LagProber: Detecting DGA-Based Malware by Using Query Time Lag of Non-existent Domains

Xi Luo[1,2(✉)], Liming Wang[1], Zhen Xu[1], and Wei An[1]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{luoxi,wangliming,xuzhen,anwei}@iie.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. Domain Generation Algorithm (DGA) has been outfitted by various malware families to extend the resistance to the blacklist-based techniques. A lot of previous approaches have been developed to detect the DGA-based malware based on the lexical property of the random generated domains. Unfortunately, attackers can adjust their DGAs to produce domains by simulating the character distribution of popular domains or words and thus can evade the detection of these approaches.

In this work, we develop an approach from a novel perspective, i.e., the query time lags of non-existent domains (NXDomain), to mitigate DGA-based malware without the lexical property. The key insight is that, unlike the benign hosts, the hosts infected by the same malware families will query a lot of NXDomains in inherent time lags. We design a system, LagProber, to detect infected hosts by analyzing the distribution of time lags. Our experiment with a week of real world DNS traffic reveals that LagProber is able to detect the infected hosts with low false positive rate.

Keywords: Domain Generation Algorithm · DGA-based malware
Time lag · NXDomain queries

1 Introduction

Domain Generation Algorithm (DGA) has been outfitted by various malware families to extend the resistance to the blacklist-based techniques. Cybercriminals utilize DGAs to produce random domains and select a small subset for actual command and control (C&C) use. The randomly generated and short lived C&C domains render detection approaches that rely on static domain lists ineffective.

As the domains generated by the DGA-based malware consist of random and unreadable character concatenations, a lot of researchers have developed detection techniques, e.g., [8,22,25,28,30–32,34], based on the lexical property.

D. Naccache et al. (Eds.): ICICS 2018, LNCS 11149, pp. 41–56, 2018.
https://doi.org/10.1007/978-3-030-01950-1_3

However, these random domains can also be generated by simulating the readable strings. For example, in [11], the authors present a method to generate domains based on the character distribution of the words in English dictionary and their experiment proves that their method can significantly degrade the lexical property-based detection techniques such as [8,34]. In this case, more intrinsic features should be extracted without the lexical property to detect DGA-based malware.

In this work, we develop an approach from a novel perspective, i.e., the query time lags of NXDomains, to mitigate DGA-based malware without the lexical property. The key insight is that, unlike the benign hosts, the hosts infected by the same malware families will query a lot of non-existent domains (NXDomains) in inherent time lags to find the rendezvous points for C&C connection. Motivated by this peculiarity, we design a system, LagProber, to detect the infected hosts by analyzing the query time lags of NXDomains. LagProber extracts features from the distribution of query time lags, and implements a clustering method to identify the infected hosts. In contrast with the other DGA-based malware detection approaches using the time-based features, e.g., periodicity of C&C connections and change points of NXDomain traffic, our features can be extracted in a shorter period and do not rely on specific time patterns. Moreover, the features extracted from time lags can be used compatibly with the periodicity-, change point- or lexical-based detection.

In summary, our research makes following contributions.

(1) We develop an approach from a novel perspective, i.e., the query time lags of NXDomains, to detect DGA-based malware. Our approach is able to identify the infected hosts without the lexical property which is easy to be obscured by attackers.
(2) We design a system, LagProber, to identify the DGA-based malware by analyzing the query time lags of NXDomains. LagProber implements an unsupervised algorithm and thus can run without prior knowledge; and the key advantage is that it can detect the DGA-based malware without the lexical property.
(3) We evaluate LagProber using a week of real world DNS traffic collected from a large ISP network to show the efficacy. The result illustrates that LagProber can accurately detect the DGA-based malware and has scalable performance.

Organization. The rest of the paper is organized as follows. In Sect. 2, we illustrate the background and motivation. The system design is introduced in Sect. 3 and evaluated in Sect. 4. We discuss the limitations in Sect. 5, and summarize the previous works in Sect. 6. At last, in Sect. 7, we conclude this work.

2 Background and Motivation

In this section, we first introduce the background knowledge of the domain generation algorithm and then illustrate the motivation of our approach.

2.1 Domain Generation Algorithm

DGA is an advanced DNS technique used by sophisticated malware families. The attackers periodically generate thousands of domains, which can be used as rendezvous points for C&C communication. Among these domains, only a small number of them are used as actual C&C domains at a certain moment. The real C&C domains only live for short periods before they are replaced with other domains; thus, if the C&C domains are retained by the responders, the C&C communication will persist. The large number of potential C&C domains complicates taking down the C&C servers.

The generated domains are computed based on a given seed, which can consist of numeric constants, the current date/time, or even Twitter trends. In most cases, the character distribution of random generated domains is distinct with that of the benign domains. One can detect DGA-based malware by identifying the random domains. However, as aforementioned in Sect. 1, the domains can be generated by simulating the English dictionary words [11], which can significantly degrade the detection approaches based on the lexical property.

Table 1. The time lags of different malware families.

#	Family	Time lag
1	PadCrypt	0 s between domains
2	Kraken	0 s between domains
3	Proslikefan	0 s between domains
4	Corebot	0 s between domains
5	Pykspa	0 s between domains
6	DirCrypt	0 s between domains
7	Necurs	0 s between domains
8	Symmi	0 s between domains
9	Ramnit	0 s between domains
10	Ranbyus	500 ms between domains
11	newGOZ	1 s between domains
12	Sisron	3 s interval between domains
13	Shiotob	5 s between domains
14	Qadars v3	20 s after 200 domains
15	Banjori	as long as DNS query for www.google.de succeeds

2.2 Query Time Lag of NXDomains

In this work, we develop our approach from a novel perspective, i.e., the query time lags of NXDomains, to detect DGA-based malware without the lexical property. This is motivated by the fact that the infected hosts don't know the

exact C&C domain and have to query a large amount of generated domains until an available response. Therefore, a sequence of query time lags of the NXDomains can be collected to extract the features for detection.

In Table 1, according to the analysis report in Johannes' blog [5], we summarize the time lags of 15 DGA-based malware families. Nine of them query their domains without waiting, five of them implement invariable intervals and one of them alternatively queries the generated domains and www.google.de. No matter what kind of the time lag the attackers implemented, the hosts infected by the same DGA-based malware will have the same distribution. Particularly, when an infected host query a lot of generated domains, the distribution of time lags is consistent. Motivated by this finding, we design a system, LagProber, to detect DGA-based malware by clustering the hosts with similar distribution of the query time lags of NXDomains. The detail of LagProber will be introduced in the following.

3 System Design

In this section, we present our system, LagProber, to show how it works based solely on the query time lags of NXDomains. It is noticeable that LagProber only analyzes the second level domains (SLDs) and domains served by dynamic DNS such as *ddns.net*. A SLD is a domain directly below a top-level domain (TLD) like *.com* and *.net*, or a ccSLD like *.ac.uk* and *.co.uk*. The dynamic DNS domains analyzed in our work are the same as the ones listed in [18]. The reason for analyzing these two types of domains is that the DGA-based malware families usually map their servers to them. Thus, unless otherwise noted, when we talk about domains or NXDomains we refer to the two types of domains.

3.1 Architecture

The architecture of LagProber is shown in Fig. 1. LagProber takes DNS traffic as input and the *Collector* records the NXDomain queries for analysis. For each host, the *Feature Generator* extracts the features from the distribution of query time lags of NXDomains to generate the vectors. The *Group Analyzer* performs a clustering process on the vectors to gather similar ones and outputs a set of candidate clusters. The *Significance Analyzer* implements a significance detection process to identify if any infected hosts there. In the following, we will introduce the details of the four components, i.e., *Collector*, *Feature Generator*, *Group Analyzer*, *Significance Analyzer*.

We maintain a finite-state machine to manage the whole workflow of our system. The state machine is shown in Fig. 2. LagProber starts from the ***Idle*** state. If there is no host querying more than 10 NXDomains ($n_{nx} < 10$), LagProber keeps waiting. When a host queries more than 10 NXDomain ($n_{nx} > 10$), LagProber comes to the ***Preparing*** state. The *Feature Generator* extracts the vector from the distribution of the time lags. If the waiting time t exceeds 1 h (the time window), LagProber comes to the ***Detecting*** state, or else it comes

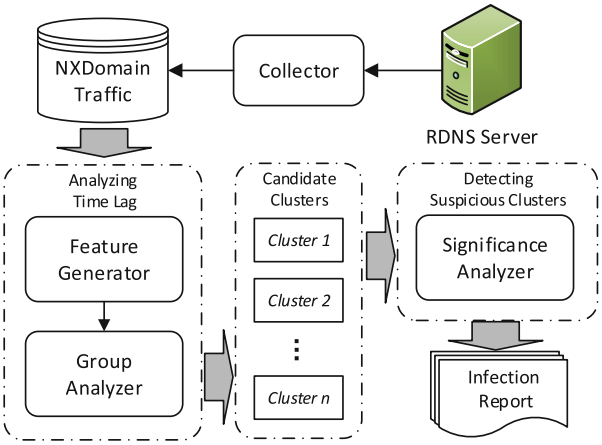

Fig. 1. System Architecture.

back to the ***Waiting*** State. In the ***Detecting*** state, the *Group Analyzer* clusters the vectors and *Significance Analyzer* reports the infected hosts. Then, LagProber resets the system (e.g., $t = 0$ and) and goes back to the ***Waiting*** state.

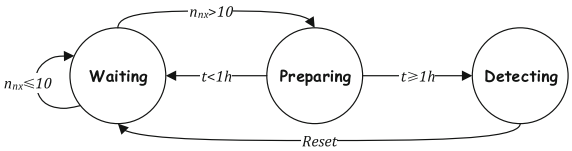

Fig. 2. System Workflow.

3.2 Collector

The *Collector* collects the NXDomain traffic produced in the monitored network and filters out the non-malicious NXDomains to improve the system efficacy. In this work, we consider an NXDomain as non-malicious when it satisfies one of the following conditions.

- Invalid Top Level Domain (TLD): A domain is considered as non-malicious if its TLD is not in the list of registered TLDs presented by IANA [4].
- Irregular characters: A domain contains the characters that should not be included in regular domain (consisting of only letters, numbers and dashes/hyphens). This domain is probably caused by the typing error or misconfiguration.
- Popular domains: We consider the top 100,000 domains in Alexa [1] and websites of world's biggest 500 companies from Forbes [3] as popular legitimate

domains. These NXDomains are mostly utilized by benign services to transfer disposable signals [10].

3.3 Feature Generator

When host h_i queries more than m NXDomains $d_1, d_2, ..., d_m$ (in this work we set $m = 10$ to achieve a fast detection) with timestamps $t_1, t_2, ..., t_m$, LagProber extracts the time lags of the queries, i.e., $S_t = \{l_k : t_{k+1} - t_k\}, k \in [1, m-1]$. We only focus on the distinct domains in a single day due to that repeatedly queried domains have no help to find the rendezvous point for C&C connection.

Although most DGA-based malware families in Table 1 prefer to query the domains with constant time lags, some sophisticated ones can still implement the time lags based on some probability distribution, e.g., Gaussian distribution, to obscure the similarity. Anyway, they have the similar statistic values, e.g., mean and standard deviation, of S_t. Therefore, six statistic values, i.e., the mean, variance, median, maximum, minimum and mode (the most frequent value), of S_t are extracted by this component to construct vector v_i.

3.4 Group Analyzer

The *Group Analyzer* performs a clustering process to gather the similar vectors. Since the number of DGA-based malware families in the monitored network is unsure, LagProber implements a hierarchical merging algorithm, which does not require the number of clusters as input. This algorithm is a clustering approach

Algorithm 1. Clustering Process

Require: $S_v = \{v_i\}, i = 0, 1, ..., p$ containing vectors generated by the Feature Generator Component;
Ensure: $S_c = \{c_k\}, k = 0, 1, ..., q$ containing the outputted clusters.

$C \leftarrow S_v$
while $|C| > 0$ **do**
  $c_i, c_j, d_{ij} \leftarrow getCloestPairOfClusters(S_v)$
  $a_i = mean(c_i) + 2 * std(c_i)$
  $a_j = mean(c_j) + 2 * std(c_j)$
  if $d_{ij} < max\{\sqrt{|v|}, min\{a_i, a_j\}\}$ **then**
    $C \cup merge(c_i, c_j)$
  end if
  if $d_{ij} > a_i$ **then**
    $S_c \cup c_i$
    $remove(C, c_i)$
  end if
  if $d_{ij} > a_j$ **then**
    $S_c \cup c_j$
    $remove(C, c_j)$
  end if
end while

that merges the most similar pairs of clusters as one moves up the hierarchical until the terminated conditions are satisfied. In our work, the Euclidean distance is selected as the similarity measures.

In Algorithm 1, we present the workflow of the clustering process. First, considering each vector generated by the *Feature Generator* component as a single cluster, the function *getCloestPairOfClusters* extracts the closest pair of clusters (c_i and c_j) and returns their distance d_{ij}. Second, LagProber determines whether d_{ij} is too large for c_i and c_j by comparing d_{ij} with the value of $a_i = mean(c_i) + 2 * std(c_i)$, where $mean(c_i)$ and $std(c_i)$ represent the mean and standard deviation of the internal distances of the vectors in c_i, respectively. As shown in Table 1, the time lags of most DGA-based malware families are less than 1 second. The distances of the vectors generated by them are very likely less than $\sqrt{|v|} = \sqrt{\sum_{i=1}^{|v|}(1^2 - 0^2)}$, where $|v|$ is the size of vector. Therefore, if $d_{ij} < max\{\sqrt{|v|}, min\{a_i, a_j\}\}$, LagProber merges c_i and c_j. Third, if $d_{ij} > a_i$ or $d_{ij} > a_j$, which indicates that there is no cluster similar to c_i, LagProber outputs cluster c_i or c_j. At last, if no cluster can be merged, the clustering process is terminated.

3.5 Significance Analyzer

When the infected hosts try to connect the C&C servers, they will query much more NXDomains than the benign ones and the feature vectors will be more similar. As a result, one or more clusters will contain much more items than the other ones. Hence, LagProber implements a outliers testing algorithm, i.e., one-side Grubbs' test [12], also known as the maximum normalized residual test or extreme studentized deviate test, to search for the significantly larger clusters in the result of *Group Analyzer*.

According to Grubbs' test, we define two hypothesis H_1 and H_0 denoting if there is a significantly large cluster or not, respectively. Assuming q clusters $c_1, c_2, ..., c_q$ have been outputted by the *Group Analyzer*, the statistic test is:

$$G = \frac{\max\limits_{i=1,...,q} \{|c_i| - |c|\}}{s}, \tag{1}$$

where $|c_i|$ is the size of cluster c_i, $|c|$ is the mean size and s is the standard deviation of the sizes. The hypothesis H_0 is rejected at a significance level α (set as 0.001 in our work) if

$$G > \frac{q-1}{\sqrt{q}} \sqrt{\frac{t^2_{\frac{\alpha}{2q},q-2}}{q-2+t^2_{\frac{\alpha}{2q},q-2}}}, \tag{2}$$

where $t_{\frac{\alpha}{2q},q-2}$ denotes the upper critical value of the t-distribution [33] with $q-2$ degrees of freedom and a significance level of $\frac{\alpha}{2q}$. In Fig. 3, we present the workflow of the significance test in this component. As the Grubbs' test only examines the maximum, when the hypothesis H_0 has been rejected, LagProber

identifies the corresponding cluster as malicious and removes it for the next round of test. If hypothesis H_0 is accepted, the test process is terminated.

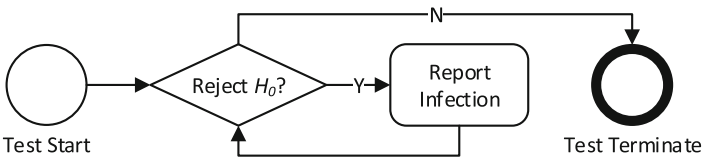

Fig. 3. Workflow of the significance test.

Besides the significantly large clusters, we also consider the clusters containing significantly similar vectors as malicious. Since most of the DGA-based malware families implements 0 s time lag to query the domains, we determine a cluster as malicious when the max distance of the internal vectors is less than $\sqrt{|v|}$. At last, LagProber reports all the hosts generating the vectors in the significantly large or similar clusters as malicious.

4 Evaluation

In this section, LagProber is evaluated on a week of real world DNS traffic. We first introduce the dataset used in our evaluation and analyze the detection result. Then, we present the system performance of LagProber.

4.1 Dataset

Our dataset is collected from an ISP network which offers Internet services to the Chinese education, research, scientific and technical communities, relevant government departments, and hi-tech enterprises. We obtained DNS traffic collected on the edge of this network from May 1st, 2018 to May 7th, 2018. The summary of our dataset is presented in Table 2. Q_{total} and Q_{nx} represent the number of total and NXDomain queries, respectively. Since LagProber only processes the NXDomain traffic (with about 5% of the volume of the total traffic), the exact volume of dataset is significantly reduced. It is noticeable that we rule out the traffic of DNS servers (hosts opening the 53 port for service) in this dataset. This is because LagProber is designed to work under the recursive servers and the behaviors of DNS server are not feasible to represent the human activities.

Labeling. To label the infected hosts in our dataset, we first extracts the response IP addresses mapping to more than 50 distinct SLDs. This is because the C&C IP addresses used by DGA-based malware families are very likely to map with multiple domains. Then, we manually examine the domains with the

Table 2. Summary of our dataset.

Date	Q_{total}	Q_{nx}
2018-05-01	154,548,745	8,352,698
2018-05-02	250,951,430	10,874,362
2018-05-03	226,918,325	10,823,140
2018-05-04	138,368,609	6,085,376
2018-05-05	159,469,521	8,260,969
2018-05-06	154,055,202	8,450,981
2018-05-07	134,670,589	6,940,916
Total	1,218,982,421	59,788,442

help of *Virustotal* [7] and *ThreatCrowd* [6] to ensure the malicious. As a results, we identify 17 infected hosts.

We classify the 17 infected hosts into 3 categories, i.e., *dga*-1, *dga*-2 and *dga*-3, based on the character distribution. The domain samples are presented in Fig. 4. The *dga*-1 domains are constructed by numerics, the *dga*-2 domains are similar with the traditional DGA-based malware families that random select characters, and the *dga*-3 domains are very likely generated by the HASH-based DGA [23]. The *dga*-1 and *dga*-3 infected hosts query domains with no waiting while the *dga*-2 infected hosts consecutively query a few domains per 7 s.

Although the number of hosts is small, the large amount of DNS traffic gives us a good chance to measure the false positive rate, which is very important for real world usage. In the following, these infected hosts are used as ground truth to analyze the result.

dga-1	***dga-2***
2682389127-77.com	uodqbomv.net
2682408961-77.com	uzyfzcier.biz
2682416897-77.com	bcmxxmaso.com
2682398078-77.com	riwaaoifjze.biz
2682417924-77.com	xpmbjqqpv.net
2682417931-77.com	dxplr.cc
dga-3	
sbxrfxowwxzabunktnruoi0jdd.com	
gjneogaqd4enpt2z2usircmyrd.com	
4lf1vuowkzei00ffdoqavnqjzg.com	
3qhdch5s5jvs2sgqqnskhrdmbh.com	
hf5ap34xohz1wospvkqqlnpjqh.com	
osoedr5uhdatiewvzgredvbtzg.com	

Fig. 4. Domain samples queried by the infected hosts.

4.2 Result Analysis

In Table 3, we present the detection result of LagProber. TPr and FPr denote the true and false positive rates, respectively. D_h denotes the ratio between the number of detected infected hosts and the total number N_{inf} of active infected hosts. As one host can generate multiple vectors, the metrics, i.e., TPr and FPr, are calculated based on whether LagProber correctly or falsely classifies a host in a certain time window (1 h). For example, if the vectors generated by an infected host are clustered in a significantly large cluster, we identify a true positive. The true negative, false positive and false negative can be identified similarly. Besides, we also manually examine the classified vectors to ensure the malicious. This is because that the infected hosts can also generate benign queries, and if they fail to connect the C&C servers the positive vectors can not be labeled solely based on the ground truth.

Table 3. Summary of the detection result.

	2018-05-01	2018-05-02	2018-05-03	2018-05-04	2018-05-05	2018-05-06	2018-05-07
TPr	90%	82%	87.5%	90%	100%	100%	88.5%
FPr	3.4%	6.8%	2.9%	0.9%	0.9%	0.6%	0.8%
D_h	100%	100%	100%	100%	100%	100%	100%
N_{inf}	8	9	11	8	5	12	11

As shown in Table 3, the TPr of LagProber is about 90% in average. The false negatives mainly emerges in the case that the number of vectors generated by the infected hosts is too small to construct a significantly large cluster. Anyway, LagProber can detect all the infected hosts (with $D_h = 100\%$).

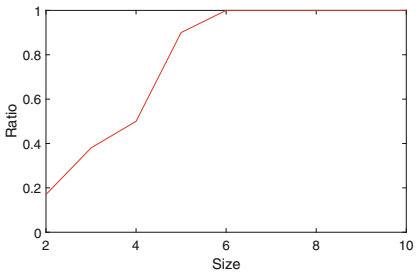

Fig. 5. The relationship between the size and maliciousness of clusters.

The FPr of LagProber is about 2% in average. The false positives are mainly caused by the small clusters being identified as significantly large by Grubbs' test. These small clusters are accidentally by the Network Address Translation

(NAT) routers in the monitored network when they query a sequence of distinct NXDomains in a short period. In Fig. 5, we present the relationship between the size and maliciousness of clusters. The abscissa axis represents the size of clusters and the vertical axis denotes the ratio of malicious ones. When the size of clusters is 5, the ratio of malicious ones is 90% and when the size exceeds 6, all of the clusters are malicious. Hence, to reduce the false positives, one can simply set the threshold as 6 to identify the significantly large clusters.

In conclusion, LagProber can detect all the infected hosts with less false positive rate. This illustrates that the features extracted from the time lags can be utilized to effectively detect the DGA-based malware. Besides, since most hosts in our dataset are the NAT routers, LagProber can achieve a better performance when processing the traffic generated from local or enterprise networks which contain more personal computers.

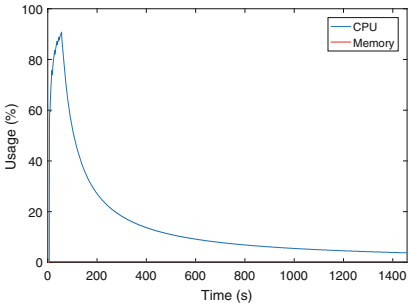

Fig. 6. System performance.

4.3 System Performance

In our experiment, we run LagProber on Ubuntu 12.04 with CPU Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00 GHz and 32 GB memory. As shown in Fig. 6, our system spends about 25 min to analyze the traffics (about 24 GB in bro DNS log format [2]) collected in May 1st, 2018. Since we rule out the repeatedly queried domains in a single day, usage percentage of CPU raises up to 90% at the beginning and then gradually decreases to about 9%. LagProber only stores the vectors generated in an hour so that the memory usage is less and stable (about 0.3%). In summary, when running on the traffic collected from a large scale network, the CPU usage rate is 22.5% in average and the memory usage is about 1 GB. All the results prove that LagProber has scalable performance.

5 Discussion

Limitation. The limitations of this work are as follows. First, the dataset only contains few kinds of DGA-based malware families due to that it is not easy

to find the real world traffic of DGA-based malware families with distinct time lags. However, as we can see in Table 1, the time lag (0 s), of most malware families is the same as *dga*-1 (Sect. 4.1), which indicates LagProber can detect most of them. Second, the attacker can implement long time lags, which is more similar with benign hosts, to evade our detection. In this case, a longer time window can be used for accurate detection. Moreover, they have to spend much more time to connect the C&C servers, and the utility of the infected hosts to attacker is reduced or limited because the attacker can no longer command his bots promptly and reliably.

Comparison. The most generally time-based features to detect DGA-based malware are the periodicity of C&C connection and the change point of the NXDomain traffic. First, for the periodicity-based detection approaches, multiple C&C connections are needed to extract the features while time lags of the NXDomains can be extracted in a single C&C connection. If the infected hosts do not periodically connect the C&C servers, the periodicity-based approaches are ineffective. Second, for change point detection approaches, it is difficult for them to accurately detect DGA-based malware because of that the benign hosts are also very likely to generate the suddenly increased traffics. Hence, the previous works [9,34] should utilize other evidences, like lexical or whois features for accurate detection, while LagProber can achieve a low false positive rate merely based on the features extracted from time lags. Last but not least, the features extracted from time lags can be used compatibly with the periodicity, change point or lexical-based detection. For example, when the periodicity or change point is detected, one can also analyze the time lags to improve the accuracy.

6 Related Work

A wealth of researches have been conducted on detecting DGA-based malware. They mainly utilize the lexical property of the generated domains. Antonakakis et al. [8] introduce a system, Pleiades, to detect DGA bots in large scale network by clustering the NXDomains with the similar character distribution generated by the same DGA-based malware families. Sharifnya et al. [26] present a reputation system to detect DGA botnets by periodically clustering DNS queries with similar characteristics. Schiavoni et al. [25] present a system, Phoenix, to track and fingerprint DGA botnets by clustering domains with similar character distributions. Wang et al. [29] present a system, DBod, to detect DGA botnet by searching for the similar set of NXDomains queried by the bots. Thomas et al. [27] present a method to detect DGA domains by clustering the NXDomains with similar character distributions queried by distinct recursive DNS servers. Yadav et al. [32] introduce three metrics, i.e., K-L divergence, Jaccard Index and Edit distance, to detect DGA domains sharing the same postfix or C&C IP addresses. Yadav et al. improve their work [32] using NXDomains and temporal correlation in [31]. Zhang et al. [34] present a system, BotDigger, to detect a single bot by searching for the suddenly increasing and decreasing random generated SLD

queries before and after C&C connection. Mowbrey et al. [22] present an approach to detect a DGA bot by examining the anomaly domain length distribution in a time slot. Luo et al. [20] and Truong et al. [28] present a set of lexical features to separate a DGA domain from a popular one. Woddbridge et al. [30] and Lison et al. [19] present deep learning methods to predict a DGA domain. While existing solutions demonstrate their effectiveness in detecting malicious servers or server infrastructures, they still can be significantly degraded by generating readable domains [11].

Krishnan et al. [15] implement Threshold Random Walk algorithm [13] to identify an infected host in a fast way without analyzing the lexical property. Their approach relies on the assumption that an infected host is more likely to query a previously unseen NXDomain than a benign host. They have to train the parameters based on at least 24 hours traffic every time when they deploy their approach. Besides, since the malicious samples are not easy to achieve, the probability that an infected host queries a previously unseen NXDomain is pretty difficult to be estimate. Conversely, LagProber detects infected hosts needs no malicious samples.

Except the above DGA-based malware techniques, some botnet or malware domain detection approaches without the lexical can also be utilized to detect DGA domains. Manadhata et al. [21] utilize belief propagation algorithm on graphical models to detect malicious domains. Lee et al. [17] develop a malicious domain detection technique using the sequential correlation property of malicious domains. Khalil et al. [14] and Rahbarinia et al. [24] present methods to infer the suspicious domains which have strong relationship with the known malicious ones. Bilge et al. [9] introduce EXPOSURE to detect malicious domains. They extract 15 features and divide the features into Time-based, DNS answer-based, TTL value-based and Domain name-based. Then, a detection model is trained by using decision tree algorithm. Kwon et al. [16] present PsyBoG to detect botnet by analyzing similar periodicity of the bots. The graph-based approaches [17,21], which need plenty of samples and time to build and process a graph structure, are resource consuming. The time-based approaches [9,16] rely on longer term time patterns, e.g., active time in a month [9] or periodicity of C&C connection. In contrast, LagProber detects DGA-based malware families in a short term mode, i.e., the time lag between two NXDomain queries, and does not rely on the periodicity.

The aforementioned systems are mostly limited by the lexical property, and thus work only on random generated domains. LagProber is a novel general detection system that does not have such limitations and can greatly complement existing detection approaches.

7 Conclusion

In this work, we develop a system, LagProber, to detect DGA-based malware that is independent of the lexical property of the generated domains. Our system exploits a new essential property of DGA-based malware, i.e., hosts infected

by the same malware family will exhibit similar patterns of the query time lags of NXDomains. In our experimental evaluation real-world network traces, LagProber shows excellent detection accuracy with a very low false positive rate on normal traffic.

Acknowledgements. The authors are very grateful to the anonymous reviewers for their valuable comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFB0801900).

References

1. Alexa top 1m. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
2. The Bro network security monitor. https://www.bro.org
3. Forbes biggest companies. http://www.forbes.com
4. IANA top level domains. http://www.iana.org/domains/root/db/
5. Johannes Bader's blog. https://johannesbader.ch/
6. ThreatCrowd. http://threatcrowd.org
7. VirusTotal. http://www.virustotal.com/
8. Antonakakis, M., et al.: From throw-away traffic to bots: detecting the rise of DGA-based malware. In: USENIX Security Symposium, pp. 491–506 (2012)
9. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: EXPOSURE: finding malicious domains using passive DNS analysis. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6–9 February 2011 (2011). http://www.isoc.org/isoc/conferences/ndss/11/pdf/4_2.pdf
10. Chen, Y., Antonakakis, M., Perdisci, R., Nadji, Y., Dagon, D., Lee, W.: DNS noise: measuring the pervasiveness of disposable domains in modern DNS traffic. In: 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, Atlanta, GA, USA, 23–26 June 2014, pp. 598–609 (2014). https://doi.org/10.1109/DSN.2014.61
11. Fu, Y.: Stealthy domain generation algorithms. IEEE Trans. Inf. Forensics Secur. **12**(6), 1430–1443 (2017). https://doi.org/10.1109/TIFS.2017.2668361
12. Grubbs, F.E.: Sample criteria for testing outlying observations. Ann. Math. Stat. **21**, 27–58 (1950)
13. Jung, J., Paxson, V., Berger, A.W., Balakrishnan, H.: Fast portscan detection using sequential hypothesis testing. In: 2004 IEEE Symposium on Security and Privacy (S&P 2004), 9–12 May 2004, Berkeley, CA, USA, pp. 211–225 (2004). https://doi.org/10.1109/SECPRI.2004.1301325
14. Khalil, I., Yu, T., Guan, B.: Discovering malicious domains through passive DNS data graph analysis. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, 30 May–3 June 2016, pp. 663–674 (2016). https://doi.org/10.1145/2897845.2897877
15. Krishnan, S., Taylor, T., Monrose, F., McHugh, J.: Crossing the threshold: detecting network malfeasance via sequential hypothesis testing. In: 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Budapest, Hungary, 24–27 June 2013, pp. 1–12 (2013). https://doi.org/10.1109/DSN.2013.6575364
16. Kwon, J., Lee, J., Lee, H., Perrig, A.: PsyBoG: a scalable botnet detection method for large-scale DNS traffic. Comput. Netw. **97**, 48–73 (2016). https://doi.org/10.1016/j.comnet.2015.12.008

17. Lee, J., Lee, H.: GMAD: graph-based malware activity detection by DNS traffic analysis. Comput. Commun. **49**, 33–47 (2014). https://doi.org/10.1016/j.comcom.2014.04.013
18. Lever, C., Kotzias, P., Balzarotti, D., Caballero, J., Antonakakis, M.: A lustrum of malware network communication: evolution and insights. In: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, 22–26 May 2017, pp. 788–804 (2017). https://doi.org/10.1109/SP.2017.59
19. Lison, P., Mavroeidis, V.: Automatic detection of malware-generated domains with recurrent neural models. CoRR abs/1709.07102 (2017). http://arxiv.org/abs/1709.07102
20. Luo, X., Wang, L., Xu, Z., Yang, J., Sun, M., Wang, J.: DGAsensor: fast detection for DGA-based malwares. In: Proceedings of the 5th International Conference on Communications and Broadband Networking, pp. 47–53. ACM (2017)
21. Manadhata, P.K., Yadav, S., Rao, P., Horne, W.: Detecting malicious domains via graph inference. In: Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, AISec 2014, Scottsdale, AZ, USA, 7 November 2014, pp. 59–60 (2014). https://doi.org/10.1145/2666652.2666659
22. Mowbray, M., Hagen, J.: Finding domain-generation algorithms by looking at length distribution. In: 25th IEEE International Symposium on Software Reliability Engineering Workshops, ISSRE Workshops, Naples, Italy, 3–6 November 2014, pp. 395–400 (2014). https://doi.org/10.1109/ISSREW.2014.20
23. Plohmann, D., Yakdan, K., Klatt, M., Bader, J., Gerhards-Padilla, E.: A comprehensive measurement study of domain generating malware. In: 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, 10–12 August 2016, pp. 263–278 (2016). https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/plohmann
24. Rahbarinia, B., Perdisci, R., Antonakakis, M.: Efficient and accurate behavior-based tracking of malware-control domains in large ISP networks. ACM Trans. Priv. Secur. **19**(2), 4:1–4:31 (2016). https://doi.org/10.1145/2960409
25. Schiavoni, S., Maggi, F., Cavallaro, L., Zanero, S.: Phoenix: DGA-based Botnet tracking and intelligence. In: Proceedings of Detection of Intrusions and Malware, and Vulnerability Assessment - 11th International Conference, DIMVA 2014, Egham, UK, 10–11 July 2014, pp. 192–211 (2014). https://doi.org/10.1007/978-3-319-08509-8_11
26. Sharifnya, R., Abadi, M.: DFBotKiller: domain-flux botnet detection based on the history of group activities and failures in DNS traffic. Digit. Investig. **12**, 15–26 (2015)
27. Thomas, M., Mohaisen, A.: Kindred domains: detecting and clustering Botnet domains using DNS traffic. In: 23rd International World Wide Web Conference, WWW 2014, Seoul, Republic of Korea, 7–11 April 2014, Companion Volume, pp. 707–712 (2014). https://doi.org/10.1145/2567948.2579359
28. Truong, D., Cheng, G.: Detecting domain-flux Botnet based on DNS traffic features in managed network. Secur. Commun. Netw. **9**(14), 2338–2347 (2016). https://doi.org/10.1002/sec.1495
29. Wang, T.S., Lin, H., Cheng, W., Chen, C.: DBod: clustering and detecting DGA-based Botnets using DNS traffic analysis. Comput. Secur. **64**, 1–15 (2017). https://doi.org/10.1016/j.cose.2016.10.001
30. Woodbridge, J., Anderson, H.S., Ahuja, A., Grant, D.: Predicting domain generation algorithms with long short-term memory networks. CoRR abs/1611.00791 (2016). http://arxiv.org/abs/1611.00791

31. Yadav, S., Reddy, A.L.N.: Winning with DNS failures: strategies for faster Botnet detection. In: Security and Privacy in Communication Networks - 7th International ICST Conference, SecureComm 2011, London, UK, 7–9 September 2011, Revised Selected Papers, pp. 446–459 (2011). https://doi.org/10.1007/978-3-642-31909-9_26
32. Yadav, S., Reddy, A.K.K., Reddy, A.L.N., Ranjan, S.: Detecting algorithmically generated malicious domain names. In: Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia, 1–3 November 2010, pp. 48–61 (2010). https://doi.org/10.1145/1879141.1879148
33. Yamane, T.: Statistics: an introductory analysis (1973)
34. Zhang, H., Gharaibeh, M., Thanasoulas, S., Papadopoulos, C.: BotDigger: detecting DGA bots in a single network. In: Traffic Monitoring and Analysis- 8th International Workshop, TMA 2016, Louvain la Neuve, Belgium, 7–8 April 2016 (2016). http://dl.ifip.org/db/conf/tma/tma2016/tma2016-final56.pdf