Received September 29, 2018, accepted November 4, 2018, date of publication November 12, 2018,
date of current version December 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2880884

Identifying Fast-Flux Botnet With AGD Names at the Upper DNS Hierarchy

XIAO-DONG ZANG[1,2,3], JIAN GONG[1,2,3], SHAO-HUANG MO[4], AHMAD JAKALAN[4], AND DE-LIN DING[1]
[1]School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China
[2]Jiangsu Provincial Key Laboratory of Computer Network Technology, Southeast University, Nanjing 211189, China
[3]Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, Nanjing 211189, China
[4]School of Computer Science and Engineering, Southeast University, Nanjing 211189, China

Corresponding author: Xiao-Dong Zang (xdzang@njnet.edu.cn)

This work was supported in part by the Jiangsu Key Laboratory of Computer Networking Technology and the Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, in part by the National Natural Science Foundation of China under Grant 61602114, in part by the Key Research and Development Program of China under Grant 2017YFB0801703, and in part by the CERNET Innovation Project under Grant NGII20170406. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of those sponsors.

ABSTRACT Modern botnets rely on a new DNS technique called fast-flux to organize compromised hosts into fast-flux service networks (FFSNs), which helps bot herds to hide their upstream servers. Given the prevalence of this mechanism, various approaches have been proposed to detect them by analyzing DNS traffic. However, these detection mechanisms either have low detection accuracy or long detection latency. Moreover, they cannot capture the behavioral regularity and other novel traits of the evolving behavior of each botnet which is being tracked. This paper proposes a new FFSNs detection scheme to solve the above-mentioned limitations. The proposed approach can recognize groups of domains generated by a domain generation algorithms or its variants that are representative of different botnets. In addition to that, it can also identify whether the algorithmically generated domain names in a cluster are using fast-flux technology or not by applying a double-stages detection mechanism. The proposed work is implemented in a real network (China Education and Research Network), and the DNS traffic is collected from backbone routers. Experimental results demonstrate that our algorithm has significantly increased the detection accuracy compared with similar works and reduced the computational complexity.

INDEX TERMS AGD detection, fast-flux service networks, DNS, Botnet, network security.

I. INTRODUCTION

As an important part of infrastructure of Internet, DNS performs the reciprocal task between domain name and IP. In addition to normal activities, a variety of malicious activities, such as botnets [1], use DNS service either to parse the IP of the command and control server (C&C) hidden behind the zombie agents. Botnets are a constant threat to the integrity of individual computers on the Internet [2], such as sending spam mail, launching distributed denial of service (DDoS) attack and hosting phishing sites. Hence one can see that domains play a crucial role in the construction and launching of malicious activities. Therefore, it is of a great significance to effectively detect and identify the domains used in these malicious activities.

Internet miscreants are always seeking for new methods to cover their traces while preserving illicit channels with the aim of maintaining high availability of their service. Two DNS techniques have been used by cyber-criminals, one is called domain fluxing, which uses different DGAs to generate a great number of seemingly random domain names dynamically based on a given seed, and only chose a small subset for actual C&C communication [3]–[5]. This makes it very compelling for bot herds to harden their infrastructure. The other is fast-flux, which is similar to content delivery networks (CDNs), whereby the domains resolved to multiple IPs related to the best nodes provide copies of data all over the world [6]. However, fast-flux can be considered as illegitimate CDNs, as it uses Round Robin Domain Name System (RRDNS) for load balancing by returning multiple IP addresses after each query to a single domain. The main difference between fast-flux and CDN is that the nodes in CDN are high performance machines with highly reliability and are tightly controlled by administrators. While, the nodes in fast-flux are typically malware-compromised machines, these bots usually turn on and off making bot herds hard to control over them. When a victim visits a domain hosted by a FFSN botnet, the traffic is redirected to the malicious website by these bots. Therefore, fast-flux is resilient even when some flux agents are blacklisted.

Several approaches have been proposed to detect fast-flux domains so far. Currently, there are two main techniques existing for fast-flux detection, the active and the passive. In the active probing approach [7], [8], the domains are extracted from malware domain blacklists or spam emails, then repeatedly query to collect their resolved IPs. In passive DNS traffic analysis method [9]–[14], whereby the domains extracted from "below" local recursive DNS (RDNS), namely the DNS traffic from single users to the local RDNS servers, which contains all the DNS traffic including the traffic in DNS cache; or extracted from "above" local RDNS servers, namely the DNS resolution traffic of a DNS server. Then, some features are extracted to classify them as being fast-flux or not. Two different goals contained in these techniques: the first one is to reduce the detection time to few seconds in only one DNS response by using real-time detection mechanism [12]–[14]. However, the considerable focus on cutting the detection time would cause lots of false positives as the similarities of FFSN and CND; the other is to improve the accuracy by using long-term characteristics [9]–[11], but it requires hours before a conclusion is drawn. Moreover, tracking the domains, especially in a low bandwidth environment, would consume too much time and resources. Although, previous works can match the basic behavior of FFSNs, most of them consider single domain independently from each other, ignoring the fact that many flux networks involve more than one domain name [10].

The proposed approach in this paper regards flux networks involving more than one domain name with the aim of achieving a quick and accurate detection mechanism. For this purpose, it firstly recognizes groups of domains generated by a DGA or its variants that are representative of the respective botnets, and then identifies whether the AGD names in a cluster are using fast-flux technology or not. As a result, the regularity and other novel knowledge of the evolving behavior of each botnet can be tracked. More specifically, after analyzing live traffic collected from the upper DNS hierarchy, we find that different DGA-generated domains are distinguishable in terms of literal composition. Based on this phenomenon, two different similarity standards including the literal features and the edit-distance similarity of domains are applied to cluster them respectively. The former is used to measure the character frequency distribution and structural feature of the domain, while the latter is used to measure the number character changes needed to convert from one domain to another. Then, correlation analysis is performed in a sliding window with fixed size, so that the elements in one cluster are more likely to be related to a specific DGA-generated domain. After that, a double-stages fast-flux detection mechanism is devised. At the online stage, four types of characteristics such as *TTL*, whois, the entropy of the utilized location and attribution of their resolved IPs and the spatial service relationship are used to construct an online classifier using extreme learning machine (ELM). This stage is efficient, as the fast-flux domains can be found in a few seconds. In order to improve the accuracy and reduce the false negative, we continuously monitor the changes of the resolved IP pool and the correlation to the known malware in the offline stage. Once the number of IPs associated with a domain cluster or the changes of network diversity reach to a predefined threshold within 24 hours, the conclusion can be drawn. As the AGD domains are highly elusive, so, short-term monitor window are used to capture their fluxiness. Specifically, the contributions of this paper include,

(1) The domains generated by a DGA or its variants are effectively identified through clustering correlation. Although, it is common to use cluster algorithm to group the domains, correlating the two outputs of different cluster algorithms have not been found. Compared with the previous research [5], the algorithm in this paper is more comprehensive and accurate in recognizing specific DGA-generated domains.

(2) Some new metrics are suggested in fast-flux domain names detection, which are the entropy of utilized location and attribution of their resolved IP, the spatial service relationship and the degree of dependency of the domains to end users. Experiments show that these metrics can shorten detection time and increase the detection accuracy than the metrics used in existing reference schemes.

(3) Experiment results with actual measured dataset show that the proposed detection approach has a better tradeoff between efficiency and latency. It provides significantly detection accuracy by the complementary of offline detection mechanism and the computation time is reduced by 25% and 21% respectively compared to existing approaches of fast-flux hunter and FluxBuster.

The remainder of the paper is organized as follows. Section 2 introduces the background and related work of fast-flux botnet detection. In section 3, the proposed methodology of identifying DGA-generated domains and detecting fast-flux botnets are described in detail. In section 4, we validate our approach by applying the described techniques with the real data set. The conclusion of the paper is given in section 6 after a brief discussion in section 5.

II. BACKGROUND AND RELATED WORK

In this section, the technical details of FFSNs, legitimate CDNs and RRDNS are presented. As the similarities among these techniques, legitimate CDNs and RRDNS are often misclassified as FFSNs. Then, a number of researchers of detecting fast-flux domains are provided.

A. FAST-FLUX SERVICE NETWORKS

Fast-flux is a DNS technique used by bot herders to organize and sustain their compromised hosts into a high-availability and load-balancing network. In FFSNs, a fully qualified domain name (FQDN) associates to multiple changed IPs in order to circumvent traditional defense strategies such as IP blacklists [8]. Those IP addresses are usually called flux agents, which redirect the traffic of a victim to a protected hidden server. In addition, the TTL in the DNS response is always configured with short values, so that for subsequent queries to the same FFSN domain, different IP will be returned. There are two different FFSNs types: single-flux and double-flux. The former is the simple type of fast-flux, where the IPs of the flux agents registered are their A records providing extra protection of their mothership servers. The latter is more sophisticated, which register the bots as their name servers, so that the hiding authoritative name servers can be protected by providing another layer. Consequently, lots of Internet miscreants and cybercriminals incorporate this technology into their botnets to hide their service infrastructures, extend their lifetime and avoid being tracked down.

B. RRDNS AND CDNS

Round-robin DNS is implemented by responding to a client's request with a list of A records instead of a single A record. The DNS server cycles through this list, so that a series of requests to RRDNS are directed to multiple servers with the aim of providing load balancing, load distribution and fault tolerance. RRDNS is commonly used for portals such as myspace.com [15].

Content delivery networks usually consist of a largely number of nodes scattered in multiple locations around the world. When a user requests a service provided by CDN, it utilize some sophisticated techniques to compute network topology and link characteristic, then the client establishes a connection with the closest nodes which provide the content with high performance here. Similarly, CNDs can also be implemented by returning multiple A records with a low *TTL* to enable them quickly react to some changes of link characteristics. As a result, CDNs have the benefits of load balancing, increased total capacity and high reliability [16] either.

Both CDN and RRDNS are technologies employed by legal commercial organizations. Unfortunately, they usually have been mistaken as FFSNs, since there are some common features exist, such as a list of A records returned in a single DNS request and the low TTL values, which makes detection more difficult. To solve this issue, lots of schemes have been proposed and surveyed in the next section.

C. RELATED WORK

There are two strategies currently has been adopted to detect fast-flux domains. One is active probing approach [7], [8], whereby the domains are extracted from malware domain blacklists or spam emails. For every domain name, lots of repeatedly DNS look ups are performed based on some digging to collect their resolved IPs. Subsequently, extracting the features to identify whether each domain is fast-flux or not. Hu *et al.* [17] used NetFlow information to identify the redirected botnets, which is a specific botnet used to set up redirection flux service not limited to domains collected in spam emails.

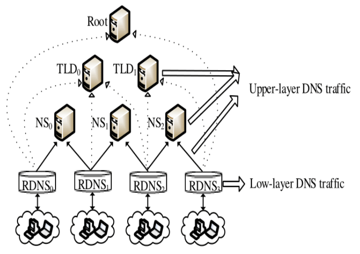


FIGURE 1. DNS traffic classification by coverage.

The other is passive DNS traffic analysis method [9]–[14]. Due to the different coverage of DNS traffic observed during the actual monitoring, we divided the DNS resolution traffic into lower-layer DNS traffic (below local recursive DNS) and upper-layer DNS traffic (above local RDNS servers). The former refers the DNS traffic from single user to their local RDNS servers which contains all the DNS traffic including the traffic in DNS cache. The latter refers to the DNS resolution traffic of a DNS server, as shown in FIGURE 1. Lots of schemes have been proposed by tracking lower-layer DNS traffic, including queries and answers. For example, Perdisci *et al.* [9] use C4.5 decision tree classifier to search for FFSN footprints by collecting DNS queries and answers traffic at various strategic locations in an ISP network. However, the detection delay of this technique is longer because it needs to monitor a domain for 5 days. In order to reduce the detection time to a few seconds, the work in [12]–[14], [18], and [19] proposed real-time detection approach. Lin *et al.* [12] designated the genetic-based real-time detection system to identify FFSNs in real time by employing some features such as the entropy of the domains' preceding nodes of all A records. Almomani [13] seeks to improve the detection accuracy and prediction of the unknown "zero-day" online fast-flux botnet by using a new system called the fast-flux hunter, which supports a new adaptive evolving fuzzy neural network algorithm. Hsu *et al.* [18] proposed a fast-flux bot detection approach with low detection time to detect FFSNs based on network delay features including the delays in fetching documents, the variable network delays and the processing delays. These works can match the basic behavior of FFSNs, however, all of them consider single domain independently ignoring the fact that many flux networks involve more than one domain name, and they will suffer from the issue of low efficiency as few A records returned in a single DNS response, making algorithms [12], [13], [19] less effective. Besides, the DNS traffic collected from a single users to the local recursive DNS, which may cause the problem of privacy violation.

To solve these limitations and achieve the tradeoff between efficiency and latency, a novel detection mechanism is proposed. The work regards flux networks involving more than one domain name, although, Perdisci *et al.* [10] consider a group of domain names as FFSNs, experiments demonstrate that there are many false positives in this algorithm. DNS traffic was collected at some border routers of CERNET backbone which does not contain the traffic in the local cache, therefore it was regarded as the upper-layer DNS traffic from that point. Our approach is similar to hybrid detection [14], however, there are two differences exist. On one hand, our work regards flux networks involving more than one domain name with the motivation of identifying and tracking the behavior of each botnet. On the other hand, some new metrics sets are used, so that, even cybercriminals apply some advanced evasion techniques can still be recognized. Besides, the end users will not worry about their privacies, as the traffic comes form the upper-layer DNS hierarchy, which don't contain any information of DNS cache. Compared to [14], not only does our work can recognized fast-flux botnet with high accuracy in an efficient time, but also it can also capture the regularity and other novel traits of the evolving behavior of each botnet, in addition, experimental results also demonstrate that our algorithms achieve very high detection accuracy and provide considerable improvement compared to other similar reference schemes.

III. THE PROPOSED METHODOLOGY

Our work considers a FFSN contain more than one domain name. At the beginning, groups of DGA-generated domains are recognized by using clustering correlation, then, a double-stages detection mechanism is devised to detect whether the AGD names in a cluster are using fast-flux technology or not. The overview of our system is shown in FIGURE 2, including four modules, namely identifying some specific DGA-generated domains, detecting fast-flux domains by using online detection module, suspicious domains filtering module, and offline monitoring mechanism.

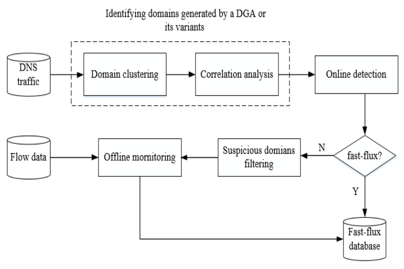


FIGURE 2. **High-level system overview.**

A. IDENTIFYING SPECIFIC DGA-GENERATED DOMAINS

In this section, a sliding window is used to process the AGD names generated based on morpheme features [20]. In our experiment, the size of the window is configured to 30 minutes. After clustering the domains respectively, specific DGA-generated domains are identified by using correlation analysis between the outputs of different cluster algorithms in each window.

Definitions and Notation: A domain name "d" is composed of strings separated by multiple tags (dot). The rightmost part is called the top-level domain (*TLD(d)*) of a domain name, followed by its second-level domain (*2LD(d)*) and so on. For example: www.baidu.com, the top-level domain is "com," the second-level domain is "baidu." Sometimes the top-level domain has more than one label, such as: ". co.uk" is a top-level domain.

1) CLUSTERING BY USING STATISTICAL FEATURES

After analyzing the upper layer DNS resolution traffic, we found that the domains generated by different DGA algorithms are either composed of random English characters or composed of all digits or composed of digits in combination with English characters. Moreover, some advanced malicious codes generate a series of readable domain names (similar to English words) [21]. In this paper, n-gram features, character entropy, digital features and the percentages of the meaningful characters are respectively calculated. Then, these domains are clustered by using X-Means algorithm.

a: N-GRAM FEATURES

Given a domain name, the frequency distribution of n-grams across the domain name strings are measured. In our work, the *2LD (d)* and *3LD (d)* of the domain are considered. Three metrics including the median, average and standard deviation are calculated to obtain the frequency distribution of n-gram values. Six statistical features are obtained as n is set to two or three. Let p represent n-gram, and $|nLD|$ represent the length of *nLD(d)* in formula 1. The extraction process of n-grams is shown in algorithm 1.

$$avg = \frac{\sum_{n-gram\ p\ in\ nLD} count(p)}{|nLD| - n + 1} \tag{1}$$

Algorithm 1 Pseudo Code for Calculating n-Grams

```
Require:
    The domain name: $dname$, $n$, $count$, and normalized English dictionary $Dict$;
Ensure:
    $Sum$, which represents the statistics of n-gram characters in the domain name;
1: Extracting n-gram "str" of length "n" into the container $\mathbf{V}$
2: Constructing a tire tree by using the words in $Dict$
3: For each n-gram "str" in $\mathbf{V}$
4: Traverse the tire tree and statistics the number of occurrences of "str" in the $Dict$ and save it in $count$
5: $Sum += count$
6: return $Sum$
```

b: ENTROPY-BASED FEATURES

In order to distinguish the readability of the domain name, the entropy of the character distribution for separate domain levels are computed. For example, we compute the character entropy of the *2LDs* and *3LDs* respectively. As the range of the character entropy values of different DGA-generated domains is often inconsistent, so it can be used to capture the "level" of randomness.

$$E(nLD) = -\sum_{i=0}^{len(nLD)} pro_i * \log_2(pro_i) \tag{2}$$

Where len($nLDs$) represents the length of the domain name's nLD, and pro_i is the probability corresponding to the ith character in each nLD.

c: DIGITAL FEATURES

A DGA-generated domain names have high similarity and the intuition is that most DGAs produce random-looking strings. Based on this trait, we extract the digital characteristics of the domain name, including the length of *2LD* and *3LD*, the levels of the domain name, namely "n," and the percentage of numeric characters in the *2LD* and *3LD* domains.

d: PERCENTAGE OF MEANINGFUL CHARACTERS (WORDS)

Some advanced DGA algorithms, in order to improve its evasion generate similar artificially domains that are more readable. Based on this phenomenon, we count the significant character ratio *(Pr)* of each *2LD* and *3LD*, where $|r|$ represents the length of nLD, $|wi|$ represents the number of significant characters in every *nLD(d)*, and $|wi| \geqslant 3$.

$$p_r = max(\frac{\sum_{i=1}^{n} |wi|}{|r|}) \tag{3}$$

In order to find clusters of similar domains, we compute the aforementioned statistical features. Then translated each domain into its corresponding feature vector. X-means clustering algorithm [22] is applied to group them into X clusters. X is automatically computed, which is more useful than K-means, as the K need to to be predefined. The output of X-means are represented as $A = \{A_k\}$, where $1 \leq k \leq n$.

2) CLUSTERING BY USING EDIT-DISTANCE

Three different DGA generation schemes are introduced below after the analysis of AGD names obtained from the output of [20] for three months. We apply another similarity standard to group these AGD names based on computing the number of character changes needed to convert from one domain name to another.

(1) Arithmetic-based DGAs, which is the most common DGA type. It always calculates a sequence of values or designates an offset to constitute the alphabet of the DGA.
(2) Wordlist-based DGAs, which is the advanced DGA type that botnets use some well-formed and pronounceable language words. This generation scheme makes these AGDs less randomly appealing and thus more camouflaging.
(3) Permutation-based DGAs, which derive all possible AGDs by permutating the character of an initial domain name.

Based on these generation schemes, we believe that the few number of transformations required to transform one string to another, the more similar between two domain names. Edit distance is used to identify the number of transformations between two domains, which is a symmetric measure, representing as integral value. The type of eligible transformations including addition, deletion and modification. Taken converting the word "cat" to "dog" as an example, the edit distance is three as it requires three times replacement. In order to determine analogous domains, we statistic the edit-distance between two domain names to construct the similarity matrix. The smaller the edit-distance is, the more similar between two domain names. The levenshtein edit distance algorithm [23] is used, as shown in algorithm 2.

Algorithm 2 Pseudo Code of Dynamic Programming Algorithm for Calculating the Edit Distance

```
Require:
    Input: domain name $s1$,$s2$;
Ensure:
    Edit distance $m[i, j]$;
1:  $m[i, j] = 0$
2:  for $i = 1$ to $|s1|$ do
3:      $m[i, 0] = i$
4:  end for
5:  for $j = 1$ to $|s2|$ do
6:      $m[0, j] = j$
7:  end for
8:  for $i = 1$ to $|s1|$ do
9:      for $j = 1$ to $|s2|$ do
10:         if $s1[i] == s2[j]$ then
11:             $tmp = 0$
12:         else
13:             $tmp = 1$
14:         end if
15:         $m[i, j]$ =min($m[[i-1, j-1]$+tmp,$m[i-1, j]+1$,m[i, j -1] + 1 )
16:     end for
17: end for
18: return $m[|s1|, |s2|]$
```

By using the similarity metric defined above, we apply the hierarchical clustering algorithm to group the domains. At the beginning, each domain name is considered as a cluster, then, two nearest clusters are merged at each step. The sequence is a tree-like data structure in which the leaves represent the original domains. In a word, the outputs of hierarchical clustering algorithm is a set $B = \{B_k\}$, where $1 \leq k \leq n$.

3) CLUSTER CORRELATION

Now we have two views of grouping the domain names based on two different definition of similarity, however, neither is perfect, as the produced clusters may still contain noise.

Cluster correlation is performed in order to filter the noise and make the output more likely to be a DGA-generated. Let A_i represent the outputs of X-means and B_j represent the results of hierarchical clustering algorithm, where $1 \leq i, j \leq k$. We calculate the Jaccard index score $\delta = \frac{I_{i,j}}{} U_{i,j}$ between all possible pairs of clusters, where $I_{i,j} = A_i \cap B_j$ and $U_{i,j} = A_i \cup B_j$. If $\delta \geq \theta$, merge them together and consider it as a DGA-generated, then, passed them to the fast-flux botnet detection module described in the next section. In the experiment, the threshold θ is set to 0.75.

B. DETECTING FAST-FLUX BOTNET

A double-stages detection system is proposed to solve some limitations discussed in section 2.3. It includes three modules, namely, identifying fast-flux domains by using online detection approach, suspicious domains filtering module and offline monitoring mechanism.

1) ONLINE DETECTION ALGORITHM

Although, most previous works claimed the ability of detecting fast-flux domains with a high accuracy, the reduction of loner detection time is required, whereas the longer time makes bot herders setting up new malicious domains to deceive legitimate users. ELM is used to identify whether the cluster is fast-flux or not in real time when a batch of domain clusters are obtained. Before talking about ELM, we will introduce some statistical features.

a: TTL FEATURES

TTL records the maximum time (seconds) of the domain name server keeping their resolution information in the DNS cache. The TTL value of a normal domain name is generally 1-5 days [24], while, the fast-flux domain name often has a shorter TTL value in order to switch quickly. Statistical analysis of the TTL values of the normal and the fast-flux domain name in the sample data, results demonstrate that the TTL value greater than 1 day of the normal domains accounting for about 30%, and the value greater than 1200s are more than 60%. However, the TTL value of the fast-flux domain names are small, usually less than 600s that account for about 45%, and more than 70% fast-flux domains with their TTL value less than1000s. Four features are extracted, including the average value of the TTL of the domain name cluster, the standard variance, the number of changes of the TTL and the percentage of the TTL change range. However, some normal network services use techniques of Round-Robin DNS and CDN for load balancing also with relatively low TTL value. Therefore, a single analysis of the TTL value will result in high false positives and false negatives, which needs the combination with other metrics.

b: WHOIS FEATURES

"Whois" is an indispensable information service in the current domain name system, through which you can know the domain name's resolution IP, registrants and other related information. Based on the analysis of experts, the life span of a normal domain name is often long, during this period of time its services can almost be accessed. However, fast-flux domains have a short life span, which facilitates frequent switch and update. Besides, the normal domain names have relatively complete registration information, while, the fast-flux names' registration information is random and incomplete. There are more than 50 "whois" items in a domain name after analyzing the mainly information item and their meaning in the "virustotal" official website. The normal domain names with 30 "whois" information item accounting for 93.28% of the total and most of them actually exists. However, more than 90% of fast-flux with their "whois" item less than 20, and most of their registration information is poorly readable. In addition, the changes of the registration time of domain names are also counted. We found that more than 87% of regular domain names have long registration time (about five years) in which 60% had not been updated. But, about 90% of fast-flux domain names with short registration time (about less than 3 years) in which 65% have expired and have already been updated. Therefore, three metrics including the life span of the domain, the active time of the domain and the completeness of its whois information are extracted.

$$Domain_{life\_span} = Date_{expiration} - Date_{created} \tag{4}$$

$$Domain_{active\_time} = Date_{update} - Date_{created} \tag{5}$$

c: RESOLVED IP FEATURES

In order to keep the availability of their service, attackers use multiple evasion techniques to ensure the survivability of their C&C controllers. The CDN network also maps its domain names to multiple IPs for load balancing. Furthermore, the servers behind the CDN are usually high-performance dedicated servers, which have stable and active IP for a long time, while the servers of FFSN are infected hosts with their IPs constantly online or offline. In order to improve service quality and reduce user access time, the CDNs provide their services based on the principle of proximity. Based on the above characteristics, we analyze the utilized location and attribution information of the resolved IP of the normal and fast-flux domain names. Researchers found that the resolved IP of FFSN domain names is more divergence than the normal domain name. What's more, statistical also shows that the number of *2LDs* the resolved IP mapping to more than 39 in CDN account for 16.8%, while in FFSN the percentage is account for 41.2%. In addition, the number of *2LDs* the resolved IP mapping to more than 10 in CDN account for 24.6%, while in FFSN is 53.5%. With the finds mentioned above, three metrics are calculated, including the entropy of utilized location and attribution of the resolved IP and the number of *2LDs* the resolved IP mapping to.

d: SPATIAL SERVICE RELATIONSHIP FEATURE

The concept of spatial service relationship refers to the service distance between a provider and a consumer, as shown

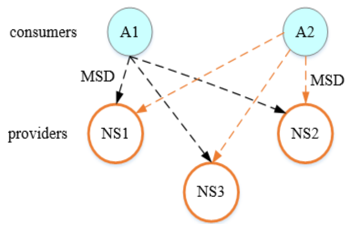


FIGURE 3. Spatial service relationship between provider and client.

in FIGURE 3. The consumer is the IP address in DNS response packet's answer section (Q_{AS}), characterized by Ai in the figure. The providers imply the IPs in additional section (Q_{NS}), such as NSi. For benign domain names, the minimized Service Distances (MSD) for each consumer is almost near to zero. The average and standard deviation of MSD are used to differentiate whether a domain name is benign or not. Let Dist represent the service distance in Euclidean space between mth IP addressin answer section(q_m) and nth IP address in additional section(q_n), a 2 norm distance in Euclidean space shows below.

$$Dist = \sqrt{(C_1(q_m) - C_1(q_n))^2 + (C_2(q_m) - C_2(q_n))^2} \quad (6)$$

For each $q_m \in Q_{AS}$, $\text{MSD}_m = min(\text{Dist}_n)$ where n belongs to Q_{NS}. Let D_{avg} and S_d represent the average and standard deviation of MSD, the calculation method are shown in Equation 7 and 8.

$$D_{avg} = (\sum_{i=1}^{|Q_{AS}|} MSD_i) / |Q_{AS}| \quad (7)$$

$$S_d = \sqrt{\sum_{i=1}^{|Q_{AS}|} (MSD_i - D_{avg})/(|Q_{AS}| - 1)} \quad (8)$$

After extracting of all the features, we use ELM proposed by Huang *et al.* [25] to classify the fast-flux domains. ELM is Single-hidden-Layer Feed forward neural Networks(SLFNs) [26] with fast learning speed and not been widely used in previous research. Although a number of machine learning algorithms have been applied to analyze the domain names, slow learning speed limit the performance for detection. Therefore, in our work, we consider the detection issue as SLFNs, according to [25], a SLFN with L hidden neurons is mathematically denoted as

$$\sum_{j=1}^{l} \beta_j \times g(W_j \times X^{(i)} + b_j) = o_i \quad (9)$$

Where W_j and β_j is the input, output weight of jth hidden node respectively, b_j is the threshold, o_i is the output value of ith neuron, and $g(x)$ is the activation function. Hence the objective of it is to minimize$\| H\beta - T \|$.

$$H = \begin{vmatrix} g(W_1 \times X^{(i)} + b_1) & \dots & g(W_L \times X^{(i)} + b_L) \\ \vdots & \vdots & \vdots \\ g(W_1 \times X^{(n)} + b_1) & \dots & g(W_L \times X^{(n)} + b_L) \end{vmatrix} \quad (10)$$

Where H is the hidden layer, β is the output weight matrix with L dimensions and T is a matrix with N dimensions used to estimate the outputs. The objective function is also equivalent to minimizing the cost function:

$$E = \sum_{i=1}^{n} (\sum_{j=1}^{L} (W_j \times X^{(i)} + b_j) - t^{(i)})^2 \quad (11)$$

After repeatedly adapting variables in each irritation, the cost function will be convergence by using the algorithms like gradient descent or Back Propagation. In addition, ELM regards $\| H\beta - T \| = 0$ as a linear system by randomly choosing W_j and b_j for hidden nodes. Therefore, it can perform the learning process in a relatively fast speed. Compared to C4.5, SVM and BPNN, ELM has a high detect efficiency.

2) SUSPICIOUS DOMAINS FILTERING

After online detection process, the identified fast-flux domains are put in the fast-flux domain database, while the others, which are considered as candidate fast-flux. A combination of a three filter rules is used to remove some benign domains among these candidate fast-flux domains and keep the rest for the offline monitoring module. Before presenting the suspicious domains filtering module, some typical characteristics are discussed, which are used to derive our filtering rules.

(1) short time-to-live;

(2) high frequently changes of their resolved IPs returned by each query;

(3) large set of resolved IPs of a domain name, and the distribution of theirs resolved IPs scattered all over the world.

Based on the characteristics mentioned above, pre-defined heuristic rules are used to perform data volume reduction by discarding clusters that are unlikely to be flux botnet. Given a domain cluster C(d), C(d) is marked as suspicious if all the following rules are matched:(1)$Num(TTL_A \le \mu) \ge \tau$; (2)$Avg\{(N_A \ge 4) \wedge (N_{ASN} \ge 2) \wedge (TTL \le 600)\}$; (3)$Div(R) \ge \theta_{div}$. Where, τ and θ_{div} are suitably chosen thresholds.

Rule 1. Identifies the number of domains in a cluster with TTL equal to μ are larger than τ. Such domains are rare in the normal DNS use, however, fast-flux botnet use it quickly to change their resolved IPs, making it hard to be detected.

Rule 2. Capture FFSNs missed by some stricter real-time detection mechanism.

Rule 3. Capture the diversity of their resolved IPs that is representative as $Div(R) = \frac{|P|}{|R|}$, where P is the number of IPs belonging to different organizations in R. Thus, the larger value $Div(R)$ is, the more scattered of resolved IPs across to different organizations in the world.

Conservatively filtering thresholds are set to make sure that the rules will not discard flux domains ($\theta_{div} = \frac{1}{3}, \mu = 30s$ and $\tau = 20$). As a result, only the domains with very large TTL value, small number of resolved IPs and low value of diversity are considered as nonflux and will be discarded. Although, some attackers configure their codes by returning only one IP after each query often with their TTL value equal to zero, they can't escape by rule 1. In this case, after each click on the same domain, a fresh IP will be obtained. In other words, the output of this module is a list of suspicious flux domains.

3) OFFLINE MONITORING METHOD

Some domain clusters are marked as ''suspicious'' after the process of suspicious domains filtering by using three pre-defined heuristic rules. In order to improve the accuracy and reduce the false negative, some long-term features are used to determine whether or not theses ''suspicious'' are fast-flux.

a: FLUXINESS FEATURES

The size and the diversity of those IPs of all the ''suspicious'' domains are estimated and use fluxiness to determine how fast the domain change their IPs. $\delta_{fluxiness} = \frac{N_{ip}}{N_{single}}$, in which N_{ip} and N_{single} represent the number of distinct IP addresses returned in all previous responses and only one DNS response respectively. In most cases, FFSNs have a larger number of IPs than CNDs, indicating that the value of N_{ip} is bigger than the benign domains. Due to the bot herds of FFSNs don't have complete control over malware-compromised machines, these machines usually online and offline, which makes them very hard to predic the uptime of every flux agent. Therefore, the bot herds must continually add new IPs to keep the availability of their malicious services. Based on the phenomenon, the value of $\delta_{fluxiness}$ are increased accordingly, while CDNs have a relatively fixed IP set and its fluxiness will soon reach to the upper boundary.

b: HISTORICAL INFORMATION RELATED TO THE DOMAIN AND IP

We use the evidence of the network and IP blacklist to describe the historical activities of the domain name. The former is used to describe the network resources assigned to the domain registrant, while the latter depicts the correlation of the domain to some known malware. Antonakakis *et al.* [27] show that Internet criminals often abuse its domain name and IP resources, however, the normal user's network is relatively stable. Therefore, we compute the number of unique ASNs in all A records for single DNS lookup, namely n_{ASN}. Although CDNs are globally distributed, the IP obtained always from the same data center after performing multiple DNS queries. Moreover, the domain name resources are relatively cheap compared to IPv4 resources, so the Internet miscreants often reuse their IP address and BGP prefix. We statistic the number of domains' resolve IP address belongs to BGP prefix (n_{BGP}) in the Spamhaus Block List number [28] to depict the correlation degree of the domain name related to the known malware. The more number indicating the higher correlation, the closer to the fast-flux domain name.

c: THE DEGREE OF DEPENDENCY TO THE DOMAIN

The importance of the domain name to users is representative of the degree of dependency that is obtained from its query behavior of the end users. However, the existence of DNS caching mechanism will block the end user's DNS query request. In other words, the number of user access to the domain ($Access_{num}$) can't directly be obtained from the upper DNS hierarchy traffic. In order to solve this problem, the flow data and the upper DNS hierarchy traffic is used to extract the dependency. As the flow data provides a summary of the communication sessions between the hosts in details, so we can obtain the communication mapping between the user and the server. On the other hand, the upper DNS hierarchy traffic provides the mapping between the domain name and the server. In combination with two different traffic we can get the mapping between the end user and the server. The more the end users are, the large probability of the domain name maybe benign.

A 4-dimensional vector $p = (x_1, x_2, x_3, x_4)$ is constructed as the input of decision function. These elements x_1, x_2, x_3, x_4 correspond to the observed FFSN long-term features $\delta_{fluxiness}$, n_{ASN}, n_{BGP} and $Access_{num}$ respectively. We use linear function to obtain the optimal decision.

$$\begin{aligned} w_1 \times \delta_{fluxiness} + w_2 \times n_{ASN} + w_3 \times n_{BGP} \\ + w_4 \times Access_{num} = \beta \end{aligned} \tag{12}$$

Where w_1, w_2, w_3, w_4 are the elements of the weight vector w, β is a bias term. A linear decision function $f(x) = w^T * x - \beta$ is used to differentiate FFSN domain dames. With the positive value of $f(x)$, FFSN domain names are determined; otherwise, they are benign.

Genetic Algorithm (GA) is applied to determine the best set of weights as it provides robust and efficient search ability with low implementation cost in determining best strategy for many real world problems [29]. Genetic algorithm simulates the problem to be solved into a process of biological evolution. The operations of evaluation, generation and convergence are used to determine next generation. The solution with a low fitness value is gradually eliminated, while the higher fitness function will be increased. After N times of such evolution, it is very likely that an individual with a very high fitness value will be generated. In determining the best set of weights, the GA uses the known benign and FFSN domains for training. Once initialized, GA undergone iterations until the stopping criteria is satisfied. The accuracy rate or the training time can be used as convergence criteria. Experimental results demonstrate that our online in combination detection mechanism provides significantly detection accuracy (99%) and the computation time is reduced by 25% and 21% respectively compared to some existing reference schemes of fast-flux hunter and FluxBuster.

IV. EXPERIMENTAL RESULTS

All experiments in this article were performed on a 2-way Intel Xeon server with one Intel(R) Xeon(R) CPU E5-2650 processor on each path. Each processor contains 8 cores at a frequency of 2.00 GHz, with the memory of 128GB. The algorithm is implemented on C++ and python language. The dataset for conducting the experiments is described in section 4.1, we analyze the accuracy of a DGA-generated domains in section 4.2, then, the performance and the experimental results of the fast-flux detection mechanism is evaluated.

A. THE DATA SET

1) TESTING DATA SET

DNS traffic at upper DNS hierarchy collected from 2017-12-1 to 2018-2-8 at some border routers of jiangsu province of China Education Research Network backbone is used to verify our approach. Eight million domain names were collected and stored in time sequence in order to emulate the real time process. In addition, IP flow record of this article originated from the Network Behavior Observation System (NBOS) is used to obtain the IP mapping between end users and the servers. NBOS is a network traffic behavior monitoring system for monitoring and managing CERNET's service quality and security status [30]. A flow is defined as a unidirectional sequence of packets between a particular source and destination IP pairs, which provides the abstraction information of an IP activity, more details can be found in table 1.

TABLE 1. Attributes of IP flow record in NBOS system.

Attributes	Semantics
srcAddr	the source IP
dstAddr	the destination IP
srcPort	the source port
desPort	the destination port
prot	protocol
firstTime	the start time of the flow
lastTime	the end time of the flow
toa	the applcation type
l_Pkts	the number of packets sent from source to destination
r_Pkts	the number of packets sent from destination to source
l_Bytes	the number of bytes sent from source to destination
r_Bytes	the number of bytes sent from destination to source
Gtime	the during time

2) TRAINING DATA SET

In order to demonstrate the effectiveness of our approach, benign domains and FFSN domains are needed. The benign domain data were collected from Alex's list of the popular top 10,000 websites. We observed once a month for three times continuously to ensure the stability of its rankings. However, the Alex website [31] only provides second-level domain names, in order to obtain complete domain names, we search the domain database to obtain the domain names that have the same "*2LD Tag*" and constitute a legal sample set. FFSN domains were collected from some credible public sources such as ATLAS [32], DNSBL [33], DNSBH [34], and ZeuS Tracker [35], which are supported by information security experts and state-of-the-art detection systems. Then use the same practices as benign domain sets to form FFSN sample set. Table 2 list the information of the different data sources for benign and FFSN domains.

TABLE 2. Information of different data sources.

Dataset	Instances	Collection time	Category
ATLAS	1523	2017.9-2018.5	FFSN
DNSBL	419	2017.1-2018.5	FFSN
DNSBH	3674	2017.1-2018.5	FFSN
ZeuS Tracker	2658	2017.1-2018.5	FFSN
Alexa	10,000	2017.12-2018.5	Benign

B. THE EVALUATION OF IDENTIFYING A DGA-GENERATED DOMAINS

After identifying the domain clusters generated by a DGA or its variants by using correlation analysis, we analyze the composition of different types of AGD domain names in testing data set. The composition of the domain clusters either make up some readable English words, or are randomly generated by using 26 English characters, or composed of digits from one to nine, and some are formed by a mixture of digits and English characters. In addition, there are also some hybrid domains generated by using dynamic DNS that are composed of connectors, digits and characters. FIGURE 4 shows statistics about the percentage of different AGD names. Our algorithm and Phoenix [5] are compared by using the metrics of accuracy and false negative after statistics the percentage of different AGD names. We manually analyze the precision of the AGD names in each cluster set. Due to the low efficiency of manual analysis approach, in order to verify the our approach, we randomly select 300,000 domain names each time. We conduct five experiments and compute the average value as the final results. For example, most of the domains in $cluster_1$ are composed of random characters and digits such as: blsh.sf123.com, cdn20.org, tl88.net. But, the domains composed of either random characters or readability English words in it are regarded as false positives such as: 0731311430.com and akamaiedge.net. Besides, the domains composed of both characters and digits in other

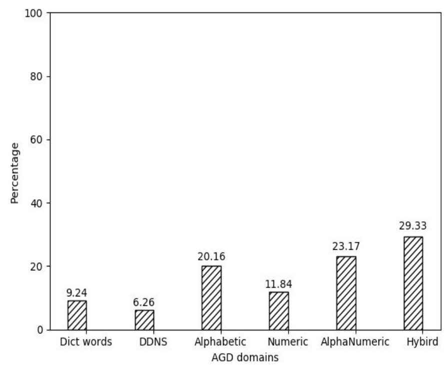


FIGURE 4. The percentage of different AGD names account for in testing data set.

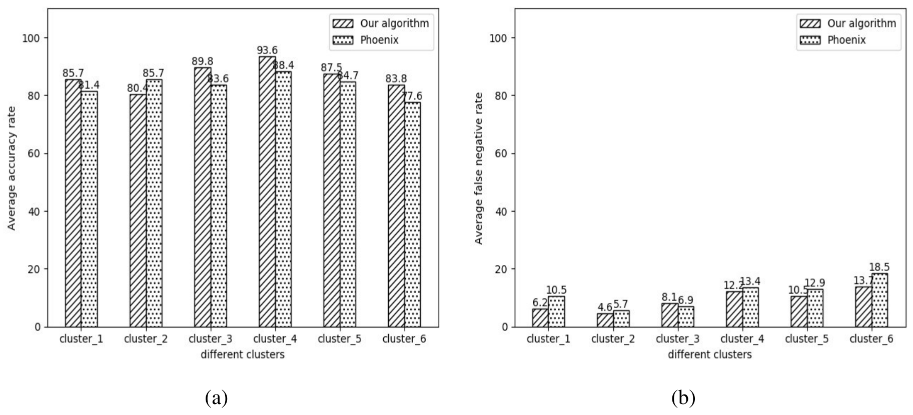


FIGURE 5. The comparison between two algorithms. (a) the accuracy rate. (b) the false negative rate.

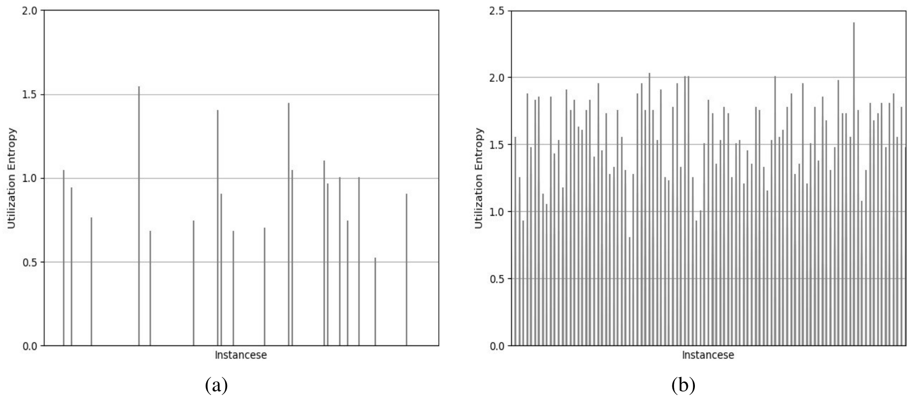


FIGURE 6. The experimental results of the entropy of utilized location. (a) benign domain names. (b) fast-flux domain names.

clusters are regarded as false negative. The statistics of the accuracy and the false negative rate of the two algorithms are shown in FIGURE 5. We can see that the accuracy of our algorithm has been improved by 3.23% and the false negative rate has been reduced by 2.1% on average. Therefore, our algorithm is more comprehensive and accurate in recognizing specific DGA-generated domains.

C. PERFORMANCE EVALUATION OF FFSN DETECTION MECHANISM

In this section, the beneficial of features in the online detection approach are evaluated firstly. Then, we verify the efficiency of our algorithm and perform some comparisons of our detection approach with other researches, finally the live analysis results are given.

1) PERFORMANCE EVALUATION OF THE FEATURES IN ONLINE DETECTION STAGE

FIGURE 6 and FIGURE 7 demonstrate the effectiveness of the new FFSN detection characteristics in online detection stage. It can be noted that the utilized location entropy and the standard deviation of the spatial service relationship of FFSNs is higher than that of those benign domain names. In other words, the geographic distribution of FFSN is more uniform than benign, as the machines in FFSNs usually consist of hijacked hosts, which has a large and randomly distribution all over the world. On the contrary, benign domains that use RRDNS or CDN techniques for load balancing with a lower standard deviation since the machines are always located at the same geographic area and the service provider by CDNs are always giving the nearest or best linked servers to the client. Therefore, the features of the utilized location entropy and the standard deviation of the spatial service relationship are more effective for detecting fast-flux domains.

Four types of features are extracted including the *TTL*, whois, the entropy of utilized location and attribution of their resolved IPs and the domain's spatial service relationship that are reprehensive of F_1, F_2, F_3, F_4 respectively. A 12-dimensional feature vector is constructed to identify fast-flux domain names. In order to achieve better results, we use the 10-fold cross-validation method to obtain the training model. The accuracy and false negative of domain names under different feature combination is evaluated as

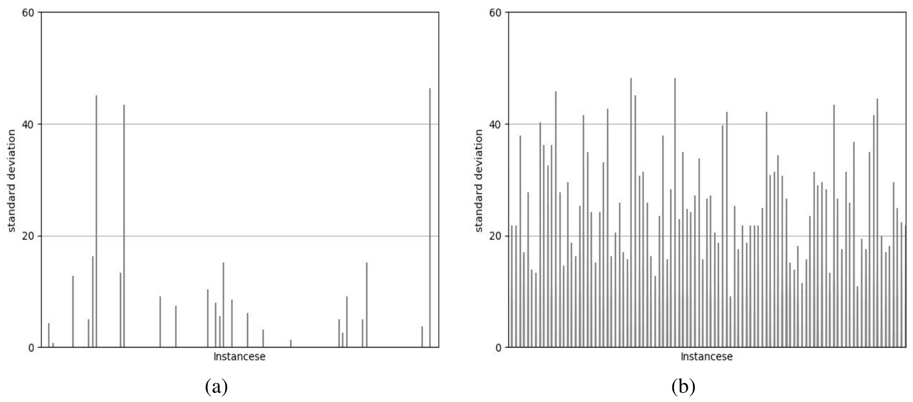


FIGURE 7. The standard deviation of the spatial service relationship. (a) benign domain names. (b) fast-flux domain names.

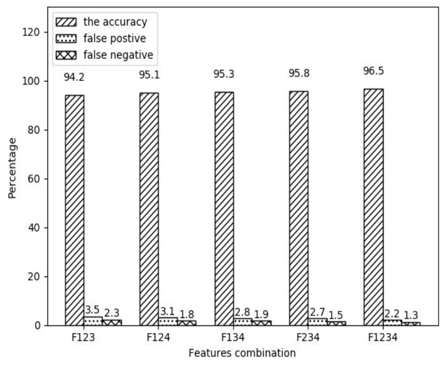


FIGURE 8. The evaluation of different feature combination.

shown in FIGURE 8. Experimental results demonstrate that the combination of four groups of features (F_1, F_2, F_3, F_4) is better than the other combinations.

2) EXPERIMENTAL RESULTS AND COMPARISONS WITH OTHER RESEARCHES

As shown in table 3, the efficiency of ELM algorithm in the online detection stage is demonstrated by using the metrics of time consumption, accuracy, false negative, and etc... In ELM, the training and testing time grow with the increase of the number of hidden nodes, while the accuracy, false negative and false positive remain high stably. Based on this phenomenon, the number of neurons is chosen in order to achieve the tradeoff between the learning speed and accuracy. In the experiment, we set the number of hidden nodes to 300 in ELM as the detection precision does not change much. After the comparison with other machine learning algorithms, ELM is chosen because it has a faster detection speed, then, we compare our double stages detection mechanism with other detection algorithm. Two online detection algorithms GRADE [12] and Fast-flux hunter [13] are chosen in order to evaluate the detection time and accuracy among these algorithms. And an offline detection algorithm FluxBuster [10] is selected, because FluxBuster believes many flux networks involve more than one domain name. So we evaluate the performance when a group of domains are considered as detection object. Besides, a hybrid detection approach [14] is also included with the aim of demonstrating the efficiency of our approach proposed in the paper. The evaluation results are shown in FIGURE 9.

TABLE 3. Comparisons of different classifier.

Classifier	Time consumption(s)	Accuract	False positive	False negative
SVM	65.5	95.6	2.7	1.7
BPNN	53.2	94.8	3.4	1.8
ELM	32.7	96.5	2.2	1.3
C4.5	41.5	96.7	2.1	1.2

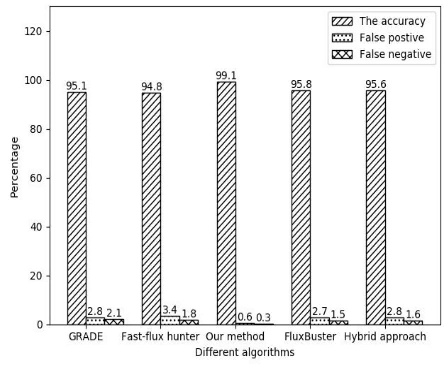


FIGURE 9. The comparison of different algorithms.

As shown in table 4, the memory overhead of this algorithm is a little larger than the other two online algorithms (GRADE, Fast-flux hunter) and smaller than FluxBuster, in addition, it is closer to hybrid detection approach. The reason is that our algorithm regards a group of domains as detection object, which requires more computation and memory resource. However, the detection accuracy of our

TABLE 4. The performance comparisons among algorithms.

Algorithms	On/off-line	Time used(s)	Memory used(MB)
GRADE	Yes	30.48	85.34
Fast-flux hunter	Yes	43.87	92.71
Our approach	Yes	32.71(Online),123.34(Offline except waiting time)	105.64
FluxBuster	No	198.45	102.32
Hybrid approach	Yes	31.85(Online),122.62(Offline except waiting time)	104.84

proposal is better than the others. Our detection mechanism is similar to hybrid approach, compared to it, the computation and memory consumption is almost the same and the difference of overall performance is not significant. But our work regards a group of domains contained in a flux network, while the hybrid approach regards one domain name as fast-flux. When the same process of AGD names identification operates on it, the time and memory consumption increased to 4.7% and 3.4% respectively, which is a little large than ours. Furthermore, our algorithm has a higher detection accuracy than the other's as shown in FIGURE 9, which fully shows that although the algorithm can't completely replace other detection algorithms, it can be used as a supplementary to other detection mechanism to some extent. Therefore, the approach proposed in the paper has the ability to detect fast-flux domains quickly at a high precision.

3) COMPLEXITY ANALYSIS

Time complexity of the proposed detection mechanism is the estimation of the amount of computation required to perform the task. In the process of fast-flux botnet detection, the running time of the program is mainly depend on the time of process the domain name and the time of extracting information from database. Let N, W, D and M represent the number of test domain names, the number of domain names in training set, max domain name size and the number of records in IPCIS database respectively. We class the features into three group, such as domain name-based features, spatial-based features and network-based features. By using domain name-based features, different cluster algorithm are used with time complexity is $O(ND^2W)$ and $O(N^2D^2)$. By using spatial-based features and network-based features, two operations are evaluated including database lookup, whois processing, with time complexity is $O(NM)$ and $O(N)$. The larger the window, the longer the time it consumes. Therefore, theoretically, the time complexity of the proposed approach is $O(N^2D^2)$. Besides, the storage space overhead is also considered, theoretically, the larger the time window is, the more data can be observed and the larger space is required. In our experiment, with the consideration of the time and memory requirements, the window size is set to 30 minutes and obtain better results.

4) LIVE RESULTS ANALYSIS

We evaluate our detection system in a real-world by using the training model mentioned above. After monitoring for a period of three months, (1,354) domain clusters as flux and (2,631) domain clusters as nonflux were obtained, which included a total of (1,815) *2LDs* and (7,667) *2LDs* respectively. Of the 1,815 fast-flux *2LDs*, the *TLD* of .com registered by fast-flux domains are more than fifty percent, table 5 statistics the percentage of *TLDs*. From the table, we can see that the *TLD* of .cn has little percentage. The reason is that individuals are no longer allowed to register ".cn" domains since December 14, 2009 as the tight control of domain registration in China [36]. Therefore, the percentage of illegal uses for ".cn" domain has dropped rapidly since 2010.

TABLE 5. The percentage of top TLDs used by FFSNs.

TLD	2LD domains	Perc
com	1078	59.39%
ru	348	19.17%
net	279	15.31%
at	23	1.27%
me	20	1.10%
cl	16	0.88%
info	13	0.72%
mobi	10	0.55%
org	8	0.44%
tk	5	0.28%
others	15	0.83%

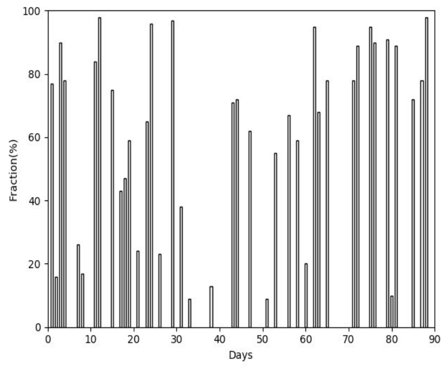


FIGURE 10. Fraction of flux agents detected that reported by known flux agents.

In addition to domain-based evaluation, we also interested in measuring the ability of recognizing the flux agents. We count the fraction of flux agents (resolved IPs) by using the online DNSBL lookup service and spamhaus Block List provided by abuse.ch and spamhaus project respectively. At the end of each epoch (24 hours), 15 flux agents are chosen randomly from the clusters labeled as flux. FIGURE 10 shows the average fraction of flux agents listed as malicious by abuse.ch and spamhaus block List. Moreover, we also evaluate the ability of identifying previously unknown flux *2LDs* as shown in table 6. From table 6 and FIGURE 10 we can see that our double-stages detection mechanism could identify unknown flux *2LDs* and flux agent

TABLE 6. Results obtained during live evalution.

Flux domains	1,815 2LDs(71,243 FQNDs)
Flux agent IPs	356,237 distinct IP addresses
Previously unknown flux 2LDs	782 through a "domain based" analysis, and 967 through a "IP based" analysis
Previously unknown flux agent IPs	39% of flux agents are evaluated against abuse.ch and Spamhaus Block List

IPs, which demonstrates that our approach can be used as a complementary to other detection mechanism in terms of flux domains and their agents further. In a word, our system can detect fast-flux networks in the wild, although, the number of resolved IPs may not indicative of the exact number of flux agents, some mainly due to the effect of DHCP churn [37].

V. DISCUSSION AND FUTURE WORK

In this section, we elaborate on the double-stages detection mechanism and possible evasion techniques.

A. DOUBLE-STAGES FFSN DETECTION MECHANISM

Traditional FFSN detection mechanism rely on spatial and temporal characteristics that has the ability to identify fast-flux. However, the temporal characteristics may result in the latency and false positive of detection, so that the bot herds could set up new malicious domains to deceive legitimate users. In order to achieve the tradeoff between efficiency and latency, a double-stage detection mechanism is proposed, which could alert the users of possible fast-flux domains in a short time with a higher accuracy by the complementary of offline monitoring mechanism. In online detection stage, the target is to achieve lower detection delay, so most of characteristics we choose are related to the information of registrant and the distribution of theirs resolved IP. The characteristics applied in this work are extracted form DNS response packet, which has a higher speed to detect FFSN domains. Although, distinct ASN numbers, distinct organizations and other features have already proposed in [7], these features may be outdated due to some evasion countermeasures. For example, the research shows that several FFSN features such as unique ASNs, the number of rDNS lookups containing ''bad words'' are indicative of compromised home computers, but the number of unique IPs may have become obsolete due to mimicry attacks [36]. Therefore, in the offline stage, by continuous monitoring these outdated and other new characteristics, a considerable improvement of detecting precision could be achieved. In other words, this doesn't mean, though, that our mechanism should replace other real-time fast-flux detection algorithms, instead it could be used as a complementary to other detection schemes.

B. POSSIBLE EVASION TECHNIQUES

1) NOISES CONTAINED IN RESOLVED IPS

Although, Knysz *et al.* [36], discussed some potential evasion techniques against flux detection, most mainly focused on evading active-probing-based systems and may not successfully thwart our detection mechanism. One possible way for evading is that an adversary configures their flux domains mapping to the resolved to IP sets containing both a number of flux agent IPs as well as some random legitimate IPs. This evasion strategy could potentially affect the accuracy of the classification process. The reason is that both characteristics in online and offline process are related to the resolved IPs. If this technology is introduced by the adversary in the DNS answer and additional sections, a large number of victims may be redirected to some legitimate IPs. Thus, the distribution of malicious content may be reduced to only a few end users. Although this evasion strategies are very sophisticated, the adversaries would also incur a significant cost to setup their flux networks. In this case, we believe that the adversaries don't use such sophisticated evasion strategies, so that the accuracy can't be affected much.

2) SINGLE IP ADDRESS

Bot herders use single IP address with the lower TTL values as evasion strategy, sometimes the TTL value is nearly to zero. Thus, the TTL value in the DNS cache is always expired, making the fast-flux botnet quickly changing their resolved IPs in order to keep the availability of their services. This approach is originally used to evade temporal-based detection mechanisms such as Flux-XOR [7] and Flux-Score [15]. As far as spatial-based approach is concerned, it also has some difficulty in resolving the single IP problem, as it is regarded as a single point in the geographic coordinate system. For this reason, it may affect the effectiveness of the metrics of the entropy of utilized location and attribution of the resolved IPs. To solve this issue, pre-defined heuristic rules were used to discard domain names in some clusters. In addition, using a single IP address can have negatively impact to bot herds, such as a quick recruitment speed to the proxy hosts are required and slow response speed of infected computers make end-user discovering the malware. Therefore, the accuracy can't be affected when they using this evasion technique.

In the future, researches need to continually track the regularity of their fluxiness and other novel knowledge of the evolving behavior of each botnet with the aim of devising and discovering new algorithm or features to combat FFSN evasion strategies.

VI. CONCLUSION

In this paper, a novel system for identifying malicious flux networks within AGD names was proposed. By using the upper DNS hierarchy traffic, specific DGA-generated domains that are representative of the respective botnets are recognized, besides, whether the AGD names in a cluster use fast-flux technology or not are also identified. New metrics called the entropy of utilized location and attribution of the resolved IP, the spatial service relationship and the degree of dependency of the domains to end users are suggested to detect fast-flux domain names, which can shorten detection time and increase the detection accuracy compared with other previous researches. Our algorithm can differentiate FFSNs and CDNs based on the features mentioned above, especially it can obtain high efficiency by using the entropy of utilized location and attribution of the resolved IP, and the spatial service relationship. In addition to that, other new discoveries are also found, including the IP overlap and the regularity of fluxiness. The former occurs more often in double-flux botnet with their A records and NA records are all flux and sometimes their IPs are overlap, the latter refers to access period of the fast-flux domains. After analyzing the access period of the fast-flux domains with one A records, we find that their access period is nearly 32 seconds on average. In a words, not only does our approach can identify fast-flux domains but it can also capture some novel traits of the evolving behavior of each botnet, which is a feature of the work and is also an important difference from others. Experimental results demonstrate that our detection algorithm achieves a high detection accuracy rate (99%) with a low false negative rate (0.3%) and provides considerable improvement compared to some existing researches. Although, our approach requires a slightly more memory than other mechanisms, long-term evaluation shows that it capable of accurately detecting previously some unknown flux networks with significantly detection accuracy.

REFERENCES

[1] K. Alieyan, A. ALmomani, A. Manasrah, and M. M. Kadhum, "A survey of botnet detection based on DNS," *Neural Comput. Appl.*, vol. 28, no. 7, pp. 1541–1558, 2017.

[2] A. Jakalan, G. Jian, and L. ShangDong, "Distributed low-interaction honeypot system to detect botnets," in *Proc. 3rd Int. Conf. Comput. Eng. Technol., (ICCET)*, 2011, p. 8.

[3] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *Proc. USENIX Secur. Symp.*, 2016, pp. 263–278.

[4] T.-S. Wang, H.-T. Lin, W.-T. Cheng, and C.-Y. Chen, "DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis," *Comput. Secur.*, vol. 64, pp. 1–15, Jan. 2017.

[5] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based botnet tracking and intelligence," in *Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA* (Lecture Notes in Computer Science), vol. 8550. Cham, Switzerland: Springer, 2014, pp. 192–211.

[6] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi, "Detecting and monitoring fast-flux service networks," in *Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA* (Lecture Notes in Computer Science), vol. 5137. Berlin, Germany: Springer, 2008, pp. 186–206.

[7] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *Proc. NDSS*, 2008, pp. 1–12.

[8] Z. Wang, M. Qin, M. Chen, and C. Jia, "Hiding fast flux botnet in plain email sight," in in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Niagara Falls, ON, Canada: Springer, Apr. 2018, pp. 182–197.

[9] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting malicious flux service networks through passive analysis of recursive DNS traces," in *Proc. IEEE Annu. Comput. Security Appl. Conf. (ACSAC)*, Dec. 2009, pp. 311–320.

[10] R. Perdisci, I. Corona, and G. Giacinto, "Early detection of malicious flux networks via large-scale passive DNS traffic analysis," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 5, pp. 714–726, Sep./Oct. 2012.

[11] A. Al-Nawasrah, A. Al-Momani, F. Meziane, and M. Alauthman, "Fast flux botnet detection framework using adaptive dynamic evolving spiking neural network algorithm," in *Proc. IEEE 9th Int. Conf. Inf. Commun. Syst. (ICICS)*, Apr. 2018, pp. 7–11.

[12] H.-T. Lin, Y.-Y. Lin, and J.-W. Chiang, "Genetic-based real-time fast-flux service networks detection," *Comput. Netw.*, vol. 57, no. 2, pp. 501–513, 2013.

[13] A. Almomani, "Fast-flux hunter: A system for filtering online fast-flux botnet," *Neural Comput. Appl.*, vol. 29, no. 7, pp. 483–493, 2016.

[14] Z. Futai, Z. Siyu, and R. Weixiong, "Hybrid detection and tracking of fast-flux botnet on domain name system traffic," *China Commun.*, vol. 10, no. 11, pp. 81–94, Nov. 2013.

[15] T. Holz, C. Gorecki, F. Freiling, and K. Rieck, "Detection and mitigation of fast-flux service networks," in *Proc. 15th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2008, pp. 1–42.

[16] J. Lai, Q. Fu, and T. Moors, "Using SDN and NFV to enhance request rerouting in ISP-CDN collaborations," *Comput. Netw.*, vol. 113, pp. 176–187, Feb. 2017.

[17] X. Hu, M. Knysz, and K. G. Shin, "Rb-seeker: Auto-detection of redirection botnets," in *Proc. NDSS*, 2009, pp. 1–17.

[18] C.-H. Hsu, C.-Y. Huang, and K.-T. Chen, "Fast-flux bot detection in real time," in *Recent Advances in Intrusion Detection. RAID* (Lecture Notes in Computer Science), vol. 6307. Berlin, Germany: Springer, 2010, pp. 464–483.

[19] S. Martinez-Bea, S. Castillo-Perez, and J. Garcia-Alfaro, "Real-time malicious fast-flux detection using DNS and bot related features," in *Proc. IEEE 11th Annu. Int. Conf. Privacy, Secur. Trust (PST)*, Jul. 2013, pp. 369–372.

[20] W. W. Zhang, G. Jian, L. Qian, S. D. Liu, and H. U. Xiao-Yan, "A lightweight domain name detection algorithm based on morpheme features," *J. Softw.*, vol. 27, no. 9, pp. 2348–2364, 2016.

[21] S. Schüppen, D. Teubert, P. Herrmann, U. Meyer, and S. Sch, "FANCI: Feature-based automated NXDomain classification and intelligence," in *Proc. 27th USENIX Conf. Secur. Symp.* Berkeley, CA, USA: USENIX Association, 2018, pp. 1165–1181.

[22] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proc. ICML*, vol. 1, 2000, pp. 727–734.

[23] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, Melbourne, VIC, Australia, Nov. 2010, pp. 48–61.

[24] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive DNS analysis service to detect and report malicious domains," *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 4, 2014, Art. no. 14.

[25] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.

[26] Y. Shi, G. Chen, and J. Li, "Malicious domain name detection based on extreme machine learning," in *Neural Processing Letters*. New York, Ny, USA: Springer, 2017, pp. 1–11.

[27] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," in *Proc. USENIX Secur. Symp.*, 2010, p. 18.

[28] T. S. Project. (2017). *SBL-spamhaus DNSBLS*. [Online]. Available: http://www.spamhaus.org/sbl/

[29] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 3–17, Apr. 1997.

[30] W. Zhang, J. Gong, W. Ding, and X. Zhang, "Nbos: A fine-grained network management system," *J. Taiyuan Univ. Technol.*, vol. 43, no. 10, pp. 41–46, 2012.

[31] (2017). *Alexa Web Information Company*. [Online]. Available: http://www.alexa.com/topsites/

[32] ATLAS. (2017). *Active Threat Level Analysis System*. [Online]. Available: http://atlas.arbor.net/summary/fastflux

[33] DNSBL. (2017). *Fighting spam by finding and listing exploitable servers*. [Online]. Available: http://www.us.sorbs.net/

[34] (2017). *DNSBH*. [Online]. Available: http://www.malwaredomains.com/wordpress

[35] (2017). *Zeus Domain Blocklist*. [Online]. Available: http://zeustracker.abuse.ch/

[36] M. Knysz, X. Hu, and K. G. Shin, "Good guys vs. Bot Guise: Mimicry attacks against fast-flux detection systems," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1844–1852.

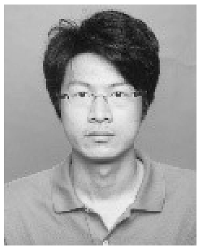
[37] B. Stone-Gross *et al.*, "Your botnet is my botnet: Analysis of a botnet takeover," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, Chicago, IL, USA, Nov. 2009, pp. 635–647.

XIAO-DONG ZANG received the M.Sc. degree in computer science and technology from the Nanjing University of Posts and Telecommunications, China, in 2013. He is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering, Southeast University, Nanjing, China. His research interests include computer networks and security, intrusion detection, and network traffic and host profiling.

JIAN GONG received the B.S. degree in computer software from Nanjing University and the Ph.D. degree in computer science and technology from Southeast University. He is currently a Professor with the School of Cyber Science and Engineering, Southeast University. His research interests are network architecture, network intrusion detection, and network management.

SHAO-HUANG MO is currently pursuing the degree majored in computer science and engineering with Southeast University, China. His research interests include network security, data analysis, and Web application development.

AHMAD JAKALAN received the B.Sc. degree in informatics engineering from Aleppo University, Aleppo, Syria, in 2005, and the M.Sc. and Ph.D. degrees in computer science and technology from Southeast University, China, in 2011 and 2016, respectively. He is currently a Post-Doctoral Researcher with Southeast University. His research interests are computer networks and security, intrusion detection, and network traffic and host profiling.



DE-LIN DING received the M.Sc. degree in computer science and technology from Southeast University, Nanjing, China, in 2015, where he is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering. His research interests include computer network architecture and software-defined network.