Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Detecting DGA-based botnets through effective phonics-based features[☆]

Dan Zhao [a,b], Hao Li [a], Xiuwen Sun [c], Yazhe Tang [a,d,*]

[a] School of Computer Science and Technology, Xi'an Jiaotong University, China
[b] Xi'an University of Finance and Economics, China
[c] School of Computer Science and Technology, Anhui University, China
[d] Science and Technology on Communication Networks Laboratory, China

ARTICLE INFO

Article history:
Received 22 March 2022
Received in revised form 23 January 2023
Accepted 29 January 2023
Available online 2 February 2023

Keywords:
DGA
Domain names
Machine learning
RFs
Botnet
Intrusion detection

ABSTRACT

Botnets are machines that are increasingly controlled by cybercriminals to perform various attacks. Traditional methods of defense, such as blocklisting, become ineffective because illegitimate domain names are sprung out by the domain generation algorithm (DGA) periodically and rapidly to maintain command and control (C&C) on servers. Deep learning and machine learning are candidate solutions to the problem. Deep learning methods leverage high accuracy but cost more time. Machine learning methods are qualified with high training speed in the context of frequent retraining to obtain high accuracy. However, the existing machine learning solutions cannot precisely capture the linguistic characteristics of domain names, which causes many false positives. For a comprehensive understanding of strings of domain names, we present the DOmain Linguistic PHonIcs detectioN (DOLPHIN) method, a novel method that can detect DGA-based botnets. Considering the context of detecting and the correspondence between pronunciations and spellings of words, we design DOLPHIN patterns. They are the classifications of variable-length vowels and consonants following the principles of phonics. Based on DOLPHIN patterns, a novel matching automation is used to reconstruct domain names with the components of variable-length vowels and consonants. From those domain names, DOLPHIN extracts phonics-based features. We implement DOLPHIN in supervised learning methods and compare them to the foremost methods FANCI, HAGDetector, and LSTM.MI. The experimental results show that, compared to FANCI with random forests, DOLPHIN can achieve a higher detection accuracy of 0.0265 with lower FPR and FNR without bringing much overhead. DOLPHIN is also able to generalize to other sources of data in the real world with the FPR decreasing by 0.0801 (62.97%) compared with FANCI. DOLPHIN can cooperate with most linguistic features and brings an improvement in performance compared to that of the existing linguistic feature-based methods.

1. Introduction

Botnets are distributed networks that consist of infected devices (bots), such as computers, cellphones, and Internet of Things devices, which can launch various attacks, e.g., DDoS, data stealing, and spam sending. They may make target networks or hosts inaccessible or crash. It is thus important to block botnets to prevent such attacks.

One mainstream way to achieve this goal is to identify and block the communication channel between bots and command and control (C&C) servers [1]. For example, since many IP addresses or domain names of C&C servers are hard-coded into malware binaries, one can undertake reverse engineering of binary and maintain a large blocklist of those IPs or domain names [2]. Then, the online security systems can block the connections to those IPs or domains according to blocklists.

Unfortunately, this method has become ineffective, as botnets tend to use the domain generation algorithm (DGA) [3] to periodically generate large numbers of pseudorandom domain names. These illegitimate domain names, known as illegitimate algorithmically generated domains (AGDs) [4], can change very frequently. Only a few of them are registered. Thus, they can barely be identified by fixed blocklists. DGA has become the cornerstone technique for botnets. Specifically, measurements indicate that the majority of the observed botnets use DGA as their only mechanism for communication. Most of them are valid for a short period, e.g., within only one day [5]. In addition, they may

[☆] The preliminary version was accepted at IEEE Global Communications Conference 2021.
[*] Corresponding author at: School of Computer Science and Technology, Xi'an Jiaotong University, China.
E-mail address: yztang@mail.xjtu.edu.cn (Y. Tang).

https://doi.org/10.1016/j.future.2023.01.027
0167-739X/© 2023 Elsevier B.V. All rights reserved.

spread out over different top-level domains worldwide. Hence, the identification of illegitimate AGDs is the key to blocking most botnets.

To overcome this problem, researchers have developed a variety of machine learning methods [6,7] and deep learning methods [8,9].

The deep learning methods leverage a high accuracy. However, they require training massive data and considerable time. In this situation, they are not acceptable for retraining to reduce the false negative rate [10] when DGAs produce new domains rapidly. For example, the famous Conficker botnet (version C) generated 50,000 domain names per day [11]. In contrast, machine learning methods are qualified with a short training time. They can be used in most contexts.

Most machine learning methods analyze the patterns in the strings of domain names: a legitimate domain must be meaningful to humans; otherwise, it tends to be an illegitimate AGD [8,9,12]. The major features they are concerned with include structural features (e.g., the length of domain names, the number of subdomains) and statistical features (e.g., n-gram, entropy). These features are used in machine learning classifiers. They are the crucial inputs to the results of identification since the classifiers learn from these inputs and give the mapping between them and the classified results [13]. More recent approaches [9] claim that involving linguistic features (e.g., the vowel ratio and the ratio of consecutive consonants) can boost the accuracy of the detection of illegitimate AGDs. Please note that "AGDs" hereafter refers to the illegitimate AGDs.

However, previous linguistic features might not sufficiently capture the real linguistic characteristics carried by a domain name, which may cause false positives and further lower the accuracy. Take the domain name `nationalgeographic.com` as an example. The classical methods tend to mark it as an AGD since they identify this domain name as having many repeated characters (5 in total) and consecutive consonants (`lg`, `gr`, and `ph`, 6 in total). Obviously, this is a false positive since this domain name is the official website for the famous magazine National Geographic.

Many false alarms are not allowed by a qualified botnet detection system in real-world deployment [6,14], because this would cause a frustrating user experience [6]. Therefore, when identifying AGDs, it is essential to keep the false positive rate as low as possible [14–16].

To determine the main factors that led to the above false positives, we analyze 2000 prediction results using a classical linguistic-feature-based method [9]. After inspecting the results, there were 4.4% false positive samples (88).

We believe this is primarily because classical methods use single letters as vowels and consonants to calculate linguistic features which is not reasonable. According to inspection, we find that 63.63% (56) of the aforementioned false positives have the issue that vowels and consonants are spelled in multiple letters. To address this problem, we explore the rationale behind the classification of vowels and consonants.

Classical methods classify them based on single letters, i.e., `a`, `e`, `i`, `o`, and `u` are classified as vowels, and the other letters are consonants. However, this classification is not precise, as vowels and consonants are defined by their pronunciations in words. Therefore, single characters may not fully capture the components in words. For example, `graphic` pronounces /græfɪk/, which contains two vowels, i.e., /æ/ and /ɪ/, and three consonants, i.e., /gr/, /f/ and /k/. That the pronunciation of /f/ maps to the spelling of `ph` is obvious. As a result, `ph` should be considered a whole consonant instead of two consecutive consonants. The spelling of `ph`, `i`, and `c` are called graphemes [17], i.e., one or multiple letters, which are the smallest spelling units [18]. The rationale behind the above example is phonics which gives the mapping between the graphemes and their pronunciations and proves that graphemes can be spelled with variable lengths [18–20]. Therefore, it is necessary to change the classification of vowels and consonants by following the principles of phonics so that the linguistic features can better contribute to the detection of AGDs.

In this paper, we propose DOmain Linguistic PHonIcs detectioN (DOLPHIN). We analyze and define DOLPHIN patterns, which present a novel classification of consonants and vowels. The patterns comprehend the correspondence between graphemes and their pronunciations. These patterns are elaborately selected in the context of identifying AGDs. Then, DOLPHIN identifies such patterns and precisely extracts linguistic features from domain names. In the former example of `nationalgeographic.com`, DOLPHIN views `a`, `ion`, `al`, `e`, `o`, `i` as vowels and `n`, `t`, `g` `gr`, `ph`, `c` as consonants. Since there are no consecutive consonants and only one repeated character (i.e., `a`) with the current classification, the domain can be correctly identified as benign.

The most important innovation of this research is combining phonics-based linguistic features and using machine learning for DGA-based botnet detection with a lower false positive rate. It offers an easy way to improve the performance of the existing methods that involve linguistic features. The significant contributions are listed as follows.

- To our knowledge, DOLPHIN is the first method to introduce phonics for detecting AGDs. Specifically, we propose DOLPHIN patterns in the context of identifying AGDs that classify variable-length vowels and consonants following the principle of phonics.
- We then design a novel matching automaton based on the AC algorithm to address the varied-length letters (i.e., vowels and consonants) in domain names. The phonics-based features can be extracted from the reconstructed domain names.
- We provide a concrete implementation of DOLPHIN and evaluate it with a real configuration and supporting elaboration on phonics-based features. The results show a lower false positive rate and a higher accuracy than the state-of-the-art approaches. The results also argue that the phonics-based features can contribute to a lower false positive rate and a higher accuracy.
- To reveal the performance of detecting unknown DGA-based botnets using traffic, a higher accuracy with a lower false positive rate of DOLPHIN is also shown using heterogeneous data in a real-world application.

The rest of this paper is organized as follows. The background of linguistics is introduced in Section 2. Section 3 describes the patterns and how we construct domain names and extract linguistic features. We outline the implementations in Section 4. The evaluation and experimental analysis are described in Section 5. The related works are introduced in Section 6. Finally, we conclude this paper in Section 7.

2. Background of linguistics

This section introduces the traditional and a new way of classification of vowels and consonants. It also presents why the new classification of vowels and consonants helps classify domain names.

Traditionally, words are made up of units, which are single letters within the alphabet. The single-letter units are classified into vowels (i.e., `a`, `e`, `i`, `o`, and `u`) and consonants (i.e., the rest within the alphabet).

linguists hold different opinions on phonology. They believe that the smallest units are vowels and consonants, which are varied-length characters. For example, `see` consists of two pronunciations: a consonant /s/ (i.e., single letter) and a vowel /iː/ (i.e., two characters). These units of pronunciation are called phonemes [21].

The above opinion is elaborated on in some of the following points. First, vowels and consonants are recognized by their pronunciations. Second, phonemes and graphemes (i.e., their corresponding written symbols) are highly related [18]. Third, the lengths of graphemes are variable.

Next, we explicate the second point. It is supported by the principles of phonics, which provide the indices of the particular associations of phonemes and the corresponding graphemes in English [18,19]. Let us take `measure` as an example. /m/, /e/, /ʒ/, and /ə/ are four phonemes and `m`, `ea`, `s`, and `ure` are the four graphemes. The correspondences could offer guides for classification, as well as predict graphemes with given phonemes or predict phonemes with given graphemes following the instruction of the principles. According to correspondence, graphemes are classified consistently with pronunciations. For example, `m` and `s` are consonant graphemes, and `ea` and `ure` are vowel graphemes. Thus, vowels and consonants are no longer written as single letters.

How do the new vowels and consonants affect how we compute the linguistic ratios of the strings of domain names?

First, we will elaborate on the constitution of the strings of domain names. A domain name string may consist of one or several words, and a word consists of one single or several syllables. A syllable is defined as a combination of vowels and consonants, and it is formed as a vowel preceded or followed by a consonant or combination of consonants in English [22].

Based on the above relationship, vowel and consonant phonemes affect the ratio of vowels and consonants in words in the following way. The frequency of vowel and consonant phonemes in a specific syllable is mostly fixed. Moreover, different syllables have different frequencies of vowel and consonant phonemes. Therefore, frequencies of the occurrence of vowel and consonant phonemes are related to the number of syllable structures and which structures are present. Thus, the ratio of vowels and consonants appears more accurately at specific ranges.

Accordingly, the frequency of pronunciations can be seen as the frequency of graphemes. Thus, those ratios for legitimate domain names will also appear at specific ranges since the names are composed of one or several meaningful words. The ratios for AGDs will not appear at the same ranges since AGDs are not composed of meaningful words.

Therefore, the new phonics-based classification of vowels and consonants leads to more accurate computing of linguistic ratios, and it can provide a solution to the inaccurate classification of the previous methods. Thus, it can lead to a better understanding of the real linguistic characteristics of domain names.

3. Design of DOLPHIN

This section presents the design of DOLPHIN. We first introduce DOLPHIN patterns that classify vowels and consonants based on phonics. Then, we present a new matching method to reconstruct domain names using DOLPHIN patterns. We also show how to extract phonics-based features using the new domain names.

3.1. DOLPHIN patterns

Vowels and consonants are defined by their pronunciations, and phonics connects the pronunciations with their spellings. The key to phonics is that the pronunciation of a vowel or consonant maps to a grapheme, which can be a single letter or a multigraph that consists of multiple letters. We introduce the new classification of vowels and consonants following the principles of phonics since they lead to more accurate linguistic ratios, as explained in 2.

Table 1
DOLPHIN patterns.

Type	Length	Graphemes
D-Vowel	***1***	a,e,i,o,u
	2	ai,al,ar,au,aw,ay,ea,ee,ei,er,eu,ew,ey,ia,ie,ir,oa,oe,oi,oo,or,ou,ow,oy,ue,ui,ur
	3	air,ear,eer,igh,ign,ing ion,oew,ore,our,ure
D-Consonant	***1***	b,c,d,f,g,h,j,k,l,m,n p,q,r,s,t,v,w,x,y,z
	2	bl,br,ch,ck,cl,cr,dr,fl,fr,gh,gl,gr,kn,ld,lk,mb,mn,mp,nd,ng,nk,nt,ph,pl,pn,pr,ps,qe,qu,rh,sc,sh,sk,sl,sm,sn,sp,st,sw,th,tr,wh,wr
	3	dge,gue,nch,que,shr,spl spr,squ,str,tch,thr

However, we cannot directly use all the graphemes in identifying AGDs. Some graphemes are not typical and offer little support for identification. For example, the grapheme `ed` is added to verbs to make the past tense. However, most domain names include no verb. If we use `ed` as a DOLPHIN pattern for identification, it may lead to more consecutive consonants. In this situation, the legitimate domain `reddit.com` will have three consecutive consonants `r`, `ed`, and `d`, which would be mistakenly classified as an AGD.

To this end, we select the graphemes based on the following rules. (1) We exclude multigraphs with more than 3 letters because they are infrequent, e.g., `ngue`. (2) We exclude the graphemes that are rare, e.g., `sch`. (3) We exclude the graphemes that do not conform to domain naming conventions by humans, e.g., `ed` and `es`. (4) We exclude the graphemes that are consecutively repeated letters, e.g., `cc`, `ss`, and `cch`. These consecutively repeated letters may cause fewer consecutively repeated letters when calculating features of AGDs.

Specifically, according to the above rules, we define DOLPHIN patterns based on the principles of phonics [18,19,23] and mappings from the spellings of graphemes to their classification, i.e., vowels or consonants.

We name the new type of vowels and consonants D-vowel and D-consonant, respectively, as shown in Table 1. The total number of selected graphemes is 118. D-vowels consist of 5 vowel characters, 27 vowel digraphs (i.e., multigraphs with 2 letters), and 11 vowel trigraphs (i.e., multigraphs with 3 letters). Analogously, D-consonants consist of 21 consonant characters, 43 consonant digraphs, and 11 consonant trigraphs. For example, `er` in `butter` is a vowel digraph by our new classification, instead of a vowel `e` and a consonant `r`.

There are still some cases of inaccurate classification. Some graphemes have variant pronunciations. For example, the grapheme of `qu` in `quarter` is pronounced as /kw/. The same grapheme in `mosquito` is pronounced as /k/.

These differences are a concern for linguists. However, we focus on the classification of graphemes rather than specific pronunciations. The classification of most variant pronunciations for a grapheme is one and only one: either vowels or consonants, e.g., the two pronunciations of `qu` all function as consonants. Hence, the use of DOLPHIN patterns is unaffected by variant pronunciations in most cases.

Thus, unlike the traditional detection system's classification of vowels and consonants, phonics-based vowels or consonants behave as different lengths of letters to stand for vowel or consonant sounds. For example, in the patterns, the lengths of the graphemes of `s` and `er` in `butter` are 1 and 2, respectively.

3.2. Extracting graphemes with Dolphin patterns

We have defined a more accurate classification of DOLPHIN patterns. Now the remaining problem is how to adopt and locate these potential graphemes in a domain name. The essential work is to split a string into several patterns defined with DOLPHIN patterns. It is not a simple string matching problem since the lengths of the graphemes and which grapheme should be used are not fixed.

A well-known algorithm for matching multiple patterns is the AC algorithm [24]. It is a finite state string pattern matching machine. It consists of states, transitions, and 3 functions that indicate behaviors: a goto function, a failure function and a output function.

However, although it is a seemingly helpful technique, AC fails in our case for the following reasons. First, AC accepts patterns if the current state is a terminal state. If we use AC, domain names should be accepted by the single letters since the state that presents a single letter is a terminal state. Second, AC only outputs the accepted characters, while in our case, the outputs should contain every character in domain names.

To solve the above problem, we design the following strategies. First, the matching method should greedily match the patterns. For example, when addressing the word of `pair`, the D-domain name is [`p`, `air`] instead of [`p`, `ai`, `r`]. Therefore, we cannot judge whether we have read the maximum-length grapheme until we read the next character. Second, it should recognize the graphemes in a domain name from left to right since that conforms to our reading habits. Third, the characters that are not accepted should be saved for further outputting.

Accordingly, we design a new finite state machine $M = (Q, \Sigma, functions, q0, F)$ based on the classical AC algorithm, where Q is the set of all states, Σ is the acceptable characters in domain names, *functions* contain 3 functions (i.e., the goto function g, the failure function f, and the output function o), $q0$ is the initial state, and F is the set of accepted states. The construction of the output function and the behavior of the matching function are rewritten. They are shown as algorithm 1 and algorithm 2, respectively. The two algorithms will make sure to accept DOLPHIN Patterns when the goto function returns *fail*. The design of the goto function and failure function remains unchanged in our method.

The constructor of the output function gives the output of each state when the goto function returns *fail*. The output function saves the mapping between the state and the output; thus, the algorithm can query the output quickly. It first calculates all the partitions of each state from the initial state with the greedy match. The output of a state s is assigned to the partitions. Then the algorithm calculates the final output of s through the output of s minus the output of the failure state of s. The subtraction is done under the condition that the depth of the state is from the minimum to 1.

The matching function puts a character into the register by reading one from the input. The transitions of states follow the goto function. When the result of the goto function is the message of fail and the register is not empty, the algorithm calls the output function and removes the output from the register. If the result of the goto function is a state, the algorithm goes on to read the next character. After reading the entire input, the state is set to fail, and then the algorithm goes on to process the characters in the register. When the register is empty, the algorithm stops.

Algorithm 1: Construction of Output Function

```
input   : DOLPHIN pattern $dp$, failure function $f$ and
          deterministic finite automation $dfa$
1  function ConstructingOutput($dp, f, dfa$)
2      $q0.output \leftarrow$ empty;
3      for each $state \neq q0$ do
4          $str \leftarrow$ the successive letters from qo to $state$;
5          $i \leftarrow 0$;
6          while $i < len(str)$ do
7              $lenGra \leftarrow len(str) - i$;
8              for $lenGra \geq 1$ do
9                  if $str[i, lenGra]$ in $dp$ then
10                     $state.output.append(str[i, lenGra])$;
11                     break;
12                 else
13                     $lenGra - -$;
14             $i \leftarrow i + lenGra$;
15     for $n \leftarrow 1 - maxLenGrpheme$) do
16         for each $state.depth == n$ do
17             $state.output \leftarrow state.output - f(state).output$;
```

Algorithm 2: Matching Function

```
input   : string of a domain name $str$ , failure function $f$, go
          function $g$, output function $o$ and deterministic
          finite automation $dfa$
1  function Matching($i, dist, len, curState$)
2      $state, i \leftarrow 0$;  $reg.append(str[0])$;
3      $state \leftarrow g(state, str[0])$;
4      while $true$ do
5          if $g(state, ch)$ is $fail$ then
6              if $reg$ is not empty then
7                  $o(state)$;
8                  $reg.remove$(chars of output);
9              else
10                 break;
11             $state \leftarrow f(state)$;
12             if there is a $ch$ then
13                 while $g(previous, ch)$ is $fail$ do
14                     $state \leftarrow f(state)$;
15         else
16             $i + +$;
17             if $i < len(str)$ then
18                 $ch \leftarrow str[i]$;
19                 $reg.append(ch)$;
20                 $state \leftarrow g(state, ch)$;
21             else
22                 $state \leftarrow fail$;
```

We take the `graphic` as an example to elaborate on how the algorithms work. First, we build M using DOLPHIN patterns and *functions*. The state diagram of M for `graphic` is shown in Fig. 1. We only use the related states and transitions for a clear description. We also omit the failure transitions emitting to $q0$ for the same purpose. In Fig. 1, the solid arrows indicate the transition in the goto function. The dashed arrows indicate the transition in the failure function. The double circle marked with x/y means that it is in the state x, and the output of the state is y. The output of each state is calculated by Algorithm 1. For example, in the first step of the algorithm, 2 and 8 greedily match $\{r\}$ and $\{gr\}$, respectively. $f(2) = 0$, and the output of 0 is empty,

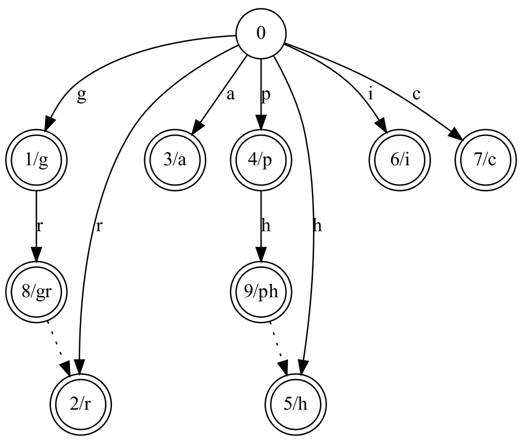

Fig. 1. The state diagram of M for `graphic`. x/y in the state circle means it is in the state x and the output of state is y.

so the final output of 2 is $\{r\}$. In the same way, $f(8) = 2$, so the final output of 8 is $\{gr\} - \{r\} = \{gr\}$.

Second, we give the operating circle of Algorithm 2 with the input of `graphic`. Let s be the current state and reg be the register for letters that have not yet been output. After it reads g and calculates $s = g(0, g) = 1$, it enters the loop of *while*. It reads r and calculates s as $g(1, r) = 8$. Then, it reads a with $reg = [g, r, a]$ and calculates s as $g(8, a) = fail$. Then, it outputs $o(8) = \{gr\}$ and sets $reg = [a]$. Now, it calculates a new $s = f(s)$ until the next result of s is not *fail*. Thus, s is set to 2, 0, 3 in turn. Then, it reads the next letter p, and the result of $f(3, p)$ is *fail*. It outputs $o(3) = \{a\}$. Analogously, $\{ph, i\}$ are outputted. After reading the last character c with $reg = [c]$, s is set to *fail* for continuing to output. Accordingly, the output of Algorithm 2 is $\{gr, a, ph, i, c\}$.

In this example, there is no digit or hyphen in the string. If there is, there should be an additional state of default characters. Accordingly, there is a transition from $q0$ to the state and a failure transition from the state to $q0$. The state is an acceptable state. Thus, all the characters in the domain name can be output.

Applying the new algorithms, M addresses a domain name with the output of a D-domain name that consists of D-vowels and D-consonants. They are regarded as a whole part in the feature calculation.

3.3. Phonics-based features

In this subsection, we explain how DOLPHIN patterns work in detecting AGDs.

The proposed DOLPHIN patterns can be used in most linguistic features since most of them are computed by the occurrences of vowels and consonants. The linguistic features equipped with DOLPHIN patterns will have a better understanding of the components of domain names. Thus, they will result in a higher sensitivity to the values of features.

Before we show how to calculate with the phonics-based features, the following definitions are given. A domain name is defined as $d = s_1.s_2.\dots.s_n.p_1.p_2.\dots.p_m$, which consists of the acceptable characters denoted as Σ. In d, $p_1.p_2.\cdots.p_m$ denotes the public suffix defined by [25] under which users can directly register domains. $s_1.s_2.\cdots.s_n$ denotes the user subdomains of d. For example, in `www.google.co.uk`, `co.uk` is a public suffix, and `www.google` is the user subdomain.

In most cases, the subdomain s_n is registered at $p_1.p_2\cdots p_m$ by an organization or person denoted as the top-user subdomain. The subdomain $s_1.s_2\cdots s_{n-1}$ is named by person or algorithm, which is benign and referred to as the host subdomain. For example, `facebook` is the top-user subdomain of `www.facebook.com` and `www` is the host subdomain.

Please note that we calculate the linguistic features only on the top-user subdomain s_n since host subdomains named by algorithms may not be meaningful and cannot be recognized well by DOLPHIN patterns.

Based on the above definitions, we choose 3 representative linguistic features to analyze: the vowel ratio, the ratio of repeated characters and the ratio of consecutive consonants.

The vowel ratio is calculated as the ratio of the number of vowel characters to the length of a domain (we use domain to represent the top-user subdomain in the rest of this section). The vowel ratio of `facebook.com` for character-based methods is 4/8 since `facebook` has 4 vowel characters. DOLPHIN yields 3/7 since it has 3 vowels and considers the `oo` as a whole part.

The ratio of repeated characters is referred to as the ratio of the number of characters that are repeated throughout a domain to the number of characters that appeared in the domain. In the previous example, this feature evaluates to 1/7 in the character-based methods. In contrast, the value is 0 as calculated by DOLPHIN.

The last linguistic feature, the ratio of consecutive consonants, is defined as the sum length of several successions (whose length is greater than or equal to 2) of consonant characters divided by the length of a domain. In the `google` example, ratio of consecutive consonants of the character-based methods is calculated as 2/6. The corresponding value by DOLPHIN is calculated as 0. Because `gl` is considered a whole part, there are no consecutive consonants.

There are some theoretical characteristics of the features based on DOLPHIN patterns. A legitimate domain tends to have a larger vowel ratio, a smaller ratio of repeated characters, and a smaller consecutive consonants. In contrast an AGD has a high probability of a smaller vowel ratio, a larger ratio of repeated chars and a larger consecutive consonant. More examples and analysis are further elaborated on in 5.3.

Some benign domain names that do not consist of words meaningful to humans can hardly be improved by phonics-based features. For example, some domains are the abbreviations of organizations or numbers. The question exists in most linguistic-based machine learning methods and sequence model-based (e.g., LSTM) deep learning methods. Their inputs are highly related to the sequences of domain names. They can be identified by combining side information. We will explore this question in our future work.

4. Implementation

To prove the effectiveness, we implement DOLPHIN and phonics-based features with machine learning techniques in the detection of DGA-generated domain names from legitimate domain names.

We present an overview of the implementation and the way we use DOLPHIN in Fig. 2. We employ supervised learning techniques to detect AGDs. According to the order of processing steps, it is made up of a preprocessing module, DOLPHIN, a feature extraction module, a training module, and a classification module.

In the preprocessing step, negative data are cleaned by deleting the samples without any public suffix. Domain names that do not end with any public suffix are unacceptable for DNS systems. Usually, these domain names are produced by mistyping or misconfiguration, so they cannot apply the principle of phonics, and they are not legitimate domains.

After that, DOLPHIN addresses the preprocessed data and yields top-user subdomains and then reconstructs the D-domain names following DOLPHIN patterns.

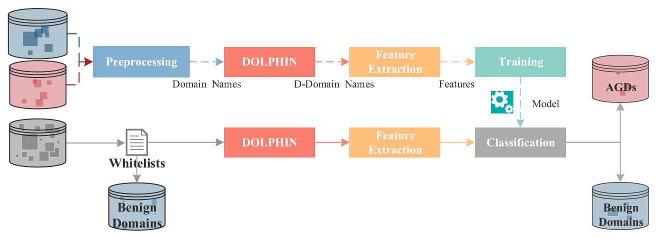

Fig. 2. The implementation of the detection of AGDs.

Then, we extract linguistic features, structural features, and statistical features from domain names in the feature extraction module. The structural and statistical features used are the same as those of other methods, such as the lengths of domain names, the numbers of subdomains, and entropy. They are straightforwardly extracted from domain names. The three linguistic features related to vowel and consonant letters are extracted from the D-domain names.

Next, in the training module, features are trained by a classifier, which yields a trained model.

Finally, in the classification module, the trained model reads a submitted domain name. It extracts features from the corresponding D-Domain name and predicts whether the domain is an AGD.

The classification model can finally be used in intrusion detection systems that detect DGA-based botnets in real time. In a real-world deployment, allowlists should be used to filter frequently used legitimate domains first to accelerate detection and enhance accuracy. These will be discussed in the future since they are outside the scope of this paper.

5. Evaluation

In this section, we evaluate DOLPHIN. The experiments are performed over a series of datasets and aim to answer the following questions:

(1) Can DOLPHIN achieve a higher accuracy, a lower false positive rate and other metrics compared to the state-of-the-art approach? We find that DOLPHIN exhibits better performance than FANCI [9] in terms of each evaluation metric on datasets with varied sizes. With DOLPHIN employed, the overall mean ACC of 5-folds increases to 0.9384 (by 0.0265) on different datasets. In particular, the false positive rate and false negative rate are reduced by 0.0220 (28.76%) and 0.0311 (31.19%), respectively.

(2) Can DOLPHIN be generalized to other classifiers? The experiment shows that DOLPHIN with XGBoost [26] and SVMs [27] also achieve overall mean accuracies of 0.9322 and 0.9229, respectively. They are 2.23% and 1.21% more accurate than FANCI with RFs [28].

(3) Can DOLPHIN be generalized to other detection methods using different linguistic features? We find that DOLPHIN can be tailored to cooperate with other linguistic features in HAGDetector [29]. The experiment shows that the accuracy of HAGDetector with DOLPHIN has an increase of 0.0201 (2.19%) compared to the original HAGDetector. The false positive rate of HAGDetector with DOLPHIN has a decrease of 0.0174 (25.15%) compared to the original method.

(4) Can DOLPHIN be generalized to applications using heterogeneous data in real cases? The experiment shows that the false positive rate of DOLPHIN has a decrease of 0.0801 (62.97%) compared to FANCI and 0.0380 (44.65%) compared to LSTM.MI [7]. The accuracy of DOLPHIN has an increase of 0.0453 (5.06%) compared to FANCI and 0.0156 (1.68%) compared to LSTM.MI.

(5) Does DOLPHIN bring additional overhead for better performance? Experiments show that the training times of FANCI and DOLPHIN using RFs are within the same order of magnitude. This means that DOLPHIN can achieve better performance without bringing additional overhead. The main factors for increasing training time are the selection of classifiers and the sample size.

5.1. Experimental setup

Datasets. The training processes of methods require labeled data. We collect multiple heterogeneous data from 4 data sources as follows.

We obtain positive samples, i.e., illegitimate DGA-generated domain names, from (1) The Open-Source Intelligence (OSINT) DGA feed [30] and (2) 360netlab [31].

We obtain negative samples, i.e., legitimate domain names, from (3) Alexa Top domain names.

We obtain (4) DNS traffic data collected from a campus network. The data last for approximately half an hour and include 53,038 DNS responses.

Preprocessing datasets. We preprocess source data for further use. To guarantee the validity of the input, we clean the negative data by deleting the samples without any public suffix. To avoid overly inflated performance, we take the following two measures. First, we remove the samples from 360netlab that are overlapping samples to reduce the correlation between the two positive data. Second, we clean the data source (4) by deleting illegitimate domain names and repeated domain names in traffic. Finally, we obtain 9727 legitimate ones.

We create two datasets after preprocessing as follows. In each dataset, the numbers of negative and positive samples are the same.

(1) In the first one, we create 40 datasets with varied sizes ranging from 500 to 20,000 using the data from the OSINT DGA feed and Alexa.

(2) In the second one, there are 9727 legitimate domain names from the cleaned traffic data and 9727 illegitimate domain names from 360netlab. The sizes of the positive and negative data are the same since the dataset is relatively balanced to reveal the metrics. The dataset is used to detect unknown AGDs in traffic.

Experimental Design. To evaluate the proposed method, we perform four experiments.

The first three experiments are conducted with a 5-fold cross validation (CV) for a less biased estimate. Each dataset is randomly split into five groups. Every group will be used as predicting data, with the remaining four as training data in turn.

(1) Models trained with RFs are performed on the first datasets with varied sizes. The purpose of the experiment is to ensure that considering phonics is the only factor that can help improve the identification of AGDs with lower FPR. Thus, the effect of other features (e.g., length of domain names) in the experiment is ruled out.

To this end, we extract three groups of features for comparison: DOLPHIN, FANCI, and the baseline. They all have 18 identical structure or statistical features. They are distinguished from 3 additional linguistic features: DOLPHIN uses phonics-based linguistic features, FANCI uses character-based linguistic features, and the baseline uses no linguistic features. Phonics-based and character-based linguistic features are calculated in different ways: they are calculated using the phonics-based and character-based classification of vowels and consonants, respectively.

These groups of features are trained, and their output models are used to classify AGDs. The classification results measure the effectiveness of DOLPHIN in detecting AGDs, if any, compared with that of FANCI.

(2) The performances of DOLPHIN with RFs, SVMs, and XGBoost are compared. We aim to demonstrate whether DOLPHIN can generalize to other classifiers in addition to RFs.

We choose these classifiers because they are representative classifiers for bagging ensemble methods, generalized linear or kernel methods, and gradient boosting ensemble methods, respectively. They have different characteristics and different advantages.

(3) DOLPHIN is applied to the linguistic features of HAGDetector [29]. The performance of HAGDetector with DOLPHIN and the original HAGDetector are compared. We aim to demonstrate whether DOLPHIN can cooperate with other linguistic features in other methods except for the aforementioned three linguistic features.

To this end, we extract two groups of features for comparison: HAGDetector and HAGDetector with DOLPHIN on the first datasets with 10,000 samples. They have 2 identical features, such as length of domain and entropy. HAGDetector has 12 other helpful linguistic features, such as the number of switches between vowels and consonants. They are calculated by the character-based classification of vowels and consonants. HAGDetector with DOLPHIN calculates these 12 features using phonics-based vowels and consonants.

(4) DOLPHIN with RFs, FANCI with RFs and LSTM.MI are compared to predict domain names in real traffic. We aim to validate whether DOLPHIN can achieve better performance than other methods in real applications.

To this end, the training and predicting data are taken from different sources. Specifically, we train DOLPHIN and FANCI using dataset (1) and predict domains using dataset (2). Therefore, we choose a one-fold model in the first experiment, which is trained on the dataset with 10,000 samples. Moreover, we also train an LSTM.MI model on the same dataset for further prediction.

All experiments are performed on an x86 PC with a 6×Intel 3 GHz CPU and 8 G RAM on Windows 10. We do not use a high-end server because the experiments focus on accuracy instead of speed.

Evaluation Metrics. To measure the quality of the methods, we introduce the evaluation metrics: accuracy, FNR and FPR. Accuracy is defined as $ACC = \frac{TP+TF}{TP+TF+FP+FN}$ and measures the ratio of correct predictions to the total number of samples. Here, TP denotes the number of AGDs correctly predicted. FN denotes the number of AGDs that are predicted to be legitimate domains. TN denotes the number of legitimate domains correctly predicted. FP denotes the number of legitimate domains that are predicted to be AGDs. The false positive rate (FPR) is defined as $\frac{FP}{TN+FP}$. This is the proportion of legitimate domains incorrectly predicted in all legitimate domains in this paper. The false negative rate (FNR) is defined as $\frac{FN}{TP+FN}$. It is the proportion of the AGDs that are predicted to be legitimate domains in all AGDs. The true positive rate (TPR) is defined as $\frac{TP}{TP+FN}$ and measures the proportion of AGDs predicted correctly in all AGDs. The true negative rate (TNR) is defined as $\frac{TN}{TN+FP}$, which is the proportion of legitimate domains correctly predicted in all legitimate domains. Among them, FPR has gained more attention in practical applications regarding user experience [14]. A higher FPR may trigger off many false alarms.

5.2. Different features with RF model

The presentation of the mean ACCs of the 5-fold results using RFs with DOLPHIN, FANCI, and the baseline are shown in Fig. 3.

Fig. 3 shows that DOLPHIN and FANCI both have larger ACCs compared with that of the baseline. Specifically, the overall mean ACCs of the two methods rise by 0.0628 (7.17%) and 0.0363 (4.15%), respectively. This illustrates that applying vowels and consonants in the linguistic features is helpful. The overall mean ACC of DOLPHIN is 0.9384 with an increase of 0.0265 (2.91%) compared with FANCI. This reveals the validity of phonics-based linguistic features; the graphemes could catch more accurate characteristics of domain names than the traditional methods. The ACCs of the above two methods tend to stabilize after the size of the dataset increases to 3000. Note that the ACC of DOLPHIN is still greater in small datasets, e.g., with a size of 2000, where the mean ACC of DOLPHIN is 0.9450.

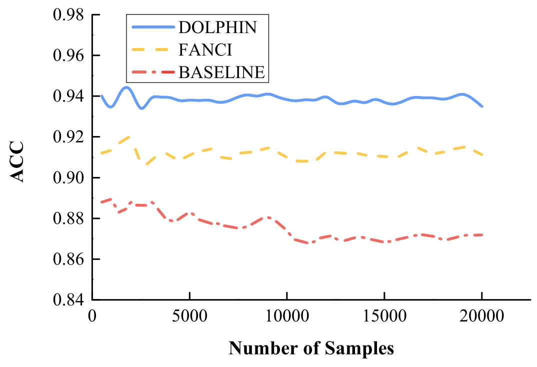

Fig. 3. ACCs of DOLPHIN, FANCI, and the baseline on different sizes of samples.

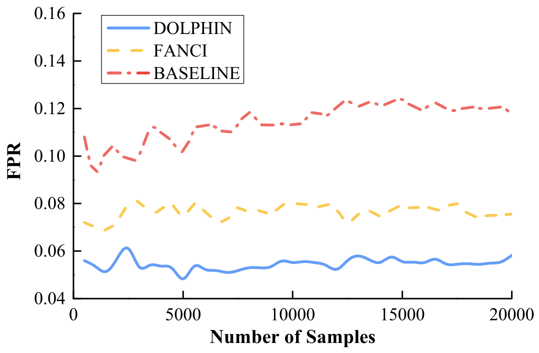

Fig. 4. FPRs of DOLPHIN, FANCI, and the baseline on different sizes of samples.

The results of FPRs among DOLPHIN, FANCI, and the baseline are shown in Fig. 4. It shows that DOLPHIN performs best, and the baseline performs worst. Specifically, their overall mean FPRs are 0.0545, 0.0765, and 0.1142 over the whole datasets, respectively. The overall mean FPR of DOLPHIN presents a reduction of 0.0220 (28.76%) compared to FANCI. As the number of samples grows, the FPRs of DOLPHIN and FANCI are more stable, but that of the baseline is growing. This means that it is more likely that the baseline method mistakenly predicts legitimate domains as AGDs, especially when using larger datasets.

The comparison for FNRs of DOLPHIN, FANCI, and the baseline is shown in Fig. 5. It shows that DOLPHIN outperforms the other methods in FNR. DOLPHIN and FANCI run relatively well compared to the baseline. Their overall mean FNRs are 0.0686, 0.0997 and 0.1347, respectively. In the worst case of the baseline (the size of the dataset is 10,500), the mean FNR of 5-fold is 0.1508. With the same dataset used, the FNR of DOLPHIN is 0.0692, representing a decrease of 33.08% compared with FANCI and 54.11% compared with the baseline. This means that applying DOLPHIN can reduce the chances of false negative errors.

The performance of FANCI in our experiments is lower than that in its original paper [9] because we do not use NXDomain as the training dataset as the paper did. DOLPHIN cannot directly use NXDomain because phonics cannot be applied to most

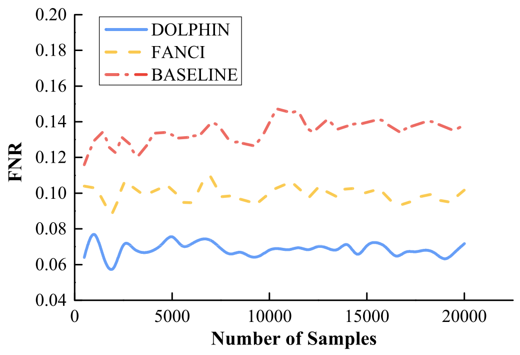

Fig. 5. FNRs of DOLPHIN, FANCI, and the baseline on different sizes of samples.

host subdomains of NXDomain. For example, one legitimate NXDomain `410.0.va8q5gjksrir5536.avts.mcafee.com` has many characters that are automatically generated, and phonics cannot be applied to it. Therefore, to control the value of a single variable in the first two experiments, we use domains that are not NXDomain to train and predict. These degradations of the performance of FANCI are also mentioned in [10] when using ALEXA as training data.

In summary, we argue that DOLPHIN can achieve better ACC and FPR by using phonics-based features. We also validate that linguistic features are helpful in the detection of AGDs.

5.3. Analysis of phonics-based features

To reveal DOLPHIN's effectiveness, we explore the relationship between the feature values of DOLPHIN and those of FANCI. We choose one-fold results for the experiment on the dataset with 10,000 samples for analyze since it manifests the median size of the datasets.

In 2000 testing samples, FP and FN of FANCI are 88 and 109, and DOLPHIN decreases those values to 48 (by 45.45%) and 69 (by 36.70%), respectively. The feature values for different methods are shown in Table 2. The *Method* – *Feature* in the first column means that *Feature* is calculated by *Method* (i.e., DOLPHIN or FANCI). *Method* – *Result* in the first two rows means that the samples are classified as *Result* by using *Method*. The values indicate the mean feature values calculated by different methods under specific conditions. For example, for the samples classified as FP by FANCI and DOLPHIN, the mean vowel ratio of these samples is 0.3556 calculated by DOLPHIN and 0.3091 calculated by FANCI.

Analysis of FP. We investigate the negative samples incorrectly classified by FANCI. With DOLPHIN employed, 56 (63.63%) of them are correctly classified, while 32 are still not able to be identified correctly by both methods.

We compare the values of the samples that are FP samples for FANCI as well as TN samples for DOLPHIN. With DOLPHIN used, the results conform with our expectations mentioned in 3.3. We summarize the observations in Table 2: the mean vowel ratio increases by 0.0678, and the mean ratio of repeated chars and the mean ratio of consecutive consonants decrease by 0.1656 and 0.2370 after DOLPHIN is employed, respectively. For each sample mentioned above, at least two feature values change. Significantly, the ratios of repeated chars of 18 samples are computed as 0, and the ratios of consecutive consonants of 7 samples are computed as 0. Only 7 samples with 9 values do not go with what we expected, regardless of 5 unchanged values. These variations in feature values lead to correct classification.

The above point is verified by the following findings. Whether DOLPHIN classifies samples correctly depends on how much variation of feature values there is. Specifically, the variation after

Table 2
Feature values. D denotes DOLPHIN, and F denotes FANCI.

Features values	F-FP		F-FN	
	D-TN	D-FP	D-TP	D-FN
D-Vowel ratio	0.4045	0.3556	0.2781	0.3639
F-Vowel ratio	0.3367	0.3091	0.3062	0.3627
D-Ratio of repeated characters	0.1019	0.1924	0.1844	0.1312
F-Ratio of repeated characters	0.2675	0.3254	0.2683	0.2054
D-Ratio of consecutive consonants	0.2602	0.4302	0.6034	0.3978
F-Ratio of consecutive consonants	0.4972	0.5653	0.5860	0.4354

applying DOLPHIN is significantly different. For the FP samples for FANCI that DOLPHIN correctly classifies, the mean percentage of variations for each domain name of the above three features are 20.14%, 61.91%, and 47.67%, respectively. In contrast, for the samples incorrectly classified by DOLPHIN (32), the mean variations are 15.04%, 40.87%, and 23.90%.

We will elaborate on the effect of DOLPHIN using the previous example of `nationalgeographic.com`. FANCI assumes that the name top-user subdomain has a length of 18 and consists of 8 vowels and 10 consonants. The vowel ratio is 8/18. The number of unrepentant characters is 12. There are 5 characters repeated in the name: `a`, `n`, `i`, `o`, `g`, and the ratio of repeated chars is 5/12. The consecutive consonants are `lg`, `gr`, `ph`, and the total number is 6, so the ratio of consecutive consonants is 6/18.

DOLPHIN considers `ion`, `al`, `gr`, and `ph` as graphemes. Therefore, the D-domain name is [`n`, `a`, `t`, `ion`, `al`, `g`, `e`, `o`, `gr`, `a`, `ph`, `i`, `c`]. In the D-domain name, the number of elements becomes 13. Now, it has 7 D-vowels: `a`, `ion`, `al`, `e`, `o`, `a`, `i`, and 6 D-consonants: `n`, `t`, `g`, `gr`, `ph`, `c`. Therefore, the corresponding vowel ratio, the ratio of repeated chars, and consecutive consonants are computed as 7/13, 1/12, and 0, respectively. Using these new feature values, DOLPHIN can now correctly classify this domain.

Analysis of FN. The work in decreasing FN has motivated us to analyze the negative samples for FANCI. Among them, 52 samples (47.71%) are correctly classified after applying DOLPHIN, and 57 AGDs cannot be identified correctly by both methods.

For these FN samples for FANCI which are also the TP samples for DOLPHIN, we observe the change from Table 2 as follows. The mean value of the vowel ratio of DOLPHIN decreases by 0.0281 compared to that of FANCI. The mean ratio of repeated chars decreases by 0.0839. The mean value of the ratio of consecutive consonants increases by 0.0174. The mean percentages of variations for each domain name of the above three features are 9.18%, 31.27% and 2.97%, respectively. As we expected, the changed values of vowels and consecutive consonants accords with the characteristics of AGDs: fewer vowels and more consecutive consonants. The value of repeated chars is slightly different from what we expected.

If DOLPHIN identifies these positive samples incorrectly (i.e., samples are FNs for DOLPHIN), the mean values for the vowel ratio and the ratio of repeated chars decrease by 0.33% and 36.13%, and the mean of the ratio of consecutive consonants increases by 8.64%.

We use three types of outcomes for features to show how DOLPHIN classifies correctly. First, at least two of the features of DOLPHIN match with the theoretical feature values (mentioned in 3.3), whose number is 24. Second, at most one feature of DOLPHIN matched with the theoretical feature values, whose number is 14. Third, there is no change in all features by using DOLPHIN, whose number is 14.

In the first type, AGDs mostly do not spell like legitimate words according to DOLPHIN patterns. For example, the D-domain name of the AGD `tferererwyatanb.com` is regarded as [`t`, `f`, `er`, `er`, `er`, `w`, `y`, `a`, `t`, `a`, `n`, `b`]. This brings the following variation: the vowel ratio from 0.3333 to 0.4167 (which is against the theoretical feature values); the ratio of repeated chars from 0.4444 to 0.3750; and the ratio of consecutive consonants from 0.4667 to 0.5000. After applying DOLPHIN, it has relatively more repeated chars and consecutive consonants, leading to correct classification.

The second and third types are classified correctly since the new model set the threshold more reasonably, and meanwhile, some of the relatively reasonable values can dominate the result. For example, the values of a third type AGD `xwfubvj.su` have no change after applying DOLPHIN. For this AGD, the vowel ratio is 0.1429, the ratio of repeated chars is 0, and the ratio of consecutive consonants is 0.8571. From the values, we can say that the value of the ratio of consecutive consonants is very high, which dominates the result. We also find that the number of unchanged feature values is much greater than in FP cases. Especially for the third type, the number of FN samples for FANCI, which are TP samples for DOLPHIN (14), is more than that of FP for FANCI, which are TN for DOLPHIN (0). That is, because the AGDs are generated randomly, they are less likely to be processed into multigraphs.

Theoretical Analysis. In summary, DOLPHIN can effectively classify domains correctly by changing the values. The feature values of AGDs mainly differ from those of legitimate domains. AGDs are generated by various kinds of algorithms. Hence, they have various characteristics that cannot easily be summarized by just one pattern. However, the structures of legitimate domain strings are relatively describable, resulting in the specific values on behalf of the structures. Therefore, the domains having those values of features in the opposite manners can be identified as AGDs. More theory can be found in 2.

Meanwhile, DOLPHIN can also classify correctly without changing all the values for the following two reasons. First, using DOLPHIN changes the length of D-domain names. Therefore, when calculating the features, the values may appear to go against the trends. Second, some values are very representative in distinguishing AGDs from other variations that are not that important in some cases. Those important values dominant the results. Third, the model trained by DOLPHIN learns more reasonable values and then sets the threshold to a reasonable value.

5.4. Different models with phonics-based features

The ACCs of DOLPHIN with RFs, XGBoost, and SVMs appear as shown in Fig. 6. The ACCs of RFs and XGBoost remain relatively stable, with overall mean values of 0.9384 and 0.9322, respectively. The overall mean ACC of SVMs is 0.9229. However, the ACC of SVMs consistently falls as the number of samples increases, which might be due to the noise in the input datasets. The overall mean ACCs of DOLPHIN using different methods are larger than the best mean ACC of FANCI (i.e., employed RFs). The ACCs for DOLPHIN with different classifiers are 2.91%, 2.23%, and 1.21% more accurate than FANCI with RFs, respectively.

We compare the FPRs of DOLPHIN using different classifiers, as shown in Fig. 7. The overall mean FPRs of SVMs and XGBoost are 0.0665 and 0.0604, respectively. The two values are smaller than that of FANCI using RFs. On the small datasets, i.e., the datasets with sizes from 500 to 2500, SVMs show the smallest FPR. XGBoost presents a greater FPR among them on datasets with sizes from 500 to 4000. On datasets larger than 2500, RFs perform with the lowest FPR. SVMs are a smart choice for small datasets if we prefer less legitimate domains mistakenly predicted. RFs are better under the condition of using more training samples.

The FNRs of different classifiers are also compared in Fig. 8. Although it shows the change of FNR on the datasets with varied

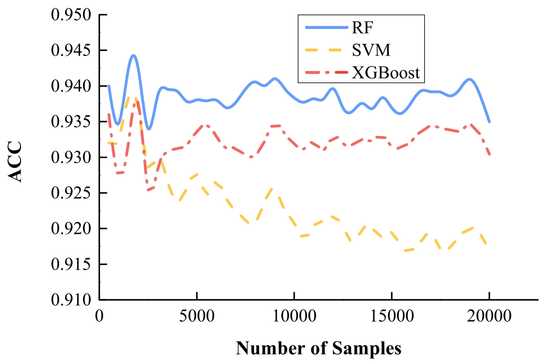

Fig. 6. ACCs of DOLPHIN using different classifiers on different sizes of samples.

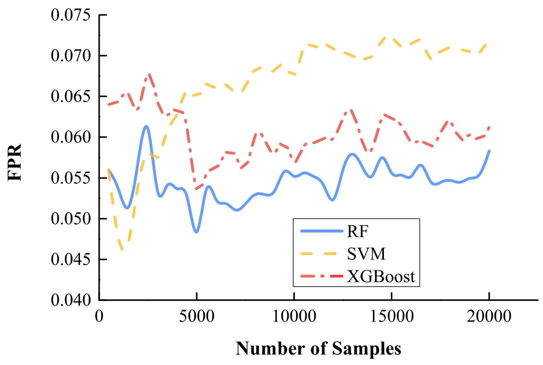

Fig. 7. FPRs of DOLPHIN using different classifiers on different sizes of samples.

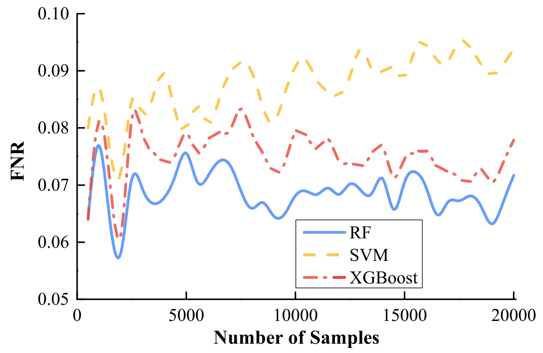

Fig. 8. FNRs of DOLPHIN using different classifiers on different sizes of samples.

sizes, the FNR of RFs can outperform on all the datasets. The lowest FNR is 0.0613 using the dataset with 19,000 samples. The overall mean FNRs of SVMs and XGBoost are 0.0876 and 0.0752, respectively.

In the experiment, we learn that the performance of the RFs classifier is generally the best. They have the largest ACC and the smallest FPR and FNR among the methods on most datasets. Meanwhile, SVMs work slightly worse than the other classifiers, which indicates performance degradation since the size of the dataset grows to 10,000. We find that the metrics of DOLPHIN with SVMs are even better than those of FANCI with RFs. Therefore, we believe that DOLPHIN can perform well with the change of classifiers; a specific classifier does not bring improvement.

5.5. Different methods with phonics-based features

This subsection presents the ability of DOLPHIN to cooperate with other methods. The performances of HAGDetector and

Table 3
Performances of HAGDetector and HAGDetector with DOLPHIN.

Method	ACC	TPR	TNR	FPR	FNR
HAG-D	0.9393	0.9304	0.9482	0.0518	0.0696
HAG	0.9192	0.9076	0.9308	0.0692	0.0924

Table 4
Generalization results.

Method	ACC	TPR	TNR	FPR	FNR
DOLPHIN	0.9399	0.9268	0.9529	0.0471	0.0732
FANCI	0.8946	0.9163	0.8728	0.1272	0.0837
LSTM.MI	0.9244	0.9339	0.9149	0.0851	0.0661

HAGDetector with DOLPHIN are shown in Table 3, where HAG-D denotes HAGDetector with DOLPHIN and HAG denotes HAGDetector. It shows that HAGDetector with DOLPHIN outperforms the original HAGdetector. The FPR and FNR of HAGDetector with DOLPHIN decrease by 0.0174 (25.15%) and 0.0228 (24.68%), respectively, compared to the original method. The ACC, TPR, and TNR of HAGDetector with DOLPHIN have increases of 0.0201 (2.19%), 0.0228 (2.51%), and 0.0174 (1.87%), respectively.

The experiment reveals that DOLPHIN can cooperate with other manually crafted linguistic features/methods with better performance. Because most linguistic features are designed with vowels and consonants, DOLPHIN has a better understanding of vowels and consonants. Thus, DOLPHIN can calculate the features more accurately. More analysis of how the phonics-based features affect the results is shown in 5.3.

5.6. Generalization

This subsection presents the ability of DOLPHIN to classify traffic data in the real world with different training data. The metrics for different methods are shown in Table 4. It shows that DOLPHIN outperforms FANCI and LSTM.MI on each metric. In detail, the ACC of DOLPHIN has an increase of 0.0453 (5.06%) compared to FANCI and 0.0155 (1.68%) compared to LSTM.MI. The FPR of DOLPHIN has a decrease of 0.0801 (62.97%) compared to FANCI and 0.0380 (44.65%) compared to LSTM.MI. This means that DOLPHIN can decrease the rate of false alarms. The TNR of DOLPHIN has a decrease of 0.0105 (12.54%) compared to FANCI but an increase of 0.0071 (10.74%) compared to LSTM.MI. This means that DOLPHIN improves the identification ability of illegitimate AGDs compared to FANCI. Thus, LTSM.MI can be used in cases requiring a better ability to identify AGDs than a good user experience. DOLPHIN is a better choice to use in the real world with a higher accuracy and FPR.

Next, we consider the reasons that FANCI and LSTM.MI cannot perform well compared with DOLPHIN in the experiment.

First and foremost, the training data are heterogeneous from the prediction data. (1) Heterogeneity in source data. The training data are primarily different from the prediction data. Particularly for the positive samples, we exclude duplicates from the predicting data. For the negative samples, the predicting and training data are collected from different sources. (2) Heterogeneity in naming behaviors. In real traffic, domains include host subdomains. They are generated by algorithms or named by programmers, and they may have different features according to the design of algorithms, coding habits, and languages from different programmers. This problem can be largely avoided by ignoring the host subdomains. Thus, DOLPHIN can achieve relatively better performance in this way.

Second, the size of the training dataset is relatively small. Supposing the size of the training dataset is large enough, the accuracy of FANCI and LSTM.MI may become much better.

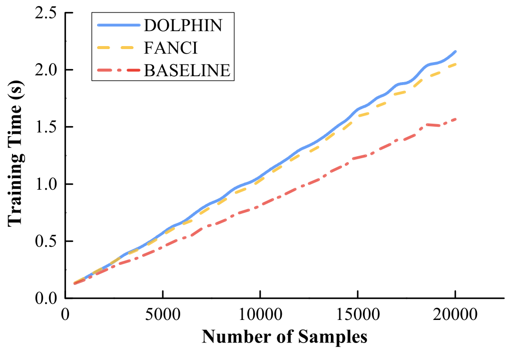

Fig. 9. Training times of DOLPHIN, FANCI, and the baseline on different sizes of samples.

Third, the negative training data of FANCI are not NXDomains. The main purpose of using NXDomain is to reduce the number of predicting samples in real cases. Instead of using NXDomain, we use all DNS request traffic as predicting data since it has the advantage that identification is done before machines controlled by C&C servers. DOLPHIN can thus be used in the intrusion detection systems for detecting known and unknown DGA-based botnets.

In conclusion, this experiment illustrates the effectiveness of DOLPHIN in the real world. Its effectiveness is less affected by generalizing to unseen data and using a small size of training data.

5.7. Training speed

This subsection presents whether DOLPHIN requires extra training time in the first two experiments.

The training times of different methods using RFs with different sample sizes are shown in Fig. 9. The figure shows that all the training times increase almost linearly as the size of the training samples increases. DOLPHIN spends slightly more training time on the same datasets than FANCI. For example, 80% of the dataset with a size of 10,000 samples, which are training data, takes DOLPHIN 1.0636 s which is 0.0331 s more than that of FANCI. It also shows that the baseline spends the least time on training step, because the number of features used in the baseline is the smallest.

The training times of the different classifiers are compared in Fig. 10. In general, the training times with different models increase at different rates. All the training times are less than one second when the size of the dataset is smaller than 9000. SVMs classifier requires more time to train on datasets with a size of over 12,500 and indicate exponential growth as the amount of data increases. XGBoost costs the minimum time when training in each dataset.

In summary, in our experiments, the linguistic features using phonics-based features are the most significant contributing factor to improving performance. Moreover, it is straightforward to generalize DOLPHIN to most linguistic features and data of varied sizes and extents.

6. Related work on AGD detection

Recent studies are mainly dedicated to the detection of AGDs [6,7,9,29], the adversarial defenses of DGA [32–36], the collaborative learning of existing methods [16], class imbalance or specific-family DGA classification [14,15,37,38], the improvement of the preprocessing stage [39], and privacy preserving and analysis [40,41].

We will present research on the detection of AGDs in terms of machine learning methods and deep learning methods. The original version is also compared with this paper in this section.

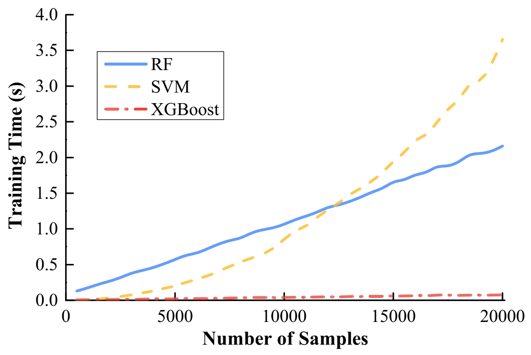

Fig. 10. Training times of DOLPHIN using different classifiers on different sizes of samples.

6.1. *Traditional machine learning approaches*

Most machine learning detection methods classify domains with manually created features. Machine learning models can be divided into two categories.

The first category is string-based methods. Pleiades [12] clusters the domains by statistical features and bipartite graphs using hidden Markov models and then classifies domains using NXDomain response traffic. FANCI [9] is one of the state-of-the-art systems [15,16,40] that uses 21 meaningful features extracted from domain names and machine learning for classification, which is the closest work to ours. It uses data obtained from NXdomain. HAGDetector [29] creatively combines traditional machine learning and deep learning methods by heterogeneous methods to detect domain names.

The second category is time-based methods that employ the chronological features of DNS transactions, e.g., time series and gaps between DNS requests and responses [42,43]. BotFinder [42] uses a machine learning model and offers the information extracted from reassembled NetFlow and traces. However, it must obtain the sequences of chronologically ordered flows first. PsyBoG [43] leverages the frequencies of botnet behavior to distinguish them from normal behaviors using power spectral density (PSD) analysis. Phoenix [8] first uses a combination of IP pools and linguistic features of domain names to cluster and identify AGDs. However, the only linguistic feature used in Phoenix is the n-gram normality score, which measures the relations within characters split by constant lengths. This is not in accord with the theory of phonemes. [44] proposes a PSO algorithm that employed features by voting on the results from a neural network algorithm, SVMs, and decision tree C4.5. The effective features are extracted from traffic, such as delay time. The work improves the accuracy of detecting botnets on the IoT dataset. While these approaches cannot be used effectively in real-time, they must collect DNS traces beforehand. In contrast, the patterns extracted from string-based methods can be directly used in an online system, e.g., an intrusion detection system [45], to identify and block C&C channels.

6.2. *Deep learning approaches*

Some approaches use deep learning techniques, e.g., neural networks, to identify illegitimate AGDs from legitimate domain names. In [6], the LSTM network-based method is first introduced for detecting DGAs. The approach only uses strings of domain names as input without any feature extraction. It can classify 90% of AGDs with a false positive rate of 1:10 000. It only takes 20 ms to predict a domain name. In [46], convolutional and recurrent neural networks (CNN and LSTM, respectively) on the NXDomain response are compared in DGA detection. To overcome some DGA families with a small size of data, which may cause multiclass imbalance problems, [7] presents the LSTM.MI method. It combines both binary and multiclass classification models to achieve higher performance in the metrics of macroaveraging recall and precision.

Some works [47,48] need to cooperate with side information, such as IP addresses, clients, and return of WHOIS lookup. This information can help enhance the performance of classifiers, but may raise privacy concerns [16] and cost extra time.

Deep learning approaches can achieve high accuracy. Some researchers argue that deep learning approaches do not perform well on new AGDs since new data are not trained by the previous model [32]. Therefore, defenders should train the model as frequently as possible to reduce FPR [10]. Machine learning methods are qualified for this term, as deep learning approaches require much more training time compared to machine learning methods.

6.3. *Comparison of previous methods*

The key novelty of DOLPHIN is that we use phonics-based vowels and consonants as the smallest units in calculating linguistic features, and we use the phonics-based features with machine learning classifiers to detect AGDs with a lower false positive rate. Thus, DOLPHIN distinguishes itself from other methods in terms of purpose, linguistic principle, features, and methods which are shown in Table 5.

Khaos [32] is an adversarial DGA instead of a detection method. The purpose of the work is different from ours. It is a very impressive work that considers domain names composed of syllables. They use the top n-grams of Alexa to split domain names, and n-gram embeddings are fed into the Wasserstein generative adversarial network. For example, they split `google` into [`goo`, `gle`]. These sequences of letters are reasonable to feed into deep learning methods. However, using n-grams to split domain names might fail to present syllables correctly. Moreover, this method uses no features and can hardly be used in linguistic features, such as the ratio of vowels.

Phoenix [8] supposes that combinations of phonemes can capture the pronounceability of domain names. It uses the phonemes to calculate the n-gram normality score and then defines a hyper-ellipsoid for detection.

In fact, the work splits domain names using n-grams and then counts the occurrences and computes the mean value. For example, it will consider the 2-gram of `google` as [`go`, `oo`, `og`, `gl`, `le`]. It is a very creative idea, but n-grams cannot truly capture the phonemes in words; e.g., `og` is not a phoneme. Meanwhile, this design is only used for calculating the n-gram normality score.

FANCI [9], HAGDetector [29] and LSTM.MI [7] all consider the smallest units of domain names to be letters, which we call character-based or letter-based. The similarity between character-based methods and DOLPHIN is that we all suppose that most benign domain names are meaningful while most illegitimate ones are meaningless. However, they are totally different in calculating linguistic features.

FANCI [9] extracts structural, statistical, and linguistic features from domain names and uses machine learning classifiers to identify AGDs. FANCI calculates linguistic features on single characters in an easy way. For example, FANCI splits `google` into [`g`, `o`, `o`, `g`, `l`, `e`] and then calculates linguistic features.

HAGDetector [29] finds that the length of domains has an impact on the performance of detection models. It splits the length of domains into three types of lengths. It uses an attention-based method to extract features and feeds them into convolution to detect short domain names. It uses an n-gram-based CNN to

Table 5
Comparison of the previous methods.

Method	Purpose	Linguistic principle	Example of split domains	Linguistic features	Classifiers or methods
DOLPHIN	Detection AGDs	Phonics-based	`g, oo, gl, e`	Involved	Machine learning classifiers
FANCI	Detection AGDs	Character-based	`g, o, o, g, l, e`	Involved	Machine learning classifiers
HAGDetector	Detection AGDs	Character and ngram-based	`g, o, o, g, l, e g, go, oo...`	Involved	Convolution, CNN, and RFs
Phoenix	Detection AGDs	Phonemes-based	`go, oo, og, gl, le`	Involved	Define Hyperellipsoid
Khaos	Adversarial AGDs	Syllables-based	`goo, gle`	None	WGAN
LSTM.MI	Detection AGDs	Character-based	`g, o, o, g, l, e`	None	LSTM

detect moderate-length domains. It also designs 14 novel features with RF classifiers to detect long domains. The method for long domains is compared in this paper because it has linguistic features.

LSTM.MI [7] uses the LSTM classifier, which employs the gating mechanism to preserve and control letters to memorize without any features. Since it is based on deep learning classifiers with no manual features, we cannot open its black boxes to improve it. We can analyze the different values of features of feature-based methods for improvement. Such methods require much more training time than methods using machine learning classifiers.

DOLPHIN considers phonics (the relationship between spellings and sounds) to split domain names; for example, DOLPHIN will split `google` into [`g`, `oo`, `gl`, `e`] and then calculate the linguistic features. To achieve this goal, we design DOLPHIN patterns, which are phonics-based vowels and consonants in the context of detecting AGDs. Then, the linguistic features are calculated and fed into machine learning classifiers for training and prediction. DOLPHIN patterns can be used in other linguistic features.

6.4. Comparison of original version

The idea of using phonics in AGD detection was first presented in our previous work in [49]. The previous work uses linguistic features and machine learning methods to detect DGA-generated domain names based on DOLPHIN patterns. This paper is an extension of the previous paper with more sufficient theoretical knowledge, explanation, and experiments with the following contributions. (1) This paper gives the background of phonics to show how phonics affects the computing of linguistic features. Additionally, this paper raises the issue that we cannot bring all phonics knowledge to detecting DGA-based botnets and gives the corresponding solutions to the issues that have not been discussed before. (2) This paper presents a novel matching automation to extract graphemes in domain names based on the AC algorithm. (3) In our previous work [49],we showed that DOLPHIN can improve performance by conducting two experiments. However, we do not analyze the causes. In this paper, we explore the different values to show how the phonics-based features affect performance. (4) This paper presents an experiment to validate the ability of DOLPHIN to cooperate with other linguistic features well. (5) This paper adds a real-world experiment, that uses heterogeneous data to provide the ability to detect unknown DGA-based botnets.

7. Conclusion

In this paper, we motivate the need for decreasing the false positive rate in linguistic-feature-based methods in the context of illegitimate AGD classification. To address this challenge, we propose DOLPHIN following the mature principles of phonics to better classify vowels and consonants. DOLPHIN patterns introduce a mapping between the vowel and consonant classification and the spelling of graphemes according to the responding sounds. Based on such patterns, DOLPHIN then identifies these patterns in domain names by a novel matching automation and extracts the linguistics features from the domain names. We conduct experiments to train those features on the same or heterogeneous data with various classifiers. The results show that when detecting AGDs, DOLPHIN can achieve a higher mean accuracy of 0.0265 and a lower mean FPR of 0.0220 (28.76%) than the state-of-the-art approach with the RFs classifier. It can also generalize to other classifiers, other linguistic features, and other data in the real world with similar improvement.

CRediT authorship contribution statement

Dan Zhao: Conceptualization, Methodology, Software, Writing – original draft. **Hao Li:** Supervision, Writing – review & editing. **Xiuwen Sun:** Visualization, Validation. **Yazhe Tang:** Supervision, Writing – review & editing.

Declaration of competing interest

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

Data availability

The authors do not have permission to share data.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China [U19B2025, 62172323, and 62102001], the Open-end Fund Project of Science and Technology on Communication Networks Laboratory [FFX22641X005], and Natural Science Foundation of Anhui Higher Education Institution of China [KJ2020A0037].

References

[1] G. Jacob, R. Hund, C. Kruegel, T. Holz, JACKSTRAWS: Picking command and control connections from bot traffic, in: 20th USENIX Security Symposium (USENIX Security 11), Vol. 2011, San Francisco, CA, USA, 2011, pp. 443–458.
[2] M. Kührer, C. Rossow, T. Holz, Paint it black: Evaluating the effectiveness of malware blacklists, in: International Workshop on Recent Advances in Intrusion Detection, Springer, 2014, pp. 1–21.
[3] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, G. Vigna, Your botnet is my botnet: analysis of a botnet takeover, in: Proceedings of the 16th ACM Conference on Computer and Communications Security, 2009, pp. 635–647.
[4] S. Yadav, A.K.K. Reddy, A.N. Reddy, S. Ranjan, Detecting algorithmically generated malicious domain names, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, 2010, pp. 48–61.
[5] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, E. Gerhards-Padilla, A comprehensive measurement study of domain generating malware, in: 25th USENIX Security Symposium (USENIX Security 16), 2016, pp. 263–278.
[6] J. Woodbridge, H.S. Anderson, A. Ahuja, D. Grant, Predicting domain generation algorithms with long short-term memory networks, 2016, arXiv preprint arXiv:1611.00791.
[7] D. Tran, H. Mac, V. Tong, H.A. Tran, L.G. Nguyen, A LSTM based framework for handling multiclass imbalance in DGA botnet detection, Neurocomputing 275 (2018) 2401–2413.

[8] S. Schiavoni, F. Maggi, L. Cavallaro, S. Zanero, Phoenix: DGA-based botnet tracking and intelligence, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2014, pp. 192–211.
[9] S. Schüppen, D. Teubert, P. Herrmann, U. Meyer, FANCI: Feature-based automated nxdomain classification and intelligence, in: 27th USENIX Security Symposium (USENIX Security 18), 2018, pp. 1165–1181.
[10] A. Drichel, U. Meyer, S. Schüppen, D. Teubert, Analyzing the real-world applicability of DGA classifiers, in: Proceedings of the 15th International Conference on Availability, Reliability and Security, 2020, pp. 1–11.
[11] N. Fitzgibbon, M. Wood, Conficker. C: A Technical Analysis, Vol. 1, Sophos Labs, Sophos Inc, 2009.
[12] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, D. Dagon, From throw-away traffic to bots: detecting the rise of DGA-based malware, in: 21st USENIX Security Symposium (USENIX Security 12), 2012, pp. 491–506.
[13] J. Cai, J. Luo, S. Wang, S. Yang, Feature selection in machine learning: A new perspective, Neurocomputing 300 (2018) 70–79.
[14] N. Gupta, V. Jindal, P. Bedi, CSE-IDS: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems, Comput. Secur. 112 (2022) 102499.
[15] A. Drichel, U. Meyer, S. Schüppen, D. Teubert, Making use of NXt to nothing: the effect of class imbalances on DGA detection classifiers, in: Proceedings of the 15th International Conference on Availability, Reliability and Security, 2020, pp. 1–9.
[16] A. Drichel, B. Holmes, J. von Brandt, U. Meyer, The more, the better: A study on collaborative machine learning for DGA detection, in: Proceedings of the 3rd Workshop on Cyber-Security Arms Race, 2021, pp. 1–12.
[17] A. Rey, J.C. Ziegler, A.M. Jacobs, Graphemes are perceptual reading units, Cognition 75 (1) (2000) B1–B12.
[18] P.R. Hanna, et al., Phoneme-Grapheme Correspondences As Cues To Spelling Improvement, ERIC, 1966.
[19] R.S. Berndt, J.A. Reggia, C.C. Mitchum, Empirically derived probabilities for grapheme-to-phoneme correspondences in English, Behav. Res. Methods Instrum. Comput. 19 (1) (1987) 1–9.
[20] M. Patricia, E. Witting, L. Stehr, Phonics They Use: Words for Reading and Writing, Pearson, 1995.
[21] G.L. Trager, B. Bloch, The syllabic phonemes of English, Language (1941) 223–246.
[22] J.B. Hooper, The syllable in phonological theory, Language (1972) 525–540.
[23] E. Fry, Phonics: A large phoneme-grapheme frequency count revised, J. Lit. Res. 36 (1) (2004) 85–98.
[24] A.V. Aho, M.J. Corasick, Efficient string matching: an aid to bibliographic search, Commun. ACM 18 (6) (1975) 333–340.
[25] Mozilla, Public suffix list, 2020, https://publicsuffix.org/ [EB/OL].
[26] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[27] C. Cortes, V. Vapnik, Support vector machine, Mach. Learn. 20 (3) (1995) 273–297.
[28] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[29] J. Liang, S. Chen, Z. Wei, S. Zhao, W. Zhao, HAGDetector: Heterogeneous DGA domain name detection model, Comput. Secur. (2022) 102803.
[30] BembenekConsulting, DGA domain feed, 2019, https://osint.bambenekconsulting.com/feeds/ [EB/OL].
[31] N.S.R.L. at 360, 2019, http://data.netlab.360.com/feeds/dga/dga.txt [EB/OL].
[32] X. Yun, J. Huang, Y. Wang, T. Zang, Y. Zhou, Y. Zhang, Khaos: An adversarial neural network DGA with high anti-detection ability, IEEE Trans. Inf. Forensics Secur. 15 (2019) 2225–2240.
[33] L. Sidi, A. Nadler, A. Shabtai, MaskDGA: An evasion attack against DGA classifiers and adversarial defenses, IEEE Access 8 (2020) 161580–161592.
[34] V. Ravi, M. Alazab, S. Srinivasan, A. Arunachalam, K. Soman, Adversarial defense: DGA-based botnets and DNS homographs detection through integrated deep learning, IEEE Trans. Eng. Manage. (2021).
[35] J. Peck, C. Nie, R. Sivaguru, C. Grumer, F. Olumofin, B. Yu, A. Nascimento, M. De Cock, CharBot: A simple and effective method for evading DGA classifiers, IEEE Access 7 (2019) 91759–91771.
[36] J. Spooren, D. Preuveneers, L. Desmet, P. Janssen, W. Joosen, Detection of algorithmically generated domain names used by botnets: a dual arms race, in: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, 2019, pp. 1916–1923.
[37] A. Drichel, N. Faerber, U. Meyer, First step towards explainable dga multiclass classification, in: The 16th International Conference on Availability, Reliability and Security, 2021, pp. 1–13.
[38] V. Le Pochat, S. Maroofi, T. Van Goethem, D. Preuveneers, A. Duda, W. Joosen, M. Korczyński, et al., A practical approach for taking down avalanche botnets under real-world constraints, in: Proceedings of the 27th Annual Network and Distributed System Security Symposium, Internet Society, 2020.
[39] M. Tong, G. Li, R. Zhang, J. Xue, W. Liu, J. Yang, Far from classification algorithm: dive into the preprocessing stage in DGA detection, in: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE, pp. 468–474.
[40] A. Drichel, M.A. Gurabi, T. Amelung, U. Meyer, Towards privacy-preserving classification-as-a-service for DGA detection, in: 2021 18th International Conference on Privacy, Security and Trust (PST), IEEE, 2021, pp. 1–10.
[41] B. Holmes, A. Drichel, U. Meyer, Sharing FANCI features: A privacy analysis of feature extraction for DGA detection, 2021, arXiv preprint arXiv:2110.05849.
[42] F. Tegeler, X. Fu, G. Vigna, C. Kruegel, Botfinder: Finding bots in network traffic without deep packet inspection, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, 2012, pp. 349–360.
[43] J. Kwon, J. Lee, H. Lee, A. Perrig, PsyBoG: A scalable botnet detection method for large-scale DNS traffic, Comput. Netw. 97 (2016) 48–73.
[44] M. Asadi, M.A.J. Jamali, S. Parsa, V. Majidnezhad, Detecting botnet by using particle swarm optimization algorithm based on voting system, Future Gener. Comput. Syst. 107 (2020) 95–111.
[45] S.A.R. Shah, B. Issac, Performance comparison of intrusion detection systems and application of machine learning to snort system, Future Gener. Comput. Syst. 80 (2018) 157–170.
[46] B. Yu, D.L. Gray, J. Pan, M. De Cock, A.C. Nascimento, Inline DGA detection with deep networks, in: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), IEEE, 2017, pp. 683–692.
[47] R.R. Curtin, A.B. Gardner, S. Grzonkowski, A. Kleymenov, A. Mosquera, Detecting DGA domains with recurrent neural networks and side information, in: Proceedings of the 14th International Conference on Availability, Reliability and Security, 2019, pp. 1–10.
[48] X. Sun, Z. Wang, J. Yang, X. Liu, Deepdom: Malicious domain detection with scalable and heterogeneous graph convolutional networks, Comput. Secur. 99 (2020) 102057.
[49] D. Zhao, H. Li, X. Sun, Y. Tang, DOLPHIN: Phonics based detection of DGA domain names, in: 2021 IEEE Global Communications Conference (GLOBECOM), IEEE, 2021, pp. 01–06.

Dan Zhao is a Ph.D. student in the School of Computer Science and Technology at Xi'an Jiaotong University and works in Xi'an University of Finance and Economics. Her research interests are network measurement and network security.

Hao Li received his Ph.D. degree in computer science from Xi'an Jiaotong University in 2016 and is now an associate professor in the School of Computer Science and Technology at the same university. His research interests are programmable networks and high-performance network functions.

Xiuwen Sun received his Ph.D. degree in computer science from Xi'an Jiaotong University in 2019 and is now an assistant professor in the School of Computer Science and Technology at Anhui University. His research interests are computer networks and network security.

Yazhe Tang received his Ph.D. degree in computer science from Xi'an Jiaotong University in 2002. He is now an associate professor of the School of Computer Science and Technology, Xi'an Jiaotong University. His research interests include software defined networking and network security.