. We hope that our solution will contribute to subduing the domineering threat posed by Thingbots.

ACKNOWLEDGMENT

The authors would like to thank the Ministry of Science and Technology of Vietnam for their support provided through the project "Multidimensional detection and automated response using artificial intelligence" (code NDT/CZ/24/01).

REFERENCES

A. Aeva, DGA Repository (2023). URL *https://github.com/andrewaeva/DGA/*

Anderson and R., S. (2011). A-Morphous Morphology. Cambridge Studies in Linguistics. Cambridge University Press, Cambridge. Aurangzeb, S., A.M.K.M.e.a. (2023). Cybersecurity for autonomous vehicles against malware attacks in smart-cities. *Cluster Computing*, 27, 3363–3378.

Aurangzeb, S., A.M.K.M.e.a. (2023). Cybersecurity for autonomous vehicles against malware attacks in smart-cities. *Cluster Computing*, 27, 3363–3378.

Ayub, M.A., Smith, S., Siraj, A., and Tinker, P. (2021). Domain generating algorithm based malicious domains detection. In 2021 *8th IEEE International Conference on Cyber Security and Cloud Computing* (CSCloud)/2021 *7th IEEE International Conference on Edge Computing and Scalable Cloud* (EdgeCom), 77–82.

Beaman, C., Barkworth, A., Akande, T.D., Hakak, S., and Khan, M.K. (2021). Ransomware: Recent advances, analysis, challenges and future research directions. *Computers Security*, 111, 102490.

Bilge, L., Sen, S., Balzarotti, D., Kirda, E., and Kruegel, C. (2014). Exposure: A passive dns analysis service to detect and report malicious domains. *ACM Trans. Inf. Syst. Secur.*, 16(4).

Bureau, V.A. (2023). Hidden costs: Three critical business ramifications of digital ad fraud.

Deri, L. and Fusco, F. (2021). Using deep packet inspection in cybertraffic analysis. In 2021 *IEEE International Conference on Cyber Security and Resilience* (CSR), 89– 94.

Eftimie, S., Moinescu, R., and R˘acuciu, C. (2022). Spear-phishing susceptibility stemming from personality traits. *IEEE Access*, 10, 73548–73561.

F. Ren, Z. Jiang, X.W.e.a. (2020). A dga domain names detection modeling method based on integrating an attention mechanism and deep neural network. *Cyber security*, 3.

Ghafir, I. and Prenosil, V. (2014). Dns query failure and algorithmically generated domain-flux detection. *International Conference on Frontiers of Communications, Networks and Applications* (ICFCNA 2014 - Malaysia), IET, Kuala Lumpur.

Grill, M., Nikolaev, I., Valeros, V., and Rehak, M. (2015). Detecting dga malware using netflow. In 2015 *IFIP/IEEE International Symposium on Integrated Network Management* (IM), 1304–1309.

Guerid, H., Mittig, K., and Serhrouchni, A. (2013). Privacy-preserving domain-flux botnet detection in a large scale network. In 2013 *Fifth International Conference on Communication Systems and Networks* (COMSNETS), 1–9.

Inc., B.C.S. (2015). Security report: Blue coat research maps the webs shadiest neighborhoods.

Kumari, P. and Jain, A.K. (2023). A comprehensive study of ddos attacks over iot network and their countermeasures. *Computers Security*, 127, 103096.

Manasrah, A.M., Khdour, T., and Freehat, R. (2022). Dga-based botnets detection using dns traffic mining. *Journal of King Saud University - Computer and Information Sciences*, 34(5), 2045–2061.

Marketing, T.D. (2024). How many domain names are there? domain stats for 2024. URL *https:// diggitymarketing.com/web-hosting/how-manydomain-names/.*

Naaz., S. (2021). Detection of phishing in internet of things using machine learning approach. *International Journal of Digital Crime and Forensics* (IJDCF), 13(2), 1–15.

Nguyen, T.D., CAO, T.D., and Nguyen, L.G. (2015). Dga botnet detection using collaborative filtering and density-based clustering. In Proceedings of the *6th International Symposium on Information and Communication Technology*, SoICT '15, 203–209. Association for Computing Machinery, New York, NY, USA.

Ozdel, S., Damla Ate¸s, P., Ate¸s, , Koca, M., and Anarım, E. (2022). Network anomaly detection with payloadbased analysis. In 2022 *30th Signal Processing and Communications Applications Conference* (SIU), 1–4.

Raman, R., Tiwari, M., Buddhi, D., Trivedi, S., Bothe, S., and Ponnusamy, R. (2023). Cyber security fraud detection using machine learning approach. In 2023 *3rd International Conference on Advance Computing and Innovative Technologies in Engineering* (ICACITE), 1037– 1042.

S. Foundation, Shadowserver Foundation - Main - Homepage (2023). URL *https://www.shadowserver.org*

S. Hochreiter, J.S. (1997). Long short-term memory. *Neural Computation*, 9.

S. Yadav, A. K. K. Reddy, A.L.N.R.S.R. (2010). Detecting algorithmically generated malicious domain names categories and subject descriptors. In *IMC '10 Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 48–61.

Shahzad, H., Sattar, A.R., and Skandaraniyam, J. (2021). Dga domain detection using deep learning. In 2021 *IEEE 5th International Conference on Cryptography, Security and Privacy* (CSP), 139–143.

Sharifnya, R. and Abadi, M. (2013). A novel reputation system to detect dga-based botnets. In *ICCKE 2013*, 417–423.

Shin, S., Gu, G., Reddy, N., and Lee, C.P. (2012). A largescale empirical study of conficker. *IEEE Transactions on Information Forensics and Security*, 7(2), 676–690.

Spamhaus (2024). The spamhaus project - the top 10 worst tlds. URL *https://www.spamhaus.org/statistics/ tlds/.*

Sun X, L.Z. (2023). Domain generation algorithms detection with feature extraction and domain center construction. In *PLoS ONE*, volume 18.

Tu, T.D., Guang, C., and Xin, L.Y. (2015). Detecting bot-infected machines based on analyzing the similar periodic dns queries. In 2015 *International Conference on Communications, Management and Telecommunications* (ComManTel), 35–40.

Woodbridge, J., Anderson, H.S., Ahuja, A., and Grant,D. (2016). Predicting domain generation algorithms with long short-term memory networks. URL

Zhao, D., Li, H., Sun, X., and Tang, Y. (2023). Detecting dga-based botnets through effective phonics-based features. *Future Generation Computer Systems*, 143, 105– 117.

Zhu, X., Huang, J., and Qi, C. (2023). Modeling and analysis of malware propagation for iot heterogeneous devices. *IEEE Systems Journal*, 17(3), 3846–3857.

Zou, F., Zhang, S., Rao, W., and Yi, P. (2015). Detecting malware based on dns graph mining. *Int. J. Distrib. Sen. Netw*., 2015.