Detect Fast-Flux Domain Name with DGA through IP Fluctuation

Hongling Jiang[1] and Jinzhi Lin[2]
(Corresponding author: Hongling Jiang)
School of Information Management, Beijing Information Science and Technology University[1]
No. 12 Xiaoying East Qinghe Road, Haidian District, Beijing, 10092, China
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences[2]
Shenzhen, 518055, China
(Email: hellojhl@163.com)
(Received July 28, 2019; Revised and Accepted Dec. 6, 2019; First Online Apr. 8, 2020)

Abstract

Many malicious networks use the DNS domain names to protect their networks. One of the techniques is the fast-flux, which maps many IP addresses to a domain name and uses recruited hosts to redirect users' requests. Fast-flux is powerful in concealing the malicious networks, thus it is widely used by attackers. Although diverse approaches have been proposed to detect the fast-flux domain names, they still suffer from limitations like either having heavy computations or be easy to be noticed by attackers. According to our research, the IP addresses of the fast-flux domain name are unstable. In this paper, we design a metric called domain score to measure the IP fluctuation. Meanwhile, we consider the feature of the domain name itself. A system called FluDD is proposed to detect the fast-flux domain name with DGA (Domain Generation Algorithm). Experimental results show that FluDD can achieve good performance and the true positive rate reaches to 99.6% and the minimal false positive rate is 0.

Keywords: DGA; DNS; Domain Name; Fast-Flux; IP Fluctuation

1 Introduction

In the early stage, malicious codes usually contained the IP addresses of the C&C servers. Once the IP addresses are detected, the whole malicious network could be shut down. Nowadays DNS (Domain Name System) plays an important role in the Internet [14]. Many internet applications depend on DNS. At the same time, many attacks leverage DNS to be more resilient [29], such as botnet, APT (Advanced Persistent Threat), spam, phishing sites and so on [12, 22, 28]. Attackers use DNS to obtain the IP addresses of their servers. In this way, the attackers can hide their Command and Control (C&C) servers [16].

Recently, many attackers use the so-called fast-flux technique to protect their networks. The fast-flux technique maps a set of IP addresses to a domain name. When a client query a domain name, a set of different IP addresses will be returned. These IP addresses are corresponding to host agents, which redirect the client's requests to the real C&C servers. Ordinary, the IP set includes hundreds and thousands of IP addresses, which can be changed rapidly. Even some of them could be detected as malicious and blocked, many others can still provide services. The fast-flux technique makes it hard to detect the malicious networks. Besides, some attackers use DGA (Domain Generation Algorithm) to generate domain names. The fast-flux combined with DGA makes it more difficult to detect the malicious domain names.

Many solutions have been proposed to detect the fast-flux domain names, but they still face different problems. Existing approaches can be divided into two categories: passive and active. For example, the passive DNS traffic based approaches suffer from heavy computations and privacy concerns [10]. Some of the active approaches may be noticed by attackers as they send requests to the servers regularly. Meanwhile, some approaches would be escaped by attackers [13].

To detect fast-flux with DGA, in this paper, we propose a lightweight approach without causing the attacker's attention. Our paper makes the following contributions.

1) We propose an approach to detect fast-flux with DGA, called FluDD. The approach not only focuses on the fast-flux technique but also pays attention to domain names generated by DGA.

2) We put forward an idea of using IP fluctuation to detect the fast-flux domain names. We propose a metric called domain score to measure the IP fluctuation.

3) Our approach does not need to analysis a large amount of DNS traffic data and the cost of compu-

tation is low. It does not send messages to rival's servers, so it's hard to be found by attackers.

The remainder of this paper is organized as follows. Section 2 introduces the background. Section 3 gives the related work. Section 4 describes our approach in detail. Section 5 shows experimental evaluations and results. Section 6 concludes the paper.

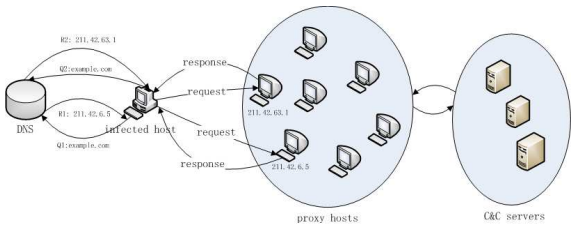

Figure 1: The fast-flux service network

2 Background

2.1 Legitimate Dynamic DNS

Dynamic DNS maps a domain name to a set of IP addresses. Some applications, such as RRDNS (Round-Robin DNS) and CDN (Content Delivery Network), utilize dynamic DNS for various purposes.

RRDNS uses dynamic DNS for load balancing, load distribution and fault tolerance [27]. RRDNS is usually utilized in large networks where the traffic is hard to be managed by a single server. DNS servers are used to distribute traffic to different physical servers. Each time a DNS request is made, one of the IP addresses is returned in a Round Robin fashion. In this way, the traffic will be distributed among the different IP addresses. Round Robin DNS depends on the TTL (Time to Live) values. The smaller the TTL is, the faster these IP addresses are rotated.

CDN utilizes dynamic DNS to serve content to end-users with high availability and high performance [11]. CDN is a globally distributed network consisting of a lot of servers. When an end-user requests the content of CDN, some algorithms are used to choose a server providing the content with high performance. When optimizing for performance, the location may be chosen for serving content. In CDN, small TTL value is required for changing the IP addresses.

2.2 Fast-Flux Service Network

Attackers use the fast-flux service network to organize their compromised hosts, improve their networks availability and hide their service infrastructures. The schematic diagram of the fast-flux service network is shown in Figure 1. In the fast-flux service network, hundreds of IP addresses are mapped to a domain name [3]. When the fast-flux domain name is inquired, different IP addresses are returned, and the IP addresses change frequently. These IP addresses act as agents to redirect the communication between the infected hosts and the C&C servers [19]. If one of the IP addresses is blacklisted, the C&C server can continue to serve through other IP addresses. It's easy to add new C&C servers by adding new IP addresses to the set of IP addresses. This dynamic DNS technique makes it difficult for intrusion detection systems to find the C&C servers hiding behind proxy hosts. To change the IP addresses for a certain fast-flux domain name, the TTL (Time To Live) in the DNS response record is usually small.

2.3 DGA (Domain Generation Algorithm)

Domain Generation Algorithm (DGA) is based on seeds to create a pseudo-random string [7]. DGA can generate a large number of domain names periodically. These domain names can be used to contact the C&C servers [8, 26]. Malware can generate thousands of domain names and contact a few of them every day, which makes them difficult to be eliminated. To be more robust, a malware can use lots of DGAs.

3 Related Work

As many attackers use the fast-flux domain name to protect their networks, it is important to detect fast-flux domain names for preserving network security. Many researchers proposed a lot of fast-flux domain name detection approaches. These approaches can be divided into two categories: Passive and active.

The passive approaches firstly collect DNS traffic including DNS requests and responses and then analyze DNS traffic to recognize the behavior features of the fast-flux domain name.

Ammar [2] presented a fast-flux hunter system to detect the fast-flux service network. The system used an evolving fuzzy neural network algorithm. It collected DNS traffic and analyzed features of the fast-flux service network. The algorithm depends on 14 features, including the number of DNS queries, average packet size, average TTL, the number of TLDs, duration, and so on.

Zhou *et al.* [30] collected DNS traffic of a real campus network. They used Passive DNS to detect the fast-flux domain. Passive DNS constructs 18 domain name features, which are categorized into diversity, time, growth and relevance. They trained a random forest to detect fast-flux domain name.

Leyla *et al.* [15] designed an EXPOSURE system to detect malicious domain names. They extracted 15 features grouped into 4 categories. The feature set includes time-based features, DNS answer based features, TTL based features, and domain name based features. They trained the J48 decision tree algorithm using different combina-

tions of feature sets. The trained classifier is used to detect malicious domain names.

The active approaches query the domain names for their IP addresses. Domain names are obtained from various sources, such as spam, social networks. For each domain name, the detection system queries DNS to get the records of the domain name information. By analyzing the answers, the detection system will judge whether a domain name is benign or malicious.

Hsu *et al.* [10] proposed a fast-flux domain detector (FFDD), which depends on the response time differences. It is based on the observation that the response time of subsequent requests to the same flux bot should be more fluctuating. The FFDD firstly obtains the IP addresses of a domain name, then sends requests to the same client host and measures their response time. The domain name with more fluctuating response time will be judged as the fast-flux domain name.

Davor *et al.* [5] presented a method which measures the network delay, document fetch delay and processing delay of the hosts related to a domain name. The method is based on the observation that the compromised network could have a larger delay than normal one.

Zang *et al.* [27] proposed a fast-flux service network detection scheme which identifies fast-flux botnet with DGA domain names. To detect fast-flux botnet, they measured the features of domain names, such as entropy of location, attribution of the resolved IP, the spatial service relationship. Meanwhile using a machine learning algorithm, the scheme could detect fast-flux service with DGA domain names.

Shi *et al.* [25] proposed a malicious domain name detection approach based on extreme learning machine (ELM). They apply ELM to classify domain names based on multiple features. These features can be divided to four categories, including construction-based, IP-based, TTL-based, and WHOIS-based.

4 Principle and Architecture of FluDD

This section introduces the differences between malicious and benign domain names, and then presents the IP fluctuation of the fast-flux domain name. A new metric called domain score is designed. Besides, this section describes the architecture of FluDD.

4.1 Differences Between Malicious and Benign Domain Names

Attackers use malicious domain names to hide their C&C servers. Malicious domain names use the fast-flux technique, meanwhile, the domain names are generated through DGA. In the fast-flux service network (FFSN), a domain is mapped to a lot of different IP addresses. Each IP address corresponds to a distinct bot. The attackers recruit these bots continually. Each time a client queries the fast-flux domain name, different IP addresses will be returned. In this way, the real malicious servers are hard to be detected.

However, some benign applications also use dynamic DNS techniques, such as RRDNS and CDN. Both of the fast-flux service networks and the benign networks use dynamic DNS, they have similar features, such as small TTL value. However, the difference between the fast-flux and the benign domain name is obvious. In the fast-flux service network, the bots are compromised hosts. The connections between bots and C&C servers are unreliable. To solve this problem, attackers recruit lots of bots and frequently change the mapping between the fast-flux domain names and IP addresses [13]. On the other hand, RRDNS and CDN have their own servers. These servers are used for load balancing. The IP addresses mapping to the benign domain name are stable.

To demonstrate the IP fluctuation of benign and malicious domain names in practice, we select a benign domain name "microsoft.com" and a malicious domain name "2e22e99ot9oofkkkf.000webhostapp.com" randomly from our dataset described in Section 5. We use the "dig" command to obtain the IP addresses from type "A" response of the domain name. Figure 2 and Figure 3 are "dig" command results of "microsoft.com" and "2e22e99ot9oofkkkf.000webhostapp.com", respectively. For each domain name, "dig" command is executed twice. The second "dig" command is executed after the TTL of the response expires. From the results, we can see that the IP addresses of "microsoft.com" are stable, and the IP addresses of "2e22e99ot9oofkkkf.000webhostapp.com" change after TTL expires.

Furthermore, some fast-flux domain names also use DGA to generate domain names. For the fast-flux with DGA domain names, we focus on the features of the domain name itself. Because the domain names are generated automatically and usually not readable, they are different from benign ones in many ways. One of the most obvious features is the length of the domain name. Benign domain names are short to be remembered by users easily, while DGA domain names are not. So the length of DGA domain names usually longer than benign ones. To reduce the computational complexity, we only compute one feature of the domain name, the length.

4.2 IP Fluctuation and Domain Score

The above analysis shows that the IP fluctuation is the main distinction between the fast-flux and the benign domain names. The IP addresses of fast-flux domain names are more fluctuating than benign ones.

Figure 4 illustrates the IP fluctuation of benign domain and fast-flux domain, respectively. The horizontal axises represent the time elapses since the domain name is queried. In the benign network, there are more IP overlaps in different time windows. On the contrary, in the fast-flux service network, there is less or even no IP over-

```
jhl@senslot2:~$ dig microsoft.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> microsoft.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10019
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1024
;; QUESTION SECTION:
;microsoft.com.                 IN      A

;; ANSWER SECTION:
microsoft.com.          1008    IN      A       13.77.161.179
microsoft.com.          1008    IN      A       40.76.4.15
microsoft.com.          1008    IN      A       40.112.72.205
microsoft.com.          1008    IN      A       40.113.200.201
microsoft.com.          1008    IN      A       104.215.148.63

;; Query time: 44 msec
;; SERVER: 125.31.58.114#53(125.31.58.114)
;; WHEN: Sat Jul 20 12:49:59 CST 2019
;; MSG SIZE  rcvd: 122
jhl@senslot2:~$ dig microsoft.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> microsoft.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30968
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1024
;; QUESTION SECTION:
;microsoft.com.                 IN      A

;; ANSWER SECTION:
microsoft.com.          1562    IN      A       13.77.161.179
microsoft.com.          1562    IN      A       40.76.4.15
microsoft.com.          1562    IN      A       40.112.72.205
microsoft.com.          1562    IN      A       40.113.200.201
microsoft.com.          1562    IN      A       104.215.148.63

;; Query time: 45 msec
;; SERVER: 125.31.58.114#53(125.31.58.114)
;; WHEN: Sat Jul 20 15:31:20 CST 2019
;; MSG SIZE  rcvd: 122
```

Figure 2: "dig" command results of "microsoft.com"

```
jhl@senslot2:~$ dig 2e22e99ot9oofkkkf.000webhostapp.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> 2e22e99ot9oofkkkf.000webhostapp.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11967
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1024
;; QUESTION SECTION:
;2e22e99ot9oofkkkf.000webhostapp.com. IN        A

;; ANSWER SECTION:
2e22e99ot9oofkkkf.000webhostapp.com. 3583 IN CNAME us-east-1.route-1.000webhost.awex.io.
us-east-1.route-1.000webhost.awex.io. 57 IN A   145.14.145.196

;; Query time: 47 msec
;; SERVER: 125.31.58.114#53(125.31.58.114)
;; WHEN: Sat Jul 20 15:41:27 CST 2019
;; MSG SIZE  rcvd: 130
jhl@senslot2:~$ dig 2e22e99ot9oofkkkf.000webhostapp.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> 2e22e99ot9oofkkkf.000webhostapp.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25214
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1024
;; QUESTION SECTION:
;2e22e99ot9oofkkkf.000webhostapp.com. IN        A

;; ANSWER SECTION:
2e22e99ot9oofkkkf.000webhostapp.com. 3490 IN CNAME us-east-1.route-1.000webhost.awex.io.
us-east-1.route-1.000webhost.awex.io. 31 IN A   145.14.144.171

;; Query time: 234 msec
;; SERVER: 125.31.58.114#53(125.31.58.114)
;; WHEN: Sat Jul 20 15:43:01 CST 2019
;; MSG SIZE  rcvd: 130
```

Figure 3: "dig" command results of "2e22e99ot9oofkkkf.000webhostapp.com"

lap in different time windows.

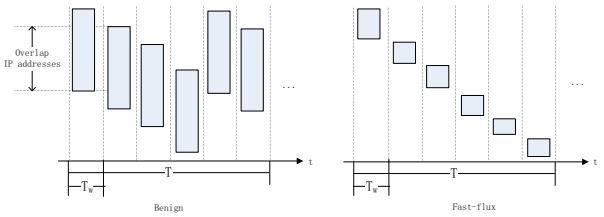

Figure 4: IP fluctuation of benign and fast-flux domain name

To measure the IP fluctuation, we design a metric, called domain score. Assume that during the total time T, a domain name d is queried several times in each time window. Denote T_w as the size of time window. For a time window t_i, the IP addresses set P_i^d is mapped to domain name d. Assume two adjacent time windows, t_i and t_j, the IP addresses sets are P_i^d and P_j^d. The similarity $J(P_i^d, P_j^d)$ between P_i^d and P_j^d is calculated using the Jaccard coefficient [20], shown as Equation (1).

$$J(P_i^d, P_j^d) = \frac{|P_i^d \cap P_j^d|}{|P_i^d \cup P_j^d|} \tag{1}$$

The domain score $S(d)$ is the average $J(P_i^d, P_j^d)$ during T. $S(d)$ is calculated as Equation (2).

$$S(d) = \frac{\sum_{i=1,j=i+1}^{i=L-1,j=L} J(P_i^d, P_j^d)}{L * (L-1)/2} \tag{2}$$

Where L is the number of time windows during T. We use domain score $S(d)$ of a domain name d as a feature to decide whether the domain name d is a benign domain name or a fast-flux one. A domain name with a low domain score is more likely to be a fast-flux domain name.

Moreover, compared to fast-flux domain names, most benign domain names are mapped to less distinct IP addresses during a period of time. Thus, in our system, if a domain name with distinct IP addresses less than 5, we judge it as a benign domain name.

We compute the domain score in two steps. Firstly, we extract the DNS response records periodically. Every T_p time, each domain name is queried for their response records(T_p is less than or equal to T_w). The total time of the DNS querying process is T. In this way, we could obtain the IP addresses mapped to each domain name in different times. It is shown in Algorithm 1. Secondly, we assign a time window ID $twID$ for each record and compute the domain score of each domain name, according to Equation (2). It is shown in Algorithm 2.

Algorithm 1 Extracting DNS response records

Input:

1) *domainList* (the list of domain names)
2) T (the total time of the DNS querying process)
3) T_p (the time interval between two consecutive DNS queries)

Output: *responseList* (DNS response records)
Begin
$starttime$ = current time
$nowtime$ = current time
while $nowtime - starttime \leqslant T$ **do**
  $qtime$ = current time
  for each d in *domainList*: **do**
    $responseList$ = response records of d
    for each r in $responseList$: **do**
      store the $r(d, IP, TTL, queryTime, \dots)$
    end for
  end for
  $nowtime$ = current time
  $qduration = nowtime - qtime$
  sleep$(T_p - qduration)$
  $nowtime$ = current time
end while
End

Algorithm 2 Computing domain score

Input:

1) *domainList* (the list of domain names)
2) $responseList$ ($d, IP, TTL, queryTime, \dots$)
3) T_w (the size of time window)

Output: $S(d)$ of each domain d in *domainList*
Begin
for each d in *domainList* **do**
  sort the $responseList$ of d by $queryTime$
  $twID = 1$
  t_1 = mininum $queryTime$ of all the records
  $t_2 = t_1 + T_w$
  for each r in $responseList$ **do**
    if $queryTime \geqslant t_1$ and $queryTime < t_2$ **then**
      assign $twID$ to r
    else
      update t_1 and t_2
      $twID = twID + 1$
    end if
  end for
  collect all the IP in each time window $twID$
  compute $J(P_i^d, P_j^d)$ between two adjacent time window, t_i and t_j
  compute domain score $S(d)$
end for
End

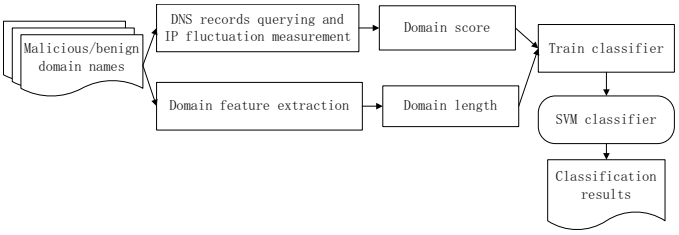

Figure 5: Architecture of FluDD

4.3 FluDD Architecture

Figure 5 gives the architecture of FluDD. Firstly, a set of domain names, which may include malicious and benign domains, are collected. Then, for each domain name, the domain score and length are computed. In order to compute the domain score, a DNS querying process is conducted, as described in Algorithm 1. After that, the IP fluctuation of each domain name is measured, and the domain score is computed according to Equation (2), as described in Algorithm 2. The features of the fast-flux domain names with DGA are quite different from benign domain names as the domain names are generated automatically by algorithm and are not readable. To reduce the cost of calculation, we only extract the length of domain names as a feature. Finally, the SVM (Support Vector Machine) classifier is trained. In the detection phase, we use the trained SVM classifier to detect malicious domain names.

5 Experimental Design and Results

5.1 Datasets

We use two datasets. One is the benign domain names according to Alexa top 500 sites on the web [1]. The other is the malicious domain names collected from sources [9, 17, 23]. After obtaining the malicious domain name datasets, for each domain name, we compute its number of distinct IP addresses. The first 500 malicious domain names with the largest number of different IP addresses are selected.

5.2 Experimental Environment

The proposed approach was implemented in Python3.5.2. The experiments were performed on a server with 4 cores Intel (R) Xeon (R) CPU @ 2.60 GHz. The operating system is 64 bit Ubuntu16.04. The database used is MySQL5.5.55.

5.3 Experimental Settings

1) The parameters for extracting DNS response records.

The total time of the DNS querying process, T in Algorithm 1, is 10 days. The time interval between two DNS queries, T_p in Algorithm 1, is 1 hour.

2) The parameters for computing domain score.

Because the number of distinct IP addresses of the benign domain name is usually small, in our experiments, we filter the domain name with distinct IP addresses less than 5. Thus lots of benign domain names will not be analyzed and the computation is reduced. The size of time window, T_w in Algorithm 2, is a variable parameter. In the following experiments, we first set T_w to 24 hours to see the distribution of features. Then, T_w is set to 1, 6, 12, 18 and 24 hours respectively to evaluate the performance of FluDD.

5.4 Performance Measurement

The performance of FluDD is evaluated using the following metric: True Positive Rate (TPR), False Positive Rate (FPR), Precision (Pr), F-Measure (Fm) [4,18,21]. TPR, FPR, Pr, and Fm are calculated as shown in Equations (3), (4), (5), and (6) respectively.

$$TPR = \frac{TP}{TP+FN} \quad (3)$$

$$FPR = \frac{FP}{FP+TN} \quad (4)$$

$$Pr = \frac{TP}{TP+FP} \quad (5)$$

$$Fm = \frac{2*Pr*TPR}{Pr+TPR} \quad (6)$$

Where TP (True Positives) is the number of malicious domain names recognized as malicious ones correctly, TN (True Negatives) is the number of benign domain names recognized as benign ones correctly, FP (False Positives) is the number of benign domain names recognized as malicious ones incorrectly and FN (False Negatives) is the number of malicious domain names recognized as benign ones incorrectly.

Fm is the weighted average of TPR and Pr. The higher the values of TPR, Pr, and Fm are, the lower the value of FPR is, the better the performance of FluDD is.

5.5 Performance Evaluation

5.5.1 Distributions of Features

In this experiment, we analyzed the distributions of features of domain names, including domain score and length. The size of time window T_w is configured as 24 hours in this experiment.

There are two approaches to obtain the distribution of samples. One is the parametric approach and the other is non-parametric approach [6]. Parametric approaches, such as GMM (Gaussian Mixture Model), LE(Likelihood Estimate), need the pre-defined model and parameter estimation [24], so we use a non-parametric approach in our experiments. The representative method of the non-parametric approach is the Kernel Density Estimation (KDE). KDE use all the sample information to approximate the target probability distribution, shown as Equation (7):

$$f(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h}). \quad (7)$$

Where n is the number of samples. h is the smoothing parameter that controls the size of the neighborhood around x. The larger the value of h, the smoother the probability density function curve, and vice versa [6]. K is the kernel controlling the weight given to the observations x_i at each point x based on their proximity. The kernel function $f(x)$ can make the probability density function by summing all these kernel functions and dividing them by n [24].

Firstly, we analysis the Kernel Density Estimation (KDE) distribution of the single-dimensional feature, domain score and length, respectively. In our experiments, K is a Gaussian kernel function. To gain the probability density curves with different smoothness, we set h to 0.02, 0.08, 0.1 and 0.2 respectively. The KDE distributions of domain score are shown in Figure 6. The x-axises are the domain score, and the y-axises are the density of KDE. The malicious domain scores are much smaller than the benign ones, and the peak domain score is about 0.3. The domains scores of benign domain names are larger, and most of them are more than 0.4. The peak domain score of benign ones is about 0.7. As indicated in previous, the large domain scores mean small IP fluctuation. Compared to malicious domain names, the IP addresses of benign domain names are more stable.

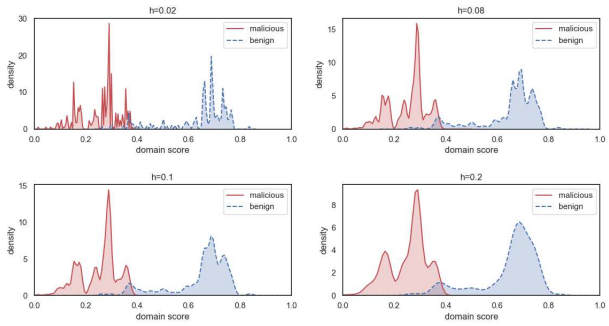

Figure 6: KDE distribution of domain score

The KDE distribution of length is given in Figure 7. The x-axises are the length, and the y-axises are the density of KDE. From Figure 7, we can see that the lengths of benign domain names are less than 20, and most of them are less than 10. However, the lengths of most malicious domain names are much larger than benign ones.

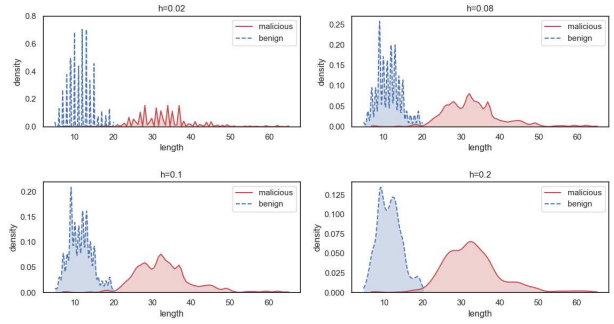

Figure 7: KDE distribution of length

Secondly, we analyze the distribution of both the two features in two-dimensional, as shown in Figure 8. As it can be seen, most of the benign and malicious domain names can be separated by the SVM classifier.

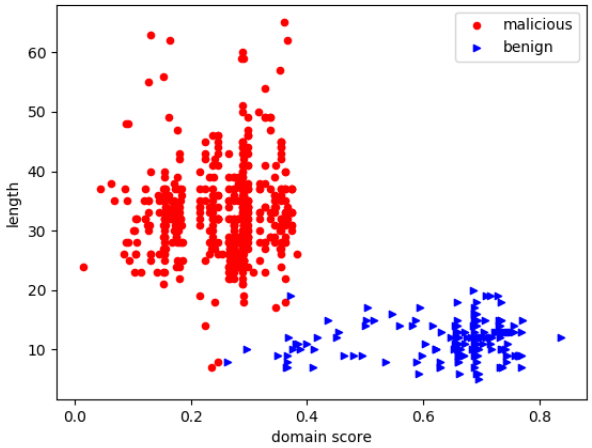

Figure 8: The distribution of two-dimensional features

5.5.2 Performance of FluDD

To achieve the best performance, we try different values of T_w. N-fold cross-validation is used to estimate the performance of the SVM classifier. In N-fold cross-validation, the dataset is partitioned randomly into N samples. The evaluations are run by N times. In each time, $N-1$ samples are selected for training and the remaining samples are used to evaluate the accuracy of the classifier. Finally, the mean value of all the results is calculated. In this experiment, N is 10 and T_w is set to 1, 6, 12, 18 and 24 hours respectively.

We analysis the Pr, Fm, and TPR in different time windows size T_w. As shown in Figure 9, all the Pr and Fm are more than 98%. When the time window size T_w is 6 and 12 hours, the Pr, Fm, and TPR achieve the maximum value, in which Pr is 100%, Fm is 99.8% and TPR is 99.6%.

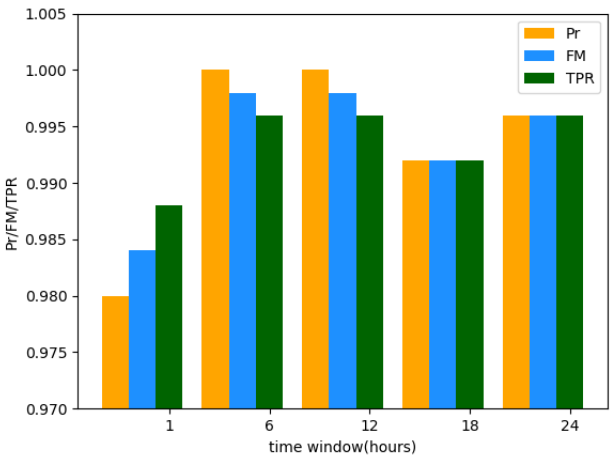

Figure 9: Pr, Fm, and TPR in different time windows sizes

FPR in different time windows size T_w are also analyzed. As shown in Figure 10, when the time window size T_w is 6 or 12 hours, the FPR is zero. From Figure 9 and Figure 10, we can see that when the time window size T_w is 6 and 12 hours, FluDD achieves the best performance.

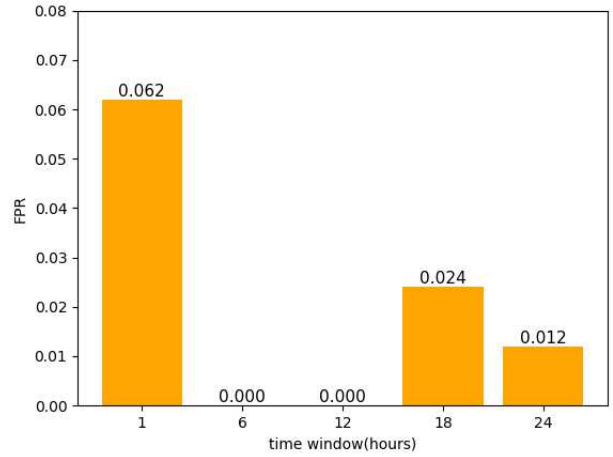

Figure 10: FPR in different time windows size

5.5.3 Discussion and Comparison

Passive fast-flux domain detection approaches need to parse and process the DNS traffic, and this brings heavy computations. While some active approaches send messages or requests to malicious servers and will attract the attention of attackers. Instead, we concern about the stability of IP addresses mapping to a domain name. IP addresses of fast-flux domain names are unstable than benign ones. Our approach utilized this phenomenon.

It is difficult for attackers to bypass the detection of FluDD. A way for attackers to escape the detection of FluDD is to make the domain score large. To do so, a

Table 1: Comparisons of different fast-flux domain detection approaches

Targets	**Hsu *et al.* [10]**	**Zeng *et al.* [27]**	**Shi *et al.* [30]**	**FluDD**
No message sending to malicious servers	No	Yes	Yes	Yes
Can detect domain name generated by DGA	No	Yes	Yes	Yes
No need of domain names generated by the same DGA	Yes	No	Yes	Yes
Fewer features to analysis	Yes	No	No	Yes

lot of stable hosts, which are Internet-connected, without anti-virus software installed and always powered on, are required. Furthermore, attackers could not recruit new hosts to scale up the fast-flux service network during the running time of the detection system. However, all the hosts used in the fast-flux service networks are controlled by dedicated persons, not attackers. It's hard for attackers to ensure most of the hosts are available. To make the fast-flux service network robust, attackers must constantly recruit new hosts to join the network. Thus, it is difficult for the attackers to make the domain score large on purpose.

Three typical related approaches for detecting fast-flux domain names are chosen to make comparisons with FluDD. As shown in Table 1, our approach is different from others and has some excellent features.

6 Conclusion

In this paper, FluDD, a system for detecting the fast-flux domain name with DGA, is proposed. We utilize the phenomenon that the IP addresses mapping to the fast-flux domain name are unstable. A new metric called domain score is designed to measure the IP fluctuation. Meanwhile, to counter the DGA domain name, the length of the domain name is considered. It is convinced that FluDD can be used to detect the fast-flux domain name with DGA. Experiments show that the true positive rate, F-measure and precision of our approach are high, and the false positive rate is low. Our approach is lightweight and requires fewer computations. Furthermore, since no information is sent to the malicious servers, it is impossible for the attacker to notice the detection. Finally, we analyze the advantages of FluDD, the possible evasion methods of attackers and make comparisons of different detection approaches. In the future, we continue to discover the features of the fast-flux domain name and combat attackers' evasion strategies.

7 Acknowledgments

This study was supported by the School Funds of Beijing Information Science and Technology University (No. 1925023).

References

[1] Alexa Internet, *The Top 500 Sites on the Web*, 2019. (`https://www.alexa.com/topsites`)

[2] A. Ammar, "Fast-flux hunter: A system for filtering online fast-flux botnet," *Neural Computing and Applications*, vol. 29, no. 7, pp. 483–493, 2018.

[3] B. Andreas, D. Alessandro, G. Wilfried, and P. Antonio, "Mining agile dns traffic using graph analysis for cybercrime detection," *Computer Networks*, vol. 100, pp. 28–44, 2016.

[4] C. Daiki, Y. Takeshi, A. Mitsuaki, S. Toshiki, M. Tatsuya, and G. Shigeki, "Domainprofiler: Toward accurate and early discovery of domain names abused in future," *International Journal of Information Security*, vol. 17, no. 6, pp. 661–680, 2018.

[5] C. Davor, S. Vlado, and D. Ivica, "Fast-flux botnet detection based on traffic response and search engines credit worthiness," *Tehnički vjesnik*, vol. 25, no. 2, pp. 390–400, 2018.

[6] Q. L. Deng, T. Y. Qiu, F. R. Shen, and J. X. Zhao, "Adaptive online kernel density estimation method (in Chinese)," *Journal of Software*, 2019. (doi: 10.13328/j.cnki.jos.005674)

[7] T. Duc, M. Hieu, T. Van, T. H. Anh, and N. L. Giang, "A LSTM based framework for handling multiclass imbalance in dga botnet detection," *Neurocomputing*, vol. 275, pp. 2401–2413, 2018.

[8] Y. Fu, L. Yu, O. Hambolu, I. Ozcelik, B. Husain, J. X. Sun, K. Sapra, and D. Du, "Stealthy domain generation algorithms," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1430–1443, 2017.

[9] hpHosts, 2019. (`http://hosts-file.net`)

[10] F. Hsu, C. Wang, C. Hsu, C. Tso, L. Chen, and S. Lin, "Detect fast-flux domains through response time differences," *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 1947–1956, Oct. 2014.

[11] L. Jeffrey, F. Qiang, and M. Tim, "Using SDN and NFV to enhance request rerouting in ISP-CDN collaborations," *Computer Networks*, vol. 113, pp. 176–187, 2017.

[12] A. Kamal, A. Ammar, and M. Ahmad, "A survey of botnet detection based on dns," *Neural Computing & Applications*, vol. 28, no. 7, pp. 1541–1558, 2017.

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

[14] J. Kwon, J. Lee, H. Lee, and A. Perrig, "Psybog: A scalable botnet detection method for large-scale dns traffic," *Computer Networks*, vol. 97, pp. 48–73, 2016.

[15] B. Leyla, S. Sevil, B. Davide, K. Engin, and K. Christopher, "Exposure: A passive DNS analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security (TISSEC'14)*, vol. 16, no. 4, p. 14, 2014.

[16] Z. Y. Liu, Y. F. Zeng, P. F. Zhang, J. F. Xue, J. Zhang, and J. T. Liu, "An imbalanced malicious domains detection method based on passive dns traffic analysis," *Security and Communication Networks*, vol. 2018, pp. 1–8, 2018.

[17] Malc0de database, 2019. (`http://malc0de.com/rss/`)

[18] S. Matija, P. J. Myrup, D. Alessandro, and R. Stefan, "A method for identifying compromised clients based on DNS traffic analysis," *International Journal of Information Security*, vol. 16, no. 2, pp. 115–132, 2017.

[19] M. Muhammad, N. Manjinder, and M. Ashraf, "A survey on botnet architectures, detection and defences.," *International Journal of Network Security*, vol. 17, no. 3, pp. 264–281, 2015.

[20] N. Natrajan and P. Suresh, "A comparative scrutinization on diversified needle bandanna segmentation methodologies," *International Journal of Electronics and Information Engineering*, vol. 10, no. 2, pp. 65–75, 2019.

[21] W. N. Niu, X. S. Zhang, G. W. Yang, J. N. Zhu, and Z. W. Ren, "Identifying APT malware domain based on mobile DNS logging," *Mathematical Problems in Engineering*, vol. 2017, pp. 9, 2017.

[22] O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, and K. Bobrovnikova, "Anti-evasion technique for the botnets detection based on the passive DNS monitoring and active DNS probing," in *The 23rd International Conference on Computer Networks (CN'16)*, pp. 83–95, June 2016.

[23] Risk Analytics, *Malware Domain Blocklist by Riskanalytics*, 2019. (`http://www.malwaredomains.com`) 2019.

[24] S. Sanghyun and K. Juntae, "Efficient weights quantization of convolutional neural networks using kernel density estimation based non-uniform quantizer," *Applied Sciences*, vol. 9, no. 12, pp. 2559, 2019.

[25] Y. Shi, G. Chen, and J. Li, "Malicious domain name detection based on extreme machine learning," *Neural Processing Letters*, vol. 48, no. 3, pp. 1347–1357, 2018.

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

[27] X. Zang, J. Gong, S. Mo, A. Jakalan, and D. Ding, "Identifying fast-flux botnet with AGD names at the upper DNS hierarchy," *IEEE Access*, vol. 6, pp. 69713–69727, 2018.

[28] X. D. Zang, G. Jian, and X. Y. Hu, "Detecting malicious domain names based on AGD," *Journal on Communications*, vol. 39, no. 7, pp. 15–25, 2018.

[29] Y. Zhauniarovich, I. KHALIL, T. Yu, and M. Dacier, "A survey on malicious domains detection through dns data analysis," *ACM Computing Surveys*, vol. 1, no. 1, pp. 1–35, 2018.

[30] C. L. Zhou, K. Chen, X. X. Gong, P. Chen, and H. Ma, "Detection of fast-flux domains based on passive DNS analysis (in chinese)," *Acta Scientiarum Naturalium Universitatis Pekinensis*, vol. 52, no. 3, pp. 396–402, 2016.

Biography

Hong-Ling Jiang received her Ph.D in the Computer Science College of Nankai University, Tianjin, China, in 2013. She is currently a lecturer in the School of Information Management at Beijing Information Science and Technology University, China. Her research interest focuses on Network Security, Artificial Intelligence, and the Internet of Things. She has published more than ten papers in recent years.

Jin-Zhi Lin received the Ph.D. degree in computer application from Nankai University, Tianjin, China, in 2015. Currently, he works as an assistant professor in Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences (CAS), Shenzhen, China. His research interests include cyber physical system, embedded system, internet of things and wireless communication. He has also published several peer-reviewed journal and conference papers in recent years.