Proactive Detection of Algorithmically Generated Malicious Domains

Jeffrey Spaulding, Jeman Park
University of Central Florida
{jjspauld, parkjeman}@knights.ucf.edu

Joongheon Kim
Chung-Ang University
joongheon@cau.ac.kr

Aziz Mohaisen
University of Central Florida
mohaisen@cs.ucf.edu

***Abstract*—Using an intrinsic feature of malicious domain name queries prior to their registration (perhaps due to clock drift), we devise a difference-based lightweight feature for malicious domain name detection. Using NXDomain query and response of a popular malware, we establish the effectiveness of our detector with 99% accuracy, and as early as more than 48 hours before they are registered. Our technique serves as a tool of detection where other techniques relying on entropy or domain generating algorithms reversing are impractical.**

***Index Terms*—DNS, Machine Learning, Classification**

I. Introduction

With a complex ecosystem around them, botnets are becoming increasingly one of the most prevalent threats on the Internet [4], [12]–[14]. Botnets typically consist of various infected hosts, command and control (C2) channels, and a botmaster. The infected hosts ("zombies"), as shown in Figure 1, are often massively distributed, whereas the command and control are channels used by a mastermind (the "botmaster") to instruct bots to perform various forms of malice; *e.g.,* launching distributed denial-of-service (DDoS) attacks [15]. To communicate with bots, there are several potential ways utilized by botmasters, and domain names as a command and control channel are one of the most common and preferred methods—because they are easy to acquire and recycle [5]. To generate such domain names, domain generation algorithms (or DGAs) are widely used today by botmasters. Usually, DGAs use time to dynamically and automatically generate potentially pseudorandom domain names that are registered by botmasters and used by bots for communication. One way of mitigating botnets is to prevent them from registering their C2 domains, or by taking such domain names down [6], [10].

To address DGAs by detection, there have been two schools of thought that either: 1) rely on reverse engineering of the bot software [8] or 2) using the intrinsic features of the generated domains [16]. The first method, while powerful in generating all domain names to be potentially used by a malware family (even in the future and can thus be used to proactively block those domains by pre-registering them), is very expensive. This method requires obtaining samples of the malware family that utilizes such DGAs. However, obtaining such malware is not the biggest hurdle: many of today's malware families employ obfuscation techniques that make their analysis a difficult task.

978-1-5386-2290-2/18/$31.00

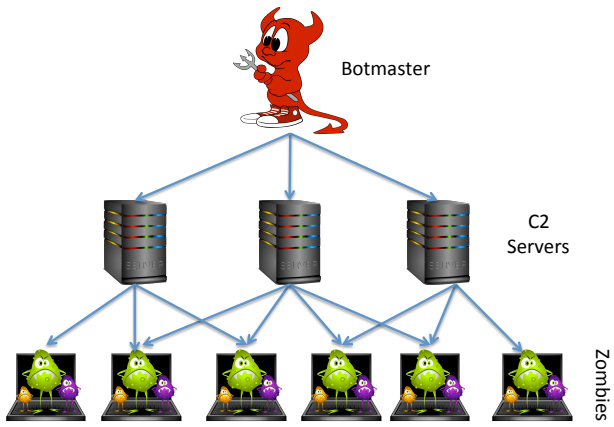

Fig. 1. An illustration of a botnet infrastructure.

The second school of thought uses pseudorandomness of algorithmically generated domains and exploits the fact that those domains have a high entropy for their detection [3]. Registered domain names that are queried by infected hosts are evaluated, a measure of their pseudorandomness using their entropy is calculated, and the likelihood of them being malicious based on their entropy score is assigned. While shown to identify malicious domains reasonably well, such techniques suffer from various drawbacks. First, domain names need to be registered in order for a monitor to be able measure such entropy and determine if a domain is malicious or not. Thus, such techniques cannot be used proactively to detect malicious domains. Second, those techniques assume that randomly generated domain names are only used in malicious activities: it is not far-fetched to imagine that domain names with high entropy are utilized for domain name parking, non-public facing domains (*e.g.,* content delivery network (CDN) addressing), among others.

Key Idea. To this end, this paper addresses the problem of proactive malicious domain name identification focusing on DGAs. We aim to identify such domain names before they are registered using domain name resolution side channels. Our main source of inference is the domain name system (DNS) query and resolution of domain names. We motivated for our study by a large-scale analysis of domain names and their queries. We find that, domains that are used for malicious activities, and especially those generated algorithmically, tend to have unique and distinguishing patterns. In particular, domain

names that are algorithmically generated tend to have a large number of DNS queries even before their registration, typically resulting in NXDomain (non-existent domain) responses. This trend persists, and the number of queries increases and peaks at the time of registration, then declines gradually–indicating the ephemeral use of those domain names for their major purpose. On the other hand, domain names that are being used for benign applications tend to have significantly less queries before registration, while their post-registration query volumes (which may fluctuate over time) do not have a single declining curve, thus highlighting a fundamentally different use model.

The contribution of this paper is as follows. First, we highlight a fundamental difference between the query patterns for domain names that are used by botnets, often generated using DGAs, and benign ones. We use this insight to differentiate between those domain names using a simple classification algorithm for proactive detection of malicious domains.

The organization of the rest of this paper is as follows. In section II we review our datasets and high-level characteristics. In section III we present our online detector. In section IV we evaluate our system. In section V, we provide concluding remarks.

II. Dataset and Characteristics

Our dataset was originally used by Thomas and Mohaisen in [12]. The dataset was collected during July of 2012 from Verisign's authoritative name servers for the COM, NET, TV and CC top-level domain (TLD) authoritative name servers. As the registry of large TLDs, Verisign has a global view of DNS traffic, giving a unique observation of malware-associated DNS traffic.

Malware Data. Conficker is one of the most well-known malware samples that employed the use of DGAs [9]. The family was originally discovered in 2008, and has been active for the past several years by infecting many hosts world-wide and by mutating into multiple variants, namely Conficker A, B, C, D, and E [11]. The Conficker Working Group, a consortium of researchers and security professionals, has successfully reverse-engineered the DGA and pre-calculated domains to be generated each day for variants A, B and C [1]. Table I shows the various TLDs for variants A, B and C and the domains to be generated on a daily basis. Variants A and B utilize the COM, NET and CC TLDs allowing us to analyze the DNS traffic for them. By April 2009, all domains generated by variant A were successfully locked or preemptively registered to mitigate the proliferation capabilities of the variant. Of the 15,500 domains to be generated by variants A and B in July of 2012 (corresponding to 500 domains per day over 31 days), 30 of the domains were registered in either COM or NET with active name servers resulting in YXDomain (name exists when it should not) traffic while the remaining DGA domains resulted in NXDomain traffic.

NXDomain Data. NXDomain is the answer type for a domain name that is unable to resolve because, among other reasons, the domain name is not registered. The term was originally used to represent DNS response code 3 in RFC 1035 [7] and RFC 2308 [2]. All of the data below corresponds to the state of the DNS resolution system operated by Verisign in middle of 2012. We note that while we are not able to use DNS traffic for some of the TLDs listed in Table I, the ones using TLDs operated by Verisign (CC, TV, NET and COM) were captured, measured and analyzed [12].

TABLE I
Conficker DGA by variant and domains per day

Variant	Domains	TLDs
A	250	biz, info, org, net, and com
B	250	biz, info, org, net, com, ..., cn
C	50k	110 ccTLDs not tv or cc

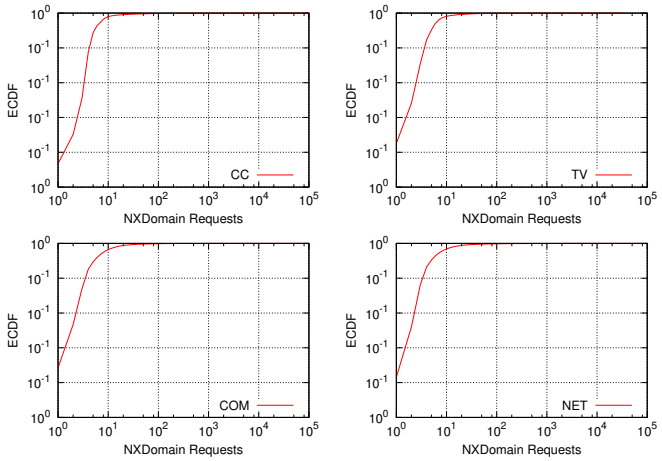

Fig. 2. NXDomain traffic volumes to major TLDs: .cc, .tv, .com, and .net, respectively. Notice that the majority of domains receive small number of queries, and a small percentage (∼3%) receive more than 10 queries.

In our dataset, a typical day in the COM zone has 2.5 billion NXDomain requests for more than 350 million unique second-level domains while NET receives about 500 million NXDomain requests for more than 60 million unique second-level domains. Smaller zones such as TV and CC receive several magnitudes less of NXDomain traffic than COM and NET. While daily volumes of requests and unique domains are extremely high, the vast majority of individual NXDomains observed receive very few requests within a given epoch of time. Figure 2 (subplot for each zone) shows a cumulative distribution function of the number of requests a given NXDomain receives in 24 hours. As depicted, more than 95% of the unique second-level NXDomains receive less than 10 requests within the 24 hour.

Conficker NXDomain DNS. The vast majority of the domains generated by the Conficker DGA falls into the NXDomain category. We analyzed various aspects of DNS traffic before, during and after the domain's generation date to understand the lifecycle of a DGA domain with respect to DNS traffic. Using the 2012 Conficker Domain list of pre-calculated DGA domains, we were able to group domains by their generation date and measure their DNS traffic [12]. Specifically, for a given domain to be generated on day x, we measured the domain's DNS traffic on days $x-5$ to $x+5$. Figure 3 depicts

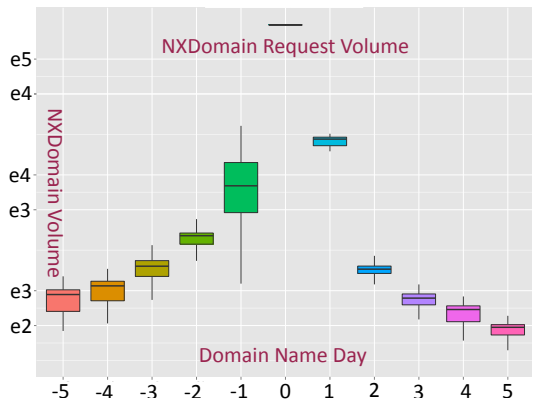

Fig. 3. Conficker NXDomain DNS lookups [12].

the pre, during and post-DNS traffic patterns for Conficker B with box plots to depict the range of DNS traffic observed on a given day. It is evident that despite a specific generation date, DGA domains receive significant volumes of traffic during its pre and post-generation date.

III. Online Detection Algorithm

Compared to the whole population of NXDomain traffic, the volume and daily reoccurrence make DGA domains statistically abnormal. Significant traffic uptick for a given DGA domain occurs both one day prior and post-generation date. Traffic volumes on the exact generation date soar several magnitudes higher than the ± 5 days baseline to $42,887$ unique 24 recursive name servers consisting of $199,097$ total NXDomain requests from 211 unique countries for an average DGA domain.

A. Rationale of Feature

There are potentially various explanations for why domain names get queried before their registration. First, domain names intended for benign usage might get queried by interested registrants who want to acquire them, thus explaining the small number of queries some of them receive before registration. On the other hand, the large number of queries that DGA domains receive might have to do with the fact that those domain names are time-dependent. As highlighted in Figure 4 for the DGA domain used initially by CryptoLocker, a bot calculates a domain name using time features as the main inputs. A large drift in the time or clock misconfiguration would result in bots contacting the domain name before or after its registration. We use such verified observation as the main feature for identifying malicious domain names. Using this *a priori* knowledge captured in a training model, we devise a method that can detect those domains even before they are registered with a high accuracy rate. The proposed approach based on this feature has various plausible benefits over the state-of-the-art in the following aspects. Mainly, the proposed technique is robust to behavior of the adversary.

```
def generate_domain(y, m, d):
    domain = ""
    for i in range(16):
        y=((y^8*y)>>11)^((y&0xFFFFFFF0)<<17)
        m=((m^4*m)>>25)^16*(m&0xFFFFFFF8)
        d=((d^(d<<13))>>19)^((d&0xFFFFFFFE)<<12)
        dm+=chr(((y^m^d)%25)+97)
    return domain
```

Fig. 4. An example of the algorithm used by CryptoLocker for initially generating algorithmic domain names for command and control.

B. Online Detection

To be able to tell whether a domain name is malicious or not, we use the notion of the difference function. The difference function provides an estimation of the derivative of a function and is defined as follows:

$$\Delta f(x) = \frac{f(x+a) - f(x-a)}{2a} \tag{1}$$

Building Feature Vectors. For a window of size $2a$ (a from left and right of a point x; note that x here can be any point in time), we calculate the difference as the change in the number of NXDomain responses to a given domain, normalized by the total time units corresponding to $2a$. The parameter a is used based on the desired performance, and x is used for all values of the observed traffic. We highlight the operation of the basic feature with an example. Let's consider an observation of $[o_1^j, o_2^j \ldots, o_k^j]$ (for a domain j), where each observation is the total number of NXDomain responses for a domain j over a fixed period of time (*e.g.,* hour). If we are to consider $a = 1$, we calculate $f(x)$ as $|o_{i+1}^j - o_i^j|/2$ for all i (resulting in a vector of values, representing the use of the given domain; $[f_i^j]^{1\times k}$). For a unified treatment of the vector, we normalize each element in it by the sum of all of its elements; this is $f^j - i/\sum_{\forall i} f^j - i$. Our detection algorithm then uses the same idea as above, over a sliding window of observations. As time goes, the window slides by forgetting the oldest observations of NXDomain responses for the given domain. Additionally, the detector updates the count vector of the NXDomain responses for the domain, and calculates our feature vector as the difference function.

Building a Model. For a set of domains $d_1, \ldots, d_t$ that are known to be malicious, we create model $\mathcal{M}$ that is calculated as a centroid feature vector corresponding to the average of the feature values of the different domains. As such, we define

$$\mathcal{M} = [m_1, \ldots, m_k] : m_i = 1/t \sum_{j=1}^{t} f_i^j \tag{2}$$

As above, for a unified treatment, we normalize each element in it by the sum of all of its elements; this is $m_i/\sum_{\forall i} m_i$

Learning Labels of Domains. For a candidate domain x defined by its difference function f^x as above, we determine the label of the domain by conducting the following. We calculate the distance between $\mathcal{M}$ and f^x. That is, we calculate:

$$D(\mathcal{M}, f^x) = \frac{1}{2} \sum_{\forall i} |m_i - f_i^x| \tag{3}$$

Then, we label the domain as malicious if $D(\mathcal{M}, f^x) > \Delta$ and as benign otherwise. Δ is determined through measurements

TABLE II
STANDARD MEASUREMENTS OF PERFORMANCE: TRUE POSITIVE, TRUE NEGATIVE, FALSE POSITIVE, FALSE NEGATIVE FOR DIFFERENT WINDOWS SIZE (AVERAGE, OVER 24 SLIDES FOR THE GIVEN W SIZE).

W	T_P	T_N	F_P	F_N	P	R	A	$F1$
8	91.3	89.4	10.6	8.7	0.89	0.91	0.90	0.90
16	97.4	92.7	7.3	2.6	0.93	0.97	0.95	0.95
24	98.1	94.5	5.5	1.9	0.95	0.98	0.96	0.96
36	99.3	95.5	4.5	0.7	0.96	0.99	0.97	0.98
48	99.4	98.3	1.7	0.6	0.98	0.99	0.99	0.99

and tuning, based on the learning of the underlying distribution of the NXDomain queries and their difference functions of malicious domains. We call this approach the **1-class learning**.

2-class learning. Alternatively, we create a model for a set of known benign domains, namely $\mathcal{B}$, where $\mathcal{B} = [b_1, \ldots, b_k]$ and assign the label of a sample x based on the following:

$$\text{Label} = \begin{cases} \text{Malicious} & : D(\mathcal{B}, f^x) > D(\mathcal{M}, f^x) \\ \text{Benign} & : D(\mathcal{B}, f^x) \leq D(\mathcal{M}, f^x) \end{cases} \quad (4)$$

We note that our scheme is less aggressive, since it prioritizes benign over malicious, as shown above. Depending on the detector objective, he might also be more aggressive by assigning a malicious label to a domain when $D(\mathcal{B}, f^x) = D(\mathcal{M}, f^x)$.

IV. EVALUATION

To evaluate the performance of our scheme, we use the dataset described in section II, with the head of the distribution of the dataset corresponding to malicious domains, and the rest of the distribution corresponding to benign domains. With labels known in advance, we proceed to evaluate the labeling capability of our scheme. For 1-class learning, and based on the distribution of the various malicious domains, we set $\Delta = 0.08$, which corresponds to 99% of detection accuracy of all the domain name samples considered and included for building the baseline model $\mathcal{M}$. To build the model $\mathcal{M}$, and to simulate a real-world scenario, we use 1,000 domains. For the unit a, we calculate the number of queries every hour, and consider a sliding window size W as 8, 16, 24, 36, and 48 hours (thus, a window of size 24 would move a step of 1 hour at a time to simulate lazy learning of a new difference vector). We start "observing" responses for each 5 days (as highlighted in our dataset) before the registration of a domain. For our evaluation, we consider a variety of evaluation metrics:

- **Standard metrics:** (i) True positives (T_p): domains correctly identified as malicious. (ii) False positives (F_p): domains incorrectly marked as malicious. (iii) True negative: (T_n) domains marked correctly an not malicious. (iv) False negative (F_n): domains incorrectly marked as not malicious. Using those outcomes, precision, recall, accuracy, and F1 score are $P = \frac{T_p}{T_p+F_p}$, $R = \frac{T_p}{T_p+F_n}$, $A = \frac{T_p+T_n}{T_p+T_n+F_p+F_n}$, and $F1 = 2 \times \frac{P \times R}{P+R}$.
- **Time:** we use how much in advance (before registration) a domain can be detected as a measure of "proactiveness".

The results for 2-class learning is shown in Table II across multiple evaluation metrics. We notice that the performance of our scheme is quite promising, especially with a limited amount of knowledge (expressed in a small window size). As for **time** as an evaluation metric, we notice that our scheme can learn with an accuracy of more than 0.90 (on average) for more than 88 hours in advance ($= 24 \times 5 - 8 - 24$) and can achieve an accuracy of more than 0.99 (on average) for more than 48 hours in advance ($= 24 \times 5 - 48 - 24$).

V. CONCLUSION

In this paper, we presented a technique for the proactive detection of algorithmically generated malicious domain names typically employed by botnets. We highlighted the fact that DGA domains tend to have a large number of DNS queries prior to registration, resulting in NXDomain responses which is then followed by a gradual overall decline. We then devised a detection algorithm using the notion of the difference function over the number of NXDomain responses for a given domain with a sliding time window. Using DNS traffic gathered from certain TLDs for the pre-calcuated list of generated domains by the Conficker malware variants, our detection algorithm was able to achieve 99% accuracy (on average) as early as 48 hours prior to registration.

Acknowledgement. This work was supported in part by NSF grant CNS-1643207, NRF grant NRF-2016K1A1A2912757. J. Kim was supported by KEPCO grant R17XA05-41 (2017)

REFERENCES

[1] —. The conficker working group. http://bit.ly/1kAYsJA, Nov 2012.
[2] M. Andrews. Negative caching of DNS queries (DNS NCACHE). RFC 2308, 1998.
[3] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *USENIX Security*, 2012.
[4] W. Chang, A. Mohaisen, A. Wang, and S. Chen. Measuring botnets in the wild: Some new trends. In *ACM ASIACCS*, pages 645–650, 2015.
[5] A. Kountouras, P. Kintis, C. Lever, Y. Chen, Y. Nadji, D. Dagon, M. Antonakakis, and R. Joffe. Enabling network security through active DNS datasets. In *RAID*, pages 188–208, 2016.
[6] C. Lever, P. Kotzias, D. Balzarotti, J. Caballero, and M. Antonakakis. A lustrum of malware network communication: Evolution and insights. In *IEEE Symposium on Security and Privacy*, pages 788–804, 2017.
[7] P. Mockapetris. Domain names: implementation and specification (november 1987). RFC 1035, 2004.
[8] A. Mohaisen and O. Alrawi. Unveiling zeus: automated classification of malware samples. In *WWW (Comp. Volume)*, pages 829–832, 2013.
[9] A. Mohaisen, O. Alrawi, and M. Mohaisen. AMAL: high-fidelity, behavior-based automated malware analysis and classification. *Computers & Security*, 52:251–266, 2015.
[10] Y. Nadji, M. Antonakakis, R. Perdisci, D. Dagon, and W. Lee. Beheading hydras: performing effective botnet takedowns. In *ACM CCS*, pages 121–132, 2013.
[11] S. Shin and G. Gu. Conficker and beyond: a large-scale empirical study. In *ACSAC*, 2010.
[12] M. Thomas and A. Mohaisen. Kindred domains: detecting and clustering botnet domains using DNS traffic. In *WWW (Companion Volume)*, pages 707–712, 2014.
[13] A. Wang, A. Mohaisen, W. Chang, and S. Chen. Delving into internet ddos attacks by botnets: Characterization and analysis. In *DSN*, 2015.
[14] A. Wang, A. Mohaisen, W. Chang, and S. Chen. Revealing ddos attack dynamics behind the scenes. In *DIMVA*, 2015.
[15] A. Wang, A. Mohaisen, and S. Chen. An adversary-centric behavior modeling of ddos attacks. In *IEEE ICDCS*, pages 1126–1136, 2017.
[16] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan. Detecting algorithmically generated malicious domain names. In *ACM IMC*, pages 48–61, 2010.