BotDigger: Detecting DGA Bots in a Single Network

Han Zhang Manaf Gharaibeh Spiros Thanasoulas Christos Papadopoulos
Department of Computer Science
Colorado State University
Fort Collins, Colorado, USA 80521
Email: zhang, gharaibe, spiros.thanasoulas, christos@cs.colostate.edu

***Abstract*—To improve the resiliency of communication between bots and C&C servers, bot masters began utilizing *Domain Generation Algorithms* (DGA) in recent years. Many systems have been introduced to detect DGA-based botnets. However, they suffer from several limitations, such as requiring DNS traffic collected across many networks, the presence of multiple bots from the same botnet, and so forth. These limitations make it very hard to detect individual bots when using traffic collected from a single network. In this paper, we introduce BotDigger, a system that detects DGA-based bots using DNS traffic without a priori knowledge of the domain generation algorithm. BotDigger utilizes a chain of evidence, including quantity, temporal and linguistic evidence to detect an individual bot by only monitoring traffic at the DNS servers of a single network. We evaluate BotDigger's performance using traces from two DGA-based botnets: Kraken and Conflicker. Our results show that BotDigger detects all the Kraken bots and 99.8% of Conficker bots. A one-week DNS trace captured from our university and three traces collected from our research lab are used to evaluate false positives. The results show that the false positive rates are 0.05% and 0.39% for these two groups of background traces, respectively.**

I. Introduction

Cyber security constitutes one of the most serious threats to the current society, costing billions of dollars each year. Botnets is a very important way to perform many attacks. In botnets, the botmaster and bots exchange information through C&C channels, which can be implemented using many protocols, such as IRC, HTTP, OVERNET. Although using P2P protocols as C&C channels is getting popular, HTTP-based botnets are still very common as they are easy to implement and maintain. In a HTTP-based botnet, the botmaster publishes the commands under a domain, then the bots query the domain to fetch the contents periodically. In the early years, the domain was hard-coded in binary, introducing a single point of failure. To become more robust, botnets began to generate C&C domains dynamically on the fly using DGA. In particular, hundreds or thousands of domains can be algorithmically generated every day, but the botmaster only registers one or a few of them as C&C domains and publishes the commands there. DGA technique evades static blacklists, avoids single point of failure, and also prevents security specialists from registering the C&C domain before the botmaster.

There are four reasons to detect DGA botnets using DNS traffic. First, the DGA bots have to send DNS queries to look up the IP addresses of C&C domains. Second, the amount of DNS traffic is much less than the overall traffic. Focusing on a relatively small amount of traffic helps to improve performance, making it possible to detect bots in real time. Third, the DNS traffic of DGA bots has different patterns compared to legitimate hosts. For example, DGA bots send more DNS queries than legitimate hosts. Last, if we can detect bots only using DNS traffic when they look for C&C domains, we can stop the attacks even before they happen.

Many previous works have been introduced to detect DGA-based botnets and malicious domains (e.g., C&C domains, phishing domains) using DNS traffic [8], [20], [9], [6], [22], [23], [17]. They share some common assumptions, such as DGA domains generated by the same algorithm have similar linguistic attributes, DGA domains' attributes are different from legitimate ones, and so forth. Based on these assumptions, classification and/or clustering algorithms are applied for detection. However, many of these past works require multiple hosts infected by the same type of botnet existing in the collected traces. Consequently, they have to collect DNS traffic at an upper level (e.g., TLD servers, authoritative servers, etc), or from multiple RDNS servers among networks. The advantage of these works is that evidences from multiple bots can be collected and analyzed. However, they also introduce several challenges. First, the DNS traffic at an upper level is hard to access for most of enterprise and/or university network operators. Second, sharing DNS traffic among networks may introduce privacy issues. Third, it is computationally expensive to run clustering/classification algorithms on large traces collected from multiple networks. Finally, the most significant challenge is that an enterprise network may not have multiple bots, especially bots infected by the same botnet. For example, Pleiades [8] detects less than 100 bots of the same botnet in a large North American ISP that includes over 2 million clients hosts per day. As a comparison, our university network has around 20,000 clients, which is only 1% of the ISP, meaning that on average only 1 host infected by the same botnet exists in the network.

In this paper, we introduce BotDigger, a system that detects an ***individual bot*** by only using DNS traffic collected from a ***single network***. This single network can be a company network, a university network, or a local area network (LAN). Notice that "detecting individual bot in a network" does not mean BotDigger cannot detect all the bots in a network. If there are multiple bots in the same network, BotDigger can still detect them, but individually. BotDigger uses a chain of evidences, including quantity evidence, linguistic evidence, and temporal evidence to detect bots. In particular, quantity evidence means that the number of suspicious second level domains (2LDs) queried by bots are much more than the legitimate hosts. Two temporal evidences are used: 1) the number of suspicious 2LDs queried by a bot suddenly increases when it starts to look for the registered C&C domain, 2) once the

bot hits the registered C&C domain, the number of queried suspicious 2LDs will decrease. The basis of linguistic evidence is that the DGA NXDomains and C&C domains queried by a bot are generated by the same algorithm, thus they share similar linguistic attributes. We apply the above evidences sequentially to detect bots and the queried DGA NXDomains. After that, we extract signatures from the DGA NXDomains and apply them on the successfully resolved domains to extract the corresponding C&C domain candidates. The contributions of this paper are listed as follows:

1) We introduce a chain of evidences, including quantity evidence, temporal evidence and linguistic evidence, to detect DGA-based botnets.
2) We introduce and implement BotDigger, a system that detects an individual DGA-based bot using DNS traffic collected in a single network.
3) We evaluate BotDigger with two datasets from our university and lab, as well as two DGA-based botnets. The results show that BotDigger detects more than 99.8% of the bots with less than 0.5% false positives.

II. Datasets

A. Botnet Dataset

We use two datasets of DGA botnets in this paper. The first dataset includes 140 Kraken botnet traces obtained from Georgia Tech [12], [19], which we call *140Samples*. Each trace in *140Samples* is collected by running captured bot binaries in an isolated environment. Each traces include DNS queries and responses of DGA domains, binary download, C&C communications and other traffic.

The second dataset includes domains generated by a well known DGA botnet - Conficker. Variant C of Conficker generates 50,000 domains every day [16]. We collected a list of all the 1,500,000 domains generated in April 2009 from [15].

B. Background Dataset

Two groups of background traces are used in our experiments. The first group includes three traces captured at our university lab, which is connected to the outside world via a 10Gb/s link. The lab has a dedicated /24 subnet and runs standard services, such as mail, web, ssh, etc., and also carries traffic for several research projects including two PlanetLab nodes and a GENI rack. All traces are in tcpdump format and include payload. They are named *Lab1*, *Lab2* and *Lab3*. The first two traces are 24-hour traces captured on Feb-22th-2011 and Dec-11th-2012, respectively. *Lab3* is a 72 hour trace captured on Dec-13th to Dec-16th-2012. Table I provides some statistics for these datasets.

The second background trace includes only DNS traffic captured at our university. For all the users connected to the university network, their DNS queries are sent to four recursive DNS servers. We collect all the DNS traffic by mirroring the ports of the four servers for a total of seven days, starting from April 2nd to 8th in 2015. We call this dataset *CSUDNSTrace*. *CSUDNSTrace* includes 1.15E9 domains queried by 20682 university hosts. 5.5E6 of the queried domains are unique. Some statistics for this dataset are listed in Table II.

TABLE I: Lab Traces

	Lab1	Lab2	Lab3
TCP	93.55%	51.87%	72.25%
UDP	1.03%	2.83%	0.258%
ICMP	5.41%	45.14%	27.27%
Others	0.001%	0.149%	0.215%
Bandwidth	14.08 Mb/s	26.16 Mb/s	18.4Mb/s
Peak BW	29.3 Mb/s	133.9 Mb/s	81.9 Mb/s
Duration	24 hours	24 hours	72 hours

TABLE II: CSU DNS Trace

Date	Total Queried Domains	Unique Queried Domains	Hosts in Trace
April 2nd	1.83E8	1.50E6	17942
April 3rd	1.68E8	1.29E6	17599
April 4th	1.48E8	8.51E5	13885
April 5th	1.41E8	9.33E5	13794
April 6th	1.67E8	1.52E6	18083
April 7th	1.75E8	1.55E6	18321
April 8th	1.76E8	1.53E6	18246

III. Methodology

A. Notations

Domain names are organized in a tree-like hierarchical name space. A domain name is a string composed of a set of labels separated by the dot symbol. The rightmost label is called top-level domain (TLD). Two commonly used TLDs are generic TLD (gTLD, e.g., .com) and country code TLD (ccTLD, e.g., .cn). A domain can contain both gTLD and ccTLD, for example, *www.foo.example.com.cn*. In this paper, we consider the consecutive gTLD and ccTLD as a single TLD for simplicity. We define the domain that is directly to the left of the TLDs as *Second Level Domain* (2LD), and define the domain to the left of the 2LD as *Third Level Domain* (3LD). The 2LD and 3LD in the above example domain is "example" and "foo", respectively. Note that in this work we denote a 2LD of a NXDomain as *2LDNX*.

B. System Overview

An overview of the methodology is shown in Figure 1. First, several filters (Section III-D) are applied to remove unsuspicious NXDomains (e.g., the domains with invalid TLDs). The remaining suspicious NXDomains are then grouped by host who sends the queries. Notice that in the following steps, we focus on the queried domains from each *individual* host, and that is the reason why our method can detect individual bot. Now the *quantity* evidence (Section III-E) is applied to extract outliers in terms of the number of queried suspicious 2LDNXs. For each outlier, we use the *temporal* evidence (Section III-F) to extract the period of time when a bot begins to query DGA domains until it hits the registered C&C domain, denoted as (t_{begin}, t_{end}). If such period cannot be extracted, then the host is considered as legitimate, otherwise the host is considered suspicious and the following analysis

is performed. The next step focuses on the suspicious NXDomains being queried between t_{begin} and t_{end}. The linguistic attributes of these NXDomains are extracted and then the *linguistic* evidence (Section III-G) is applied on the extracted attributes. The assumption of linguistic evidence is that a bot queries many suspicious NXDomains that have similar linguistic attributes thus they will likely be clustered together. In particular, a hierarchical clustering algorithm is applied on the attributes. The output is one or more clusters of linguistic attributes and the corresponding suspicious NXDomains. We consider the clusters whose sizes are greater than a threshold (named *BotClusterThreshold* as defined in Section III-G3) as bot NXDomain cluster candidates. If all the clusters of a host are smaller than the threshold, then the host is considered legitimate. Finally, to identify the C&C domains, the DGA domain signatures are extracted from the bot NXDomain cluster candidates and matched against all the successfully resolved domains queried between t_{begin} and t_{end}. The domains that match the signatures are considered as C&C domain candidates, and the host is labeled as a bot. The host is not labeled if no C&C domain candidate can be extracted. Note that we do not precisely label C&C domain. Instead, we label multiple C&C domain candidates. It is possible, however, unlikely, that a successful request is done by the infected host right after a series of failed requests, and the legitimate domain in the request is close in lexicographic distance. For this reason we can never be absolutely certain that a successful DNS request is a C&C server. Precisely labeling single C&C domain is future work.

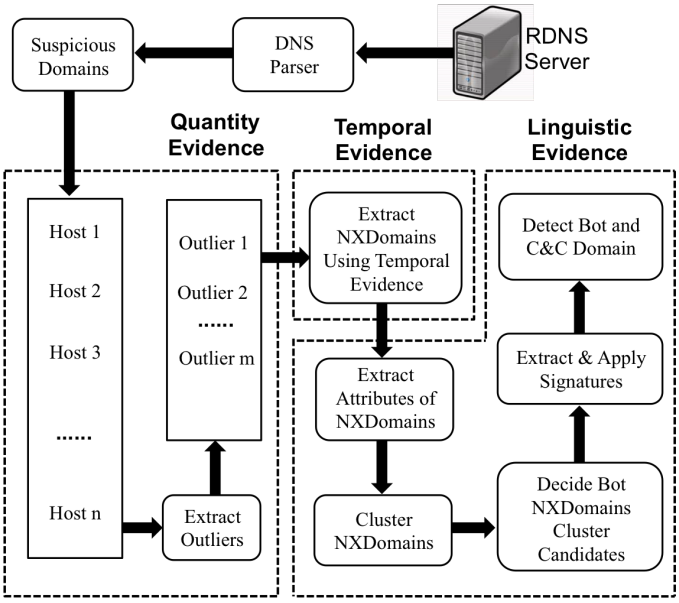

Fig. 1: System Overview

C. Ethical Considerations

BotDigger does not require the IP address of the host making DNS requests, but only the domain request itself. This decouples the address of the host from the actual request; if there are many hosts in the network then it is hard to associate the request with the host. BotDigger still needs to report bots when detected, but this can be done using an opaque identifier for the host IP address, not the IP address itself. The identifier may be a mapping known only to the network operator and never revealed to BotDigger.

We acknowledge that in networks with a few hosts or in cases where the domain request itself contains host-specific information, privacy may still be compromised. However, we envision that BotDigger will be used the same way network operators use IDSs such as Snort and Suricata. Such IDS' require full access to packet headers and payload, and their use is justified as long as operators use them for network operations and security.

D. Filters

Previous work shows that many of the failed domains are non-malicious [13], [14]. Jiang et al. categorize failed DNS queries into seven groups in [13]. In [14], the authors classify NXDomains into nine groups. Based on their classifications, we build five filters to remove non-suspicious NXDomains.

1) Overloaded DNS: Besides fetching the IP address of a domain, DNS queries are also "overloaded" to enable anti-spam and anti-virus techniques [18]. We collect 64 websites that provide blacklist services to filter overloaded DNS.
2) Invalid TLD: We obtain the list of all the registered TLD from IANA [3]. A domain is considered as unsuspicious if its TLD is not registered.
3) Typo of popular domains: We consider the top 1,000 domains in Alexa [1] and websites of world's biggest 500 companies from Forbes [2] as popular legitimate domains. If the *Levenshtein distance* between a given domain and any of these popular domains is less than a threshold, then the domain is considered as a typo.
4) Excluded words: These domains contain the words that should not be included in queried domains, including ".local", "wpad.", "http://", and so forth.
5) Repeated TLD: These domains could be introduced by misconfiguration of a web browser or other applications. An example is "www.example.com.example.com".

We use dataset CSUDNSTrace to study the effectiveness of each filter. Before applying the filters, we remove all the queried NXDomains (e.g., test.colostate.edu) under our university domain "colostate.edu". We list the number of filtered domains and their ratio to the total number of NXDomains in Table III. From the table we can see that the filters can remove 87.3% of NXDomains, saving lots of computation resources and improving the performance.

TABLE III: Filters Statistics

Filters	Filtered Domains	Percentage
Overloaded DNS	1232214	7.1%
Unregistered TLD	2488055	14.4%
Typo Domains	174515	1.01%
Misconfiguration words	7172856	41.4%
Repeated TLD	4046912	23.4%
All Filters	15114552	87.3%

E. Quantity Evidence

Quantity evidence is based on the assumption that most hosts in a network are legitimate, and they do not query many suspicious 2LDNXs. On the other hand, a bot queries a lot of suspicious 2LDNXs. As a result, a bot can be considered as outlier in terms of the number of queried suspicious 2LDNXs. To define an outlier, we first calculate the average and standard deviation of the numbers of suspicious 2LDNXs for all the hosts, denoted as *avg* and *stddev*, respectively. Then we set the threshold as $avg + 3 * stddev$. If the number of suspicious 2LDNXs queried by a host is greater than the threshold, then the host is labeled as an outlier.

We demonstrate quantity evidence using a real bot. First we pick one hour DNS trace from CSUDNSTrace and one bot trace from 140Samples. Then, we blend them together by adding the bot's DNS traffic on a randomly selected host in the background trace. Finally we have a one hour mixed DNS trace including 11,720 background hosts and a known bot. The CDF for the number of suspicious 2LDNXs queried by a host is shown in Figure 2. From the figure we can see that more than 95% hosts do not query any suspicious 2LDNX. On the contrary, the bot queries more than 20 suspicious 2LDNXs thus it is labeled as an outlier.

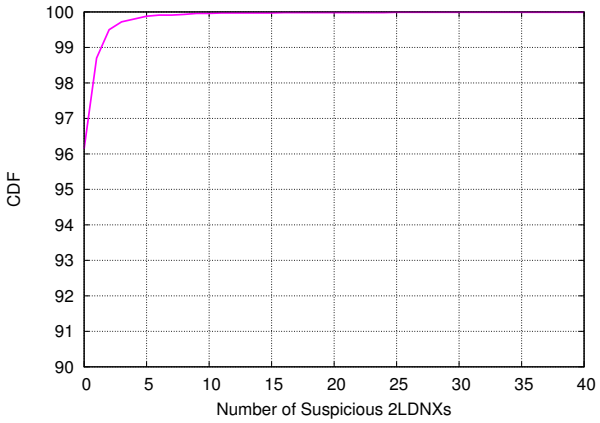

Fig. 2: CDF for Suspicious 2LDNXs

F. Temporal Evidence

Most of the time, a bot behaves like legitimate host and it does not query many suspicious 2LDNXs. However, when the bot wants to look for the C&C domain, it will query many suspicious 2LDNXs. Consequently, the number of suspicious 2LDNXs suddenly increases. Then, once a bot hits the registered C&C domain, it will stop querying more DGA domains, thus the number of suspicious 2LDNXs will decrease. In other words, we are detecting a period of time when the number of suspicious 2LDNXs suddenly increases and decrease. This can be considered as a *Change Point Detection* (CPD) problem. In this paper, we use *Cumulative Sum* (CUSUM) as the CPD algorithm because it has been proved to be effective and has been used in many other works, e.g., [21].

Let $X_n, n = 0, 1, 2, ...$ be the number of suspicious 2LDNXs queried by a given host every minute during a time window.

Let

$$\begin{cases} y_n = y_{n-1} + (X_n - \alpha)^+, \\ y_0 = 0, \end{cases} \tag{1}$$

where x^+ equals to x if $x > 0$ and 0 otherwise. α is the upper bound of suspicious 2LDNXs queried by legitimate host every minute. The basic idea of this algorithm is that $X_n - \alpha$ is negative when a host behaves normally, but it will suddenly increase to a relatively large positive number when a bot begins to query C&C domain.

Let b_N be the decision of whether a host has sudden increased suspicious 2LDNXs at time *n*. A host is considered to have a sudden increase in suspicious 2LDNXs when b_N equals to 1. *N* is a threshold that indicates the number of suspicious 2LDNXs a host must reach before considered as bot candidate. *N* is specified by the user, as discussed in Section IV-A.

$$b_N(y_n) = \begin{cases} 0, & if \quad y_n \leq N \\ 1, & if \quad y_n > N \end{cases} \tag{2}$$

The method to detect a decrease in the number of suspicious 2LDs is similar to equation 1, and is defined in equation 3. The function to detect sudden decrease of suspicious 2LDNXs is the same as equation 2.

$$\begin{cases} y_n = y_{n-1} + (X_n - X_{n-1})^+, \\ y_0 = 0, \end{cases} \tag{3}$$

Now we investigate the temporal evidences using the same one hour DNS traffic blended in the last section. As it is hard to plot all the 11,720 hosts and due to the fact that most of them do not query many suspicious 2LDNXs, we only plot the suspicious 2LDNXs for 15 hosts including the bot in Figure 3. From the figure we can clearly see there is a spike appearing between minute 29 and 32, standing for the bot. However, we also notice that besides the bot, some legitimate hosts (e.g., host 10) also have spikes, thus they might also be labeled as suspicious. Spikes by legitimate hosts can result from user typos. Another explanation may be that each time Google Chrome starts it generates a number of failed DNS requests to determine if NXDomain rewriting is enabled [24]. We manually checked the NXDomains queried by the hosts that generated the spikes. While we found some of them not suspicious, we could not precisely pinpoint the reason for the spikes.

It is true that a single evidence is not strong enough to label a host as a bot. This is the reason why we use a chain of multiple evidences that helps to reduce false positives.

G. Linguistic Evidence

Linguistic evidence is built on two assumptions. The first is that the NXDomains queried by a bot are generated by the same algorithm, thus they share similar linguistic attributes (e.g., entropy, length of domains, etc.). On the contrary, legitimate domains are not generated algorithmically but they are selected such that people can remember them easily. Consequently, the linguistic attributes of the NXDomains

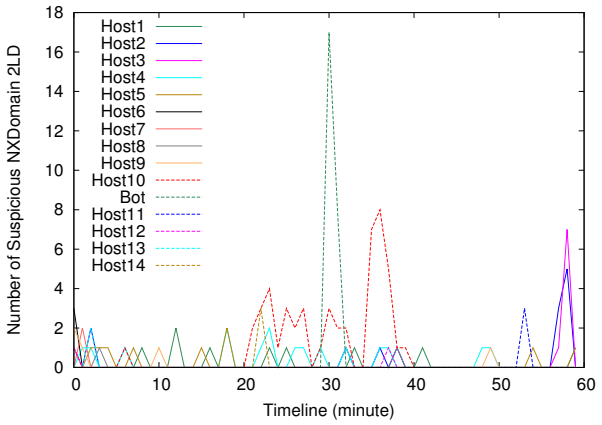

Fig. 3: Timeline for Suspicious NXDomains 2LD

queried by a legitimate host are not similar to each other. The second assumption is that both DGA registered C&C domains and NXDomains are generated by the same algorithm. The only difference between them is whether the domain is registered or not. Consequently, C&C domains have similar linguistic attributes with DGA NXDomains. Based on these two assumptions, we first extract linguistic attributes from suspicious NXDomains and cluster the domains that have similar attributes together. After that, bot NXDomain cluster candidates are decided. Next, signatures are extracted from the cluster candidates and applied on successfully resolved domains looking for registered C&C domains.

1) Linguistic Attributes: We extend the attributes used in prior work [8], [20], [9], [6], [22] to 23 and list them in Table IV. From the table we can see that some attributes are dependent (e.g., length of dictionary words and percent of dictionary words). Currently we use all of these attributes, and we leave the study of attributes selection as a future work.

TABLE IV: Domain Linguistic Attributes

Index	Linguistic Attributes
1, 2	length of dictionary words in 2LD and 3LD
3, 4	percent of dictionary words in 2LD and 3LD
5, 6	length of the longest meaningful substring (LMS) in 2LD and 3LD
7, 8	percent of the length of the LMS in 2LD and 3LD
9, 10	entropy in 2LD and 3LD
11, 12	normalized entropy in 2LD and 3LD
13, 14	number of distinct digital characters in 2LD and 3LD
15, 16	percent of distinct digital characters in 2LD and 3LD
17, 18	number of distinct characters in 2LD and 3LD
19, 20	percent of distinct characters in 2LD and 3LD
21, 22	length of 2LD and 3LD
23	number of domain levels

2) Dissimilarity Calculation: Now we describe how to calculate dissimilarity of a single linguistic attribute between two domains. We denote two domains as D_1 and D_2, and denote their attributes as $a_{ij}, i = 1, 2$ and $1 \leq j \leq 23$. Dissimilarity of attribute j between D_1 and D_2 is denoted as $S_j(D_1, D_2)$ and calculated as follows.

$$S_j(D_1, D_2) = \begin{cases} 0 & \text{if } a_{1j} = 0 \, and \, a_{2j} = 0 \\ \frac{|a_{1j} - a_{2j}|}{max(a_{1j}, a_{2j})} & \text{else} \end{cases} \quad (4)$$

We use a modified version of Euclidean distance to calculate the overall dissimilarity of all the 23 attributes between two domains. Euclidean distance [11] has been used by others [6], [9]. The overall dissimilarity between D_1 and D_2 is denoted as $S_{All}(D_1, D_2)$ and calculated as equation 5. The smaller the dissimilarity, the more similar D_1 and D_2 are.

$$S_{All}(D_1, D_2) = \sqrt{\frac{\sum_{j=1}^{23} S_j(D_1, D_2)^2}{23}} \quad (5)$$

3) Clustering NXDomains: After extracting attributes from NXDomains and calculating dissimilarities, we run the single linkage hierarchical clustering algorithm to group the NXDomains that have similar attributes together. Process of the clustering algorithm is shown in Figure 4. Initially, each NXDomain denoted as orange dot is in a cluster of its own. Then, the clusters are combined into larger ones, until all NXDomains belong to the same cluster. At each step, the two clusters having the smallest dissimilarity are combined. The result of the clustering process can be depicted as a dendrogram, where a cut is used to separate the clusters, giving us a set of NXDomain clusters. For example, the dendrogram cut in Figure 4 gives two clusters. As we normalize the dissimilarities between domains, the dendrogram height is between 0 and 1. Currently the dendrogram cut height is determined experimentally, as shown in Section IV-A. As a future work, we plan to use statistical methods to cut the dendrogram dynamically. Finally, we compare the size of every cluster with a *BotClusterThreshold*. If a cluster has more NXDomains than the threshold, it is considered as a *bot NXDomain cluster candidate*, and the host is considered as a bot candidate.

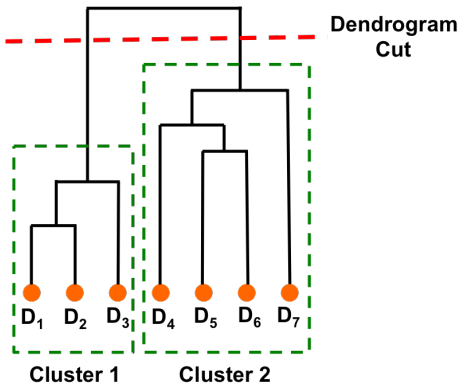

Fig. 4: Hierarchical Clustering Dendrogram

4) C&C Domain Detection: After detecting bot NXDomain cluster candidates, we extract signatures from them, and then apply the signatures on successfully resolved domains to detect C&C domain.

For a given bot candidate c, all its bot NXDomain cluster candidates are denoted as $C_1, C_2, ...C_n$. We first combine these clusters as a $C = \bigcup_{i=1}^{n} C_i$. The NXDomains included in C are denoted as d_j. Each domain d_j contains 23 linguistic attributes a_{jk}, $1 \leq k \leq 23$. Signatures are composed of an upper signature and a lower signature, each of them includes 23 values. The upper signature is denoted as $Sig_{upper} = (s_1, s_2, ...s_{23})$. Each value in the upper signature is defined as the maximum value of the corresponding linguistic attribute of all the domains in C, $s_k = max(a_{jk}), 1 \leq k \leq 23$, $d_j \in C$. Similarly, the lower signature, Sig_{lower} is defined as the minimum values.

Once we obtain the signatures, we apply them on the successfully resolved domains that are queried during (t_{begin}, t_{end}) to extract C&C domains. Recall that (t_{begin}, t_{end}) is decided by temporal evidence in Section III-F. For a given successfully resolved domain, we first extract its 23 attributes. Then, for each attribute we check whether it falls within the corresponding attribute upper and lower bounds in the signature. Finally we compare the total number of matched attributes with a *SignatureThreshold*. If the former is greater, then we label the domain as C&C domain, and label the host as a bot.

IV. Evaluation

A. True Positives

First, we evaluate the performance of BotDigger on detecting DGA-based bots. We use two botnets, Kraken and Conficker in our experiments. Recall that the dataset 140Samples includes 140 real Kraken traces that contain DNS queries/responses, C&C communication and other traffic. On the other hand, for the evaluation with the Conficker dataset we only have Conficker's DGA domains. We use these DGA domains to simulate 1000 bots. Each simulated bot randomly queries 20 domains from the Conficker domain pool every 10 seconds during the time window. We use 5 minutes as the time window in all of our experiments. At the end of every time window BotDigger analyzes the collected information and looks for bots. Users can decrease the time window if they want to detect bots more quickly, but decreasing the time window risks missing bots with slow activity.

Before running BotDigger on the two evaluation datasets we experimentally determine the parameters and thresholds introduced in Section III. First, we use a one-day DNS trace from the CSUTrace to decide α in equation 1. We find that more than 98% of the hosts query less than two suspicious 2LDNXs per minute, so we pick 2 as the α. N in equation 2 is set to the value of BotClusterThreshold. We then set SignatureThreshold experimentally by trying different values of the threshold and run BotDigger on the 1000 Conficker bots simulated above. In this specific experiment, we use 0.05 as the dendrogram cut. We will discuss how to pick the proper dendrogram cut in the next paragraph. The results are plotted in Figure 5. The x-axis and y-axis are bot detection rate and SignatureThreshold respectively and the different lines stand for different BotClusterThreshold. From the figure we can see that the bot detection rate is stable when the SignatureThreshold is less than 16, after that the rate drops quickly. As a result we set 16 as the SignatureThreshold.

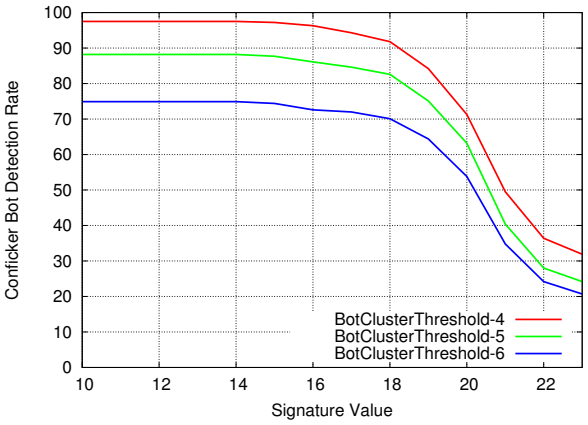

Fig. 5: Signature Threshold Selection

We run BotDigger on Kraken and Conficker bots using combinations of two parameters, BotClusterThreshold and dendrogram cut in the hierarchical clustering algorithm. The results are shown in Figure 6 and Figure 7. The x-axis is the dendrogram cut, y-axis is the percentage of detected bots, and different lines stand for different BotClusterThreshold. From the figures we can see that by using 0.10 as the dendrogram cut, we are able to detect all the Kraken bots and 99.8% of Conficker bots.

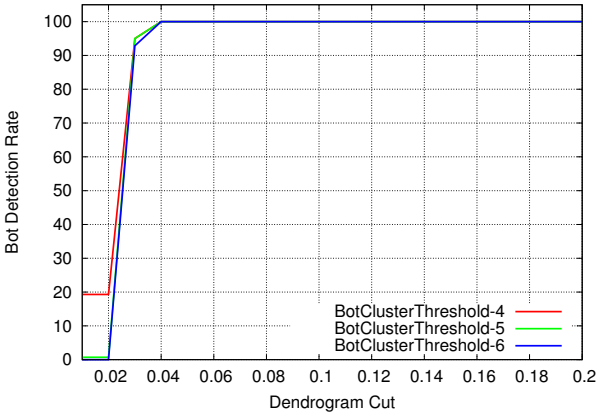

Fig. 6: Kraken Bots Detection

B. False Positives

Besides DGA-based bots, some legitimate hosts could also query similar NXDomains that are clustered together and the host could falsely be labeled as a bot. We now evaluate such false positives by using the dataset CSUTrace. As most of the users connected to CSU network are required to install anti-virus software, we assume CSUTrace does not contain many bots. We use 0.10 as the dendrogram cut and 4 as BotClusterThreshold.

33 hosts (0.16% of all the hosts) are labeled as bots during the entire period of one week. We use two resources to check whether these domains and corresponding IPs are suspicious. The first resource is *VirusTotal* [5], a website that

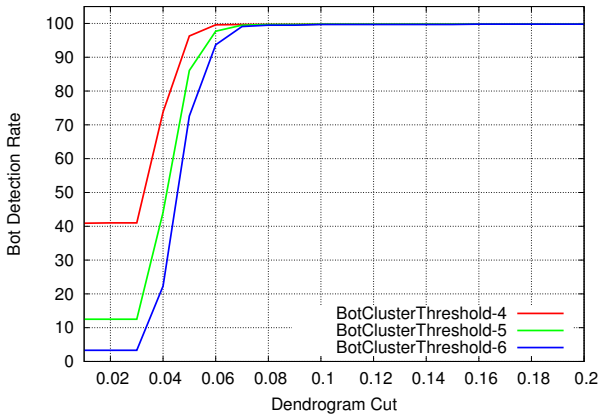

Fig. 7: Conficker Bots Detection

provides virus, malware and a URL online scanning service based on 66 URL scanners, including ZeusTracker, Google Safebrowsing and Kaspersky among others. Another resource is *TrustedSource* by McAfee labs [4], a real-time reputation system that computes trustworthy scores for different Internet entities including IP addresses, domains and URLs. The scores are computed based on analyzing real-time global threat intelligence collected from McAfee's research centers' centralized scanning systems and honeypots and from customers and end point software solutions around the world. We check the labeled C&C domains and the corresponding IPs for each host using the above resources. If any of them is labeled as malicious or high risk, we consider the host as malicious. The results show that 22 hosts are labeled as malicious. In summary, we falsely label 11 hosts as bots, resulting in the false positive rate as 0.05%.

Within the 22 malicious hosts, we find that 5 hosts are highly suspicious. Most of the NXDomains queried by these hosts are random looking. In Figure 8, we list the NXDomains queried by such a host. After extracting signature and applying it on the successfully resolved domains, "sfaljyxfspbjftnv5.com" is labeled as C&C domain. The IP address of this C&C domain is 85.25.213.80. Upon further investigation on VirusTotal, we find that this IP is related to malicious activity and mapped to various random looking domains. Consequently, we believe this host is a bot.

obhgiunuht7f.com,	hlxgrygdmcpu8.com
ompxskwvcii3.com,	nnnwyujozrtnulqc5p.com
nwpofpjgzm6c.com,	dgpvsgsyeamuzfg2.com
kgapzmzekiowylxc5k.com,	nellwjfbdcfjl3g.com
rfsbkszgogjqlbm.com,	rcfwptxhgoiq27.com
unzxnzupscqxu.com,	okihwecmaftfxwz.com

Fig. 8: NXDomains Queried by Suspicious Host A

Within the 11 false positives, we manually checked their queried NXDomains and found some interesting results. We include one example in Figure 9. From the figure we can find that some words appear frequently in the domains, such as "coach", "louis", "outlet", etc. In addition, we can also see that the domains contain typos. For example, "coach" and "louis" are misspelled as "coachh" and "llouis". One explanation of these typos is that the host tries to avoid conflicts with registered domains. In addition, we extract signature from NXDomains and apply it on the successfully resolved domains. By doing this, domain "www.golvlouisvuittonbags.com" is extracted. This domain is still in use, trying to sell replica Louis Vuitton. The above example shows that BotDigger is not only able to detect bots and suspicious hosts that query random looking domains, but is also capable of detecting hosts whose queried domains are generated from a set of dictionary words.

www.llouisvuittonoutlet.org,	www.iulouisvuittonoutlet.org
www.coachoutletsstoree.com,	www.illouisvuittonoutlet.com
www.coachfactorysoutlets.net,	www.louisvuittonof.com
www.coachfactoryonlines.net,	www.colvlouisvuittonoutlet.net
www.coachfactorystoreoutlete.org,	www.coachhoutlet.com

Fig. 9: Domains Queried by Suspicious Host B

In addition to the CSUTrace we also ran BotDigger on the traces captured in our research lab, Lab1, Lab2, and Lab3. Note that we do not regard these traces as ground truth. Although our network is well protected we cannot be certain it is bot-free. BotDigger detects four hosts as potential bots, denoted as $H_i, 1 \leq i \leq 4$. Then we try to confirm by running a bot detection system - BotHunter, on the three lab traces for independent verification. BotHunter labels H_1 as bot. Upon further investigation, we find that H_1 queries many domains and these domains are very similar as the ones queried by H_2. Moreover, some of the labeled C&C domains of H_1 are resolved to 80.69.67.46, which is the same IP as the labeled C&C domains for H_2. As a result, H_2 is very likely to be a bot. Besides, H_3 is highly suspicious because it queries many NXDomains, all of them beginning with "zzzzzzzzzzzgarbageisgreathere" (e.g., zzzzzzzzzzzgarbageisgreathere.funatic.cz, .penguin.se, inspirit.cz, etc). In summary, BotDigger introduces 1 false positive (0.39%) in three traces.

V. Limitations

A bot can bypass BotDigger by querying C&C very slowly, for example, query a domain every 5 minutes. In this case, BotDigger may not detect it if a small time window is used. However, at least we make the bots less effective, meaning that it may take hours to contact to the C&C domains if the bots have to query tens of domains slowly. Notice that we can increase the time window to detect the bots that query domains slowly, but we expect more false positives will be introduced.

The quantity evidence in the evidence chain of BotDigger requires that the number of NXDomains queried by a bot is comparable more than legitimate hosts. As a results, BotDigger will fail to work if the bot is "lucky", meaning that it only queries a very small number of domains and hits the C&C domain. However, when the current C&C domain expires, the bot needs to look for the new C&C domain. It is very unlikely that the bot is "lucky" every time it looks for the C&C domain. Consequently, once the number of NXDomains queried by this bot matches the quantity evidence, BotDigger will analyze its DNS traffic for detection. In summary, a bot may evade the system for one time, but not all the time.

VI. Related Work

Researchers introduced a lot of works to detect malicious domains (e.g., C&C domains, phishing pages, etc.) and DGA-based botnets. In [9], Bilge et al. introduce EXPOSURE to detect malicious domains. In particular, they first extract 15 features from a domain and all the IPs that the domain is mapped to. Then they build a decision tree from a training set. After that, any given domain can be classified as malicious or legitimate using the tree. In [6], Antonakakis et al. introduce Notos to build models of known legitimate and malicious domains using 18 network-based features, 17 zone-based features, and 6 evidence-based features. These models are then used to compute a reputation score for a new domain. Schiavoni et al. introduce Phoenix to distinguish DGA domains from non-DGA domains by using both linguistic and IP features [20]. However, Phoenix only works for randomly generated domains, but not the domains generated based on pronounceable words. On the contrary, our work can detect the suspicious domains generated using a set of pronounceable works, as shown in Figure 9 in Section IV-B. Antonakakis et al. introduce Pleiades [8] to detect DGA-based bots by looking for large clusters of NXDomains that have similar linguistic features, and are queried by multiple possible infected hosts. In [22], Yadav et al. measure K-L with unigrams, K-L with bigrams, jaccard index, and edit distance from training data set, and then use them to detect DGA-based botnets. Later on, they introduce another method that utilizes entropy of NXDomains and C&C domains for detection in [23]. In [7], Antonakakis et al. introduce a system named Kopis to detect malware-related domains, including C&C domains. Kopis is deployed in upper level DNS servers to provide global visibility of DNS queries and responses for detection.

A big difference between the above work and ours is that many of them use IP/domain blacklists and reputation systems. BotDigger does not require any of these. Moreover, the above works either require DNS traffic collected at upper level (e.g., TLD servers) or among multiple networks, and require multiple bots belonging to the same botnet appearing in the collected data set. As we discussed in Section I, the above requirements introduce many challenges. On the contrary, our method is capable of detecting *individual* DGA-based bot using DNS traffic collected in *a single network*.

Another work similar as ours is [10]. In this work, the authors also use NXDomains to detect bots, and use NoError domains to track C&C domains. However, the introduced method ignore all the domains queried by a single host and only focus on detecting a group of bots. Consequently, this method cannot detect individual bot.

VII. Conclusions

In recent years, botnets began to use DGA techniques to improve the resiliency of C&C servers. In this paper, we introduce BotDigger, a system that detects DGA-based bots without a priori knowledge of the domain generation algorithm. A big advantage of BotDigger is that it can detect an individual bot by only analyzing DNS traffic collected from a single network. Any network administrator can run BotDigger without requiring additional information from other networks. A novel method - a chain of evidences, including quantity evidence, temporal evidence and linguistic evidence, is used in BotDigger for detection. We first use synthetic traffic to investigate each individual evidence and find many false positives. A chain of evidences helps reduce false positives because most of the legitimate hosts match one or two evidences but not all three. Two DGA-based botnets and two groups of background traces are used to evaluate BotDigger. The results show that BotDigger detects more than 99.8% of the bots with less than 0.5% false positives.

References

[1] Alexa top sites. http://www.alexa.com.

[2] Forbes worlds biggest companies. http://www.forbes.com.

[3] Iana root zone database. http://www.iana.org/domains/root/db.

[4] McAfee TrustedSource reputation system. http://trustedsource.org.

[5] Virustotal. https://www.virustotal.com/en/.

[6] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a dynamic reputation system for dns. In *USENIX security symposium*, pages 273–290, 2010.

[7] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, II, and D. Dagon. Detecting malware domains at the upper dns hierarchy. In *The 20th USENIX Conference on Security*, 2011.

[8] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou II, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *USENIX security symposium*, 2012.

[9] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *NDSS*, 2011.

[10] A. Br, A. Paciello, and P. Romirer-Maierhofer. Trapping botnets by dns failure graphs: Validation, extension and application to a 3g network. In *INFOCOM*, pages 3159–3164. IEEE, 2013.

[11] M. M. Deza and E. Deza. *Encyclopedia of distances*. Springer, 2009.

[12] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: Malware analysis via hardware virtualization extensions. In *The 15th ACM Conference on Computer and Communications Security*, 2008.

[13] N. Jiang, J. Cao, Y. J. 0001, E. L. Li, and Z.-L. Zhang. Identifying suspicious activities through dns failure graph analysis. In *ICNP*, 2010.

[14] Y. Kazato, K. Fukuda, and T. Sugawara. Towards classification of dns erroneous queries. In *The 9th Asian Internet Engineering Conference*, pages 25–32. ACM, 2013.

[15] F. Leder and T. Werner. Containing conficker - webpage and tools. http://iv.cs.uni-bonn.de/conficker.

[16] F. Leder and T. Werner. Know your enemy: Containing conficker.

[17] M. Mowbray and J. Hagen. Finding domain-generation algorithms by looking at length distribution. In *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2014.

[18] D. Plonka and P. Barford. Context-aware clustering of dns query traffic. In *The 8th ACM SIGCOMM Conference on Internet Measurement*, 2008.

[19] P. Royal. Analysis of the kraken botnet, 2008.

[20] 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*, pages 192–211. Springer, 2014.

[21] H. Wang, D. Zhang, and K. G. Shin. Detecting syn flooding attacks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies.*, 2002.

[22] S. Yadav, A. K. K. Reddy, A. Reddy, and S. Ranjan. Detecting algorithmically generated malicious domain names. In *The 10th ACM SIGCOMM Conference on Internet Measurement*, pages 48–61, 2010.

[23] S. Yadav and A. N. Reddy. Winning with dns failures: Strategies for faster botnet detection. In *Security and Privacy in Communication Networks*. Springer, 2012.

[24] F. Yarochkin, V. Kropotov, Y. Huang, G.-K. Ni, S.-Y. Kuo, and Y. Chen. Investigating dns traffic anomalies for malicious activities. In *Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on*, pages 1–7. IEEE, 2013.