2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering

BotCensor: Detecting DGA-Based Botnet Using Two-Stage Anomaly Detection

Biao Qi*†, Jiang Jianguo*, Shi Zhixin*, Rui Mao*, WangQiwen*
*Institute of Information Engineering,
Chinese Academy of Sciences, BeiJing, China
†School of Cyber Security,
University of Chinese Academy of Sciences, Beijing, China
Email: {qibiao, jiangjianguo, shizhixin, maorui, wangqiwen}@iie.ac.cn

***Abstract*—Nowadays, most botnets utilize domain generation algorithms (DGAs) to build resilient and agile command and control (C&C) channels. Specifically, botmasters employ DGAs to dynamically produce a large number of random domains and only register a small subset for their actual C&C servers with the purpose to defend them from takeovers and blacklisting attempts. While many approaches and models have been developed to detect DGA-based botnets, they suffer from several limitations, such as difficulties of DNS traffic collection, low feasibility and scalability, and so forth. In this paper, we present BotCensor, a new system that can determine if a host is infected with certain DGA malware with two-stage anomaly detection. In the first stage, we preliminarily attempt to identify malicious domains using a Markov model, and in the second stage, we re-examine the hosts that requested aforementioned malicious domains using novelty detection algorithms. Our experimental results show that our approach performs very well on identifying previously unknown DGA-generated domains and detects DGA bots with high efficiency and efficacy. Our approach not only can be regarded as security forensics tools, but also can be used to prevent malware infections and spread.**

***Index Terms*—DGA-based botnet detection, Two-stage anomaly detection, Markov model, Novelty detection algorithms, DNS traffic;**

I. Introduction

A botnet is a network of hijacked computers and devices infected with bot malware and remotely controlled by an attacker (i.e. botmaster) via a command and control (C&C) channel [1], [2]. Compared to other malware, botnets have the distinguishing characteristic of a C&C channel [3]. A botmaster issues an order to bots and learn the conditions of them through the C&C channel. Botnets today have been a popular tool of choice for cyber-criminals to achieve their nefarious purposes such as stealing private credentials [4], sending spam emails [5], performing click-frauds [6], launching distributed denial of service (DDoS) attacks [7]. A fundamental aspect of botnets is that of coordination, in other words, how the bots find and communicate with their botmaster (or C&C server). Traditionally, IP addresses or DNS names of C&C servers or node IDs in peer-to-peer (P2P) botnets were hard-coded in bots [8]. This facilitates bots to locate C&C servers or other peers. However, such mechanisms suffer from the single point of failure problem because the C&C domains or IP addresses will be easily detected and taken down, resulting in the botmaster loses control over the entire botnet.

To overcome this problem, botnets have evolved to be more advanced over recent years. A widespread use of domain generation algorithms (DGAs) [9] has been witnessed in current botnets [10]. In DGA-based botnet, the botmaster and bots within the botnet share the same DGA. By using DGA, each bot dynamically generates a list of pseudo-random domain names and then sends DNS queries to domains on the list sequentially until one of the domains resolves to the IP address of a C&C server or terminates after a restrictive number of trials. At the same time, botmaster only needs to register several domains on the same list to serve as C&C servers. This strategy provides a prominent level of agility and viability for even if one or more domain names or IP addresses are identified and blacklisted, the bots finally can track the C&C server via DNS queries to the next set of automatically generated domains (AGDs). Some known examples of DGA-based botnets are Torpig [11], Conficker-A/B/C [12], Murofet [13], Bedep [14], Mirai [15], and so on.

Many previous works have been proposed to identify DGA-based botnets or automatically generated domains using DNS traffic. While such techniques make some contributions to mitigating botnet threats in some respects, as far as we know there are several shortcomings. First, some approaches like [16] can automatically detect AGDs and correlate them to botnets, however, they have to access large-scale Internet traffic data, which sometimes poses practical data collection and repeatability issues. Besides, it is unreasonable that they employ NXdomain [17] replies alone to infer the families of AGDs. Second, bulk of existing detection mechanisms (e.g., [18], [19]) attempt to explore malicious domains using DNS traffic, but they fail to verify the abuses of such domains. In addition, another limitation of such works is that they require labelled malicious domains as training dataset. Third, also lots of detection systems employ discriminant features that are ambiguous and not easily obtainable such as meaningful characters and substrings in domains [20] that will be troublesome to bootstrap the detection systems. Finally, some early articles detect the vast majority of AGDs by malware reverse-

The corresponding author is ShiZhixin, the email is shizhixin@iie.ac.cn.

2324-9013/18/31.00 ©2018 IEEE
DOI 10.1109/TrustCom/BigDataSE.2018.00109

engineering (e.g., [21], [22]). Albeit reverse engineering bots is a very effective means to pre-compute current and future candidate C&C domains generated by bots and later block or sinkhole them so that security personnel can drastically take down the entire botnet, it is labor-intensive and time-consuming process.

In this paper, we propose a novel detection system, called BotCensor, to identify DGA-based bots within a monitored network with two-stage anomaly detection. Specifically, in the first stage, BotCensor preliminarily utilizes a Markov model to discover malicious domains and afterwards re-examines the hosts that sends DNS queries to such malicious domains with fine-grained anomaly detection in the second stage. The first anomaly detection stage plays a role in moderating the complexity of computation tasks in the whole DGA-based bots detection procedure. And in the second anomaly detection stage, BotCensor extracts three fine-grained distinguishing features, i.e., the DNS successful responses to queries ratio, the entropy of the number of DNS queries, and the entropy of DNS request type. Afterwards, BotCensor employs three novelty detection algorithms to model on only legitimate DNS traffic and then predict unknown DGA-bots.

In summary, this paper makes the following contributions:

- We propose BotCensor, a lightweight DGA-based botnet detection system that only utilize legitimate DNS traffic in a monitored network to automatically detect DGA-based bots. Specifically, botCensor employs a two-stage anomaly detection skill to accomplish the detection goal.
- We implemented a prototype version of BotCensor, and evaluated its detection performance with various datasets such as public Alexa domains, malicious domains and DNS traffic collected in controlled environment. Experimental results demonstrate that BotCensor can accurately detect DGA-bots.
- Last but not the least, we offer a new DGA-based botnet detection idea. To best of our knowledge, BotCensor is the first prototype using two-stage anomaly detection to discovery DGA-bots. On the one hand, BotCensor can detect previously unknown AGDs without malware reverse-engineering, on the other hand, BotCensor scales very well and can be used as a practical security forensics tool.

The remainder of this work is organized as follows. We introduce the background of DGA-based botnet in section 2. We present our system design and implementation in section 3. We analyze collected datasets and conduct numerous experiments to validate the proposed approach in section 4. We present our experimental results and analyses in section 5. We survey related work in section 6 and finally conclude the paper in section 7.

II. Background

The Domain Name System (DNS) service is critical for the normal functioning of almost all Internet services. DNS plays a vital role in translating human-readable domain names into machine-level IP addresses or vise versa. DNS utilizes the hierarchical structure to efficiently cache the copies of name-IP associations, which can guarantee that domain name resolution is fast, reliable and highly distributed. The domain name resolution has many advantages such as load balancing and failover restarts. Unfortunately, the agility of DNS is also abused by attackers to operate their botnets. Concretely, Botmasters make full use of domain fluxing [23] and fast fluxing [24] to make their botnets more resilient and, more importantly, harder to be taken down. In this paper, we mainly focus on domain fluxing botnets. In domain flux botnet infrastructures, botnet spews out a large number of automatically generated domains, and botmaster simply needs to register one or several domains beforehand. Bots will query the generated domain list until an IP address of C&C server returns or terminates under the restrictive condition.

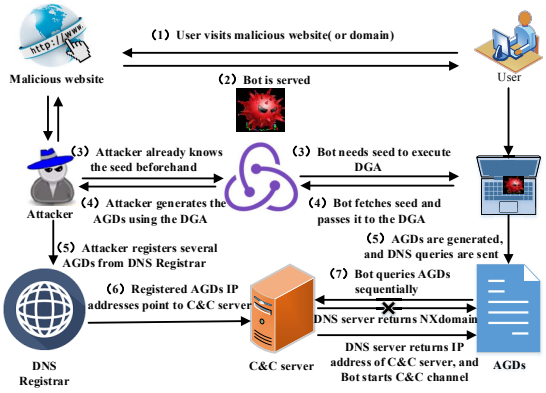


Fig. 1: DGA-based botnet communication mechanism

Figure 1 shows DGA-based botnet communication mechanism. It needs to be emphasized that in step 7, when bots query AGDs sequentially, local DNS server or recursive DNS server will return a failing NXdomain reply [25] (under the circumstance that the queried domain name for which no IP addresses exist) or a successful IP address of the C&C server response [1].

For inherent regularity and automation, bots will sent DNS queries according to the AGDs list continually until a successful reply returns or aborts. During the query loop, some discriminate characteristics that differ from legitimate hosts emerge. First, vast amount of unsuccessful DNS resolutions i.e., NXdomains appear in the DNS traffic regularly. Second, bots query domain names in a regular pattern, for example, we look over DNS traffic generated by Zeus [26] to discover that Zeus almost sends a DNS request every five seconds. Third, the DNS request type of bots is very single for bots just want to get the IP address of the C&C server. Compared to bots, legitimate users visit websites more purposely. Specifically, if a user visits a website, he or her will linger here for some time and then browse the next link, so that DNS request behavior of benign users are very uncertain. Consequently,

[1]It may also return a CNAME record, it means that the queried domain has a canonical name.

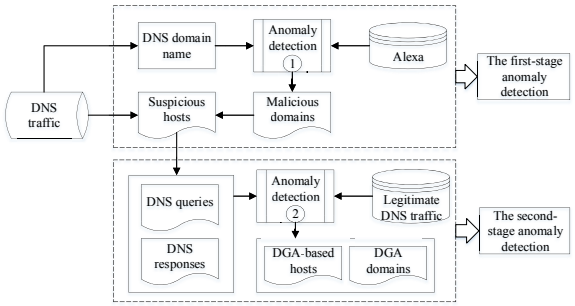


Fig. 2: Architecture of BotCensor

we can conclude: (1) DNS queries of benign users will not produce much NXdomains (certainly NXdomains may be accidentally generated by typos or mis-configurations), (2) the number of DNS queries of benign users is not permanent, and (3) the DNS request type of benign users is various. In a word, BotCensor exploits the aforementioned distinguishing attributes to identify DGA-based bots.

III. System design and implementation

In this section, we provide a high-level overview of our DGA-bot detection system BotCensor. As shown in Figure 2, BotCensor consists of two anomaly detection modules. In the first anomaly detection, BotCensor mainly focuses on identifying malicious domain names from DNS traffic collected at the network edge. In the second anomaly detection, BotCensor emphatically takes hosts that sent DNS queries to the aforementioned malicious domain names into account. And the first anomaly detection can contribute to reducing the computation complexity of the whole detection procedure.

A. The First-Stage Anomaly Detection

The first-stage anomaly detection mainly analyzes domain names extracted from DNS traffic as seen from the above layer in Figure 2. Concretely, BotCensor employs a Markov model [27] to model on legitimate domain data i.e., Alexa [2], and then judges whether a domain name is malicious or not.

A Markov model is a stochastic method for randomly changing systems where it is assumed that future states do not depend on past states. These models show all possible states as well as the transitions, rate of transitions and probabilities between them. Markov models are often used to model the probabilities of different states and the rates of transitions among them. The method is generally used to model systems such as speech systems. Markov models can also be used to recognize patterns, make predictions and to learn the statistics of sequential data.

In brief, a Markov model consists of three tuples, i.e., $\mathcal{M} = (S, \pi, A)$, where S represents a finite number of states $S = \{s_1, s_2, \ldots, s_N\}$, π is an initial state vector, and A denotes a transition matrix whose element a_{ij} is referred as state transition probabilities, and $a_{ij} = P(x_t = S_j | x_{t-1} = S_i), 1 \leq i, j \leq N$, such that $a_{ij} \geq 0, \sum_{j=1}^{N} a_{ij} = 1$.

Given a sequence of states $\mathcal{X} = \{s_1, s_2, \ldots, s_K\}$, the likelihood can be written as the following formulas:

$$P(\mathcal{X}|\mathcal{M}) = P(x_K, x_{K-1}, \ldots, x_1) \tag{1}$$

$$= P(x_K|x_{K-1}, \ldots, x_2, x_1) \cdots P(x_2|x_1)P(x_1) \tag{2}$$

$$= P(x_1) \prod_{n=2}^{K} P(x_n|x_{1:n-1}) \tag{3}$$

To simplify the above calculation a Markov assumption is applied

$$P(x_n|x_{n-1}, \cdots, x_2, x_1) \approx P(x_n|x_{n-1}) \tag{4}$$

This approximation is called the First-order Markov assumption because the outcome of x_n is only dependent on the outcome at x_{n-1}. The assumption means that Equation 3 can be written as a product of conditions on the previous sample

$$P(\mathcal{X}|\mathcal{M}) = P(x_1) \prod_{n=2}^{K} P(x_n|x_{n-1}) \tag{5}$$

For convenience we exploit the First-order Markov chain to model our first-stage anomaly detection.

It is well known that a domain name d consists of a set of labels separated by dots, for example, *www.domain.com*. The rightmost label is called the top-level domain (TLD), e.g., com, net, org. The second-level domain (2LD) contains the two rightmost labels separated by a dot. The third-level domain (3LD) contains the three rightmost labels, i.e., *www.domain.com*, etc. Note that we only consider the first level of a chosen prefix such as *domain.com* referring to [28]. Literally, a domain actually consists of 26 English letters, 10 numbers, '-' and a period. Hence, a shining and intuitive idea is that we can regard 38 elements (i.e., 26 English letters, 10 numbers, '-' and a period) as 38 states, and the transition matrix A (as shown in Equation 6) contains the transition probabilities of any two states.

$$A = [a_{ij}] = \begin{array}{c} \\ a \\ 0 \\ \vdots \\ . \end{array} \begin{array}{c} \begin{array}{cccc} a & 0 & \cdots & . \end{array} \\ \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{138} \\ a_{21} & a_{22} & \cdots & a_{238} \\ \vdots & \vdots & \cdots & \vdots \\ a_{381} & a_{382} & \cdots & a_{3838} \end{pmatrix} \end{array} \tag{6}$$

Therefore the probability of any domain can be computed according to the Equation 5. It makes sense to do so. Generally speaking, a humanly generated domain is user-friendly and easy to remember, but an AGD is not. Hence, the larger the probability of a domain name, the stronger its legitimacy.

B. The Second-Stage Anomaly Detection

As shown in the lower part of the Figure 2, the second-stage anomaly detection mainly focuses on differentiating DGA-bots from legitimate hosts. Due to the use of DNS traffic, we consider the following three observations.

[2]Alexa. http://www.alexa.com/topsites/

TABLE I: Comparisons between DGA-bots and legitimate hosts in the three features

Feature	DGA-bots	Legitimate hosts
$\pi_{Res/Req}$	small	large
ξ_q	large	small
$\delta_{\triangle t}$	small	large

- Most of the DGA-generated domains that a bot queries would result in NXdomains responses, hence successfully DNS replies to DNS queries ratio of DGA-bots is smaller than that of legitimate hosts.
- The number of DNS queries sent by DGA-bots is different from that by legitimate users in confined time. Besides, DGA-bots regularly request AGDs e.g., Zeus bots sent a DNS request every five seconds, but in contrast the randomness of legitimate users query a DNS is more strong.
- The type of DNS request of DGA-bots is usually homogeneous for bots simply want to get the IP addresses of the C&C server. Compared to bots, legitimate users will visit a plenty variety of websites, and duo to the nature of websites, type of DNS requests of legitimate users will be various.

Based on the above considerations, we extract three discriminant features from DNS traffic. We elaborate them as the follows:

Feature 1: the rate of successful DNS responses to DNS queries within limited time. For the above feature, we give the formal definition:

$$\pi_{Res/Req} = \frac{\pi_{Res}}{\pi_{Req}} \tag{7}$$

Where π_{Res} represents the number of successful DNS responses, and π_{Req} denotes the number of DNS requests.

Feature 2: the entropy of the number of DNS queries within limited time. The feature has the following definition:

$$\xi_q = -\sum_{i=1}^{N}(\frac{q_i}{\sum_{i=1}^{N} q_i} log \frac{q_i}{\sum_{i=1}^{N} q_i}) \tag{8}$$

Where q_i denotes the number of DNS queries in the ith time interval, N denotes N time intervals.

Feature 3: the entropy of the type of DNS queries within limited time. Given a period of time, we have the definition:

$$\delta_{\triangle t} = -\sum_{i=1}^{M}(\frac{s_i}{\sum_{i=1}^{M} s_i} log \frac{s_i}{\sum_{i=1}^{M} s_i}) \tag{9}$$

Where $\triangle t$ denotes the given limited time, s_i denotes the number of the ith DNS request type such as A, AAAA, etc in $\triangle t$, and M denotes the total number of DNS request types, in our work M=9.

According to the above description and considerations, we can draw the obvious conclusions as shown in the Table I.

For the convenience of calculation, we perform the operations of fine-grained segmentation according to timestamps on DNS data. For example, we treat DNS traffic of a host as a time series, we partition DNS data into many parts according to timestamps, and each part has the same period e.g., 5s, and then we extract the three above-mentioned features from each part.

Note that if we set the time period as 5s, $\pi_{Res/Req}$ in **Feature 1** denotes the rate of successful DNS responses to DNS queries within 5 seconds, q_i in **Feature 2** denotes the number of DNS queries in the ith second, and naturally the N is 5, and s_i in **Feature 3** denotes the number of the ith DNS request type in 5 seconds, $\triangle t$ is 5.

After the feature extraction, we exploit three novelty detection algorithms (i.e., One-Class SVM with non-linear kernel (RBF) [29], Isolation Forest [30], Multivariate Gaussian [31]) to identify the abnormal hosts i.e., DGA-bots.

IV. Experiments

In this section, we empirically evaluate the performance of BotCensor. To validate BotCensor, we conduct a comprehensive measurement study using both several public source data and real DNS traces. Concretely, in the first anomaly detection process, we exploit malicious domain data and top 1-million static domains by Alexa to verify BotCensor's ability to identify a malicious domain. And in the second anomaly detection process, we use DNS traces to measure the efficacy of BotCensor to recognize DGA-bots.

A. Evaluation Dataset

The first-stage anomaly detection of BotCensor actually resembles a domain reputation system that can tell whether a domain is generally considered as malicious. Hence BotCensor requires a mass of legitimate domain data as training dataset and malicious domain data as testing dataset in the first anomaly detection process. Here we collect top 1-million static domains by Alexa as domain whitelist benchmark which can be downloaded in [3]. Besides, we acquire malicious domains to validate the anomaly detector, specifically, we crawl accepted malicious domains from [32], [33], [34] where include various of malicious infrastructures such as ZeuS C&Cs, phishing sites, trojan downloading, AGDs, etc. Note that there may be some malware domains in the Alexa top 1 million lists, so we choose the top 100,000 domains as our final legitimate domain benchmark. Moreover, we sort out 50,000 malicious domains as the testing dataset to verify BotCensor's detection capability.

The main goal of the second-stage anomaly detection of BotCensor is to judge whether a host is infected by DGA bots. To this end, we collected both DNS traces of legitimate hosts and malicious DGA-bots to carry out the validation. Specifically, in the second anomaly detection process, BotCensor takes as input pure legitimate DNS traffic to model the behaviors of legitimate users. For this purpose we captured legitimate campus DNS traffic at the edge link in our research institute for 72 hours. After pretreatment, we obtained a total of about

[3]http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

TABLE II: Malware DNS traces used in the evaluation

DNS traces	#DNS requests	Type	During
Zeus	5717	Trojan.WLDCR.C	3hours and 48minutes
Cridex	4736	W32.Cridex	3hours and 7minutes
Duqu2	4660	W32.Duqu.B	3hours and 5minutes
Dyre	1016	Trojan.Dyreza.Gen.3	24minutes
Kazy	1694	Gen:Variant.Razy.127505	59minutes
Kelihos	2942	Trojan.VIZ.Gen.1	1hours and 47minutes
Locky	940	Gen:Variant.Ransom.Blocker.3	26minutes
Matsnu	706	Gen:Variant.Symmi.23512	23minutes
Sinowal	2830	Gen:Variant.Zusy.61111	1hours and 49minutes
ZeroAccess	1040	Gen:Variant.Sirefef.443	32minutes

33,730 DNS requests. We can ensure that DNS requests we collected are benign for two reasons, the one is the sites students and staffs visited are all legal, the other one is security protection measures have been taken to guard our network environment.

To verify the detection functions of BotCensor in the second anomaly detection, we also captured ten DGA-bots DNS traces in controlled environment (i.e., VMware Station) using a DNS traffic capture tool called DNSQuerySniffer[4]. And the characteristics of captured malware DNS traces are summarized in Table II. After we preprocess each trace such as removing noise DNS packets, filtering whitelist, we obtain a number of DNS requests for each botnet as shown in the second column in Table II. The third column represents the output result of VirusTotal [5] about each malware that generates the DGA-botnet. Moreover, capture duration for each DGA-botnet is shown in the last column. Note that ten DGA-botnets DNS traces are generated by malware that come from different families, the behaviors between the ten DGA-botnets may be different. So ten DGA-botnets DNS traffic can represent ten different kinds of DGAs.

B. Performance metric

Essentially we complete twice novelty detection tasks in our experimental setup. BotCensor takes as input legitimate data (i.e., legitimate domains and legitimate DNS requests) to discover and isolate abnormalities. So the performance criteria for our scenario should be specified. First, we exploit detection rate (DR) to evaluate the abnormal detection capability of BotCensor, second, we use false positive rate (FPR) to measure the false alarm level of BotCensor. Note that the above two metrics apply to both anomaly detection processes. Specifically, for example, in the first-stage anomaly detection process, a DR represents how many malicious domains can be detected correctly, and a FPR represents how often a legitimate domain is falsely decided as malicious case. It is the same way in the second-stage anomaly detection process. Naturally, a high DR and a low FPR are the desirable outcomes.

We implemented our detection procedure in Python 3.6 and employed the popular scikit-learn machine learning framework and libraries [35] for our novelty detection in the second anomaly detection process. To void bias, in both detection procedures, we randomly extract four fifths of legitimate data as training set, and the remaining are used for testing FPR. Besides, all malicious data are used to verify DR. This process is repeated 10 times, and we take the mean of 10 times test outcomes as the final result.

[4]https://www.nirsoft.net/utils/dns_query_sniffer.html

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

TABLE III: Experimental results of the first-stage anomaly detection

Strategy	DR(%)	FPR(%)
max	100.00	12.76
min	72.54	0.00
weight	99.90	4.93
upper_quartile	95.39	2.75
lower_quartile	79.83	0.04

V. Experimental results and analyses

A. Experimental results of the first-stage anomaly detection

In the first-stage anomaly detection process, we utilize a Markov model to identify malicious domains. Specifically, we use four fifths legitimate domains (i.e., 80,000) derived from Alexa top 100,000 domains to build transition matrix according to formula 6, and then we can compute the probability of any domain. Here we need a means to determine the probability threshold that can decide whether a domain is legitimate or not. To this end, we design five strategies (as shown in formula 10-14) to set the threshold. Note that the probability threshold is actually the trick combination of N probabilities of training domains that computed with formula 5.

$$max = max_i^N\{P_i\} \quad (10)$$

$$min = min_i^N\{P_i\} \quad (11)$$

$$weight = \frac{\sum_{n=1}^{N} lenght(i) * P_i}{\sum_{n=1}^{N} lenght(i)} \quad (12)$$

$$upper\_quartile = quantile(\{P_i\}, \sigma = 0.75) \quad (13)$$

$$lower\_quartile = quantile(\{P_i\}, \sigma = 0.25) \quad (14)$$

Table III summarizes the experimental results of the five strategies in the first-stage anomaly detection. From Table III, we can draw the conclusion that on the one hand, if the threshold value produced by certain strategy is too large, both

TABLE IV: Experimental results DR (%) of the second-stage anomaly detection

Approach	Zeus	Cridex	Duqu2	Dyre	Kazy	Kehihos	Locky	Matsnu	Sinowal	Zero Access
One-class SVM	99.46	99.55	99.67	99.23	99.46	99.56	99.11	99.11	99.46	99.40
Isolation Forest	99.74	99.82	99.80	99.85	99.85	99.89	100	99.89	99.78	99.40
Multivariate Gaussian	99.40	99.64	99.88	99.67	99.68	99.64	99.89	99.11	99.11	99.67

TABLE V: Experimental results FPR (%) of the first-stage anomaly detection

Approach	FPR(%)
One-class SVM	1.86
Isolation Forest	6.09
Multivariate Gaussian	4.28

DR and FPR will be high, on the other hand, if the threshold is too small, both DR and FPR will be low. So we require to choose a proper strategy satisfying a high DR and a low FPR.

B. Experimental results of the second-stage anomaly detection

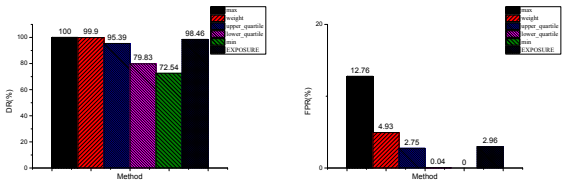
Table IV shows the experimental results DR of the second-stage anomaly detection. From Table IV, we can see that all novelty detection algorithms perform very well in terms of DR. Moreover the performance of Isolation Forest is better than other approaches. Isolation Forest based approach detects almost the mean of 99.80% malicious DGA-based DNS traffic across the all datasets. In addition to the results of DR, we also evaluate the false alarm rates of the three approaches. Table V summarizes the experimental results FPR of the second-stage anomaly detection. One-class SVM performs best as the anomaly detection approach used in terms of FPR as shown in Table V. From the Table V, we can draw the conclusion that One-class SVM based approach achieves the lowest FPR i.e., 1.86%.

In total, our evaluation results demonstrate that BotCensor can achieve wonderful detection rate with an acceptable false positive rates.

C. Comparisons with State of the Art

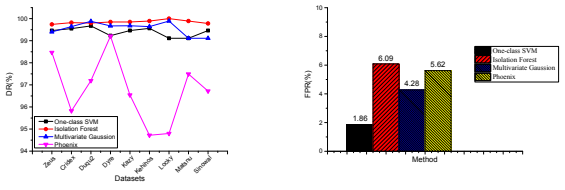
Next we compare our approach with the state of the art. Note that our comparisons to previous researches are twofold. First, we compare BotCensor with other excellent malicious domains detection methods in the first-stage anomaly detection, and second we compare BotCensor with current DGAs detection frameworks in the second-stage anomaly detection.

1) Comparison with other malicious domains detection methods: Despite detecting malicious domains is an age-old problem in security community, it's still a hot topic. Even so, to best of our knowledge, EXPOSURE [18] is a milestone work so far. Therefore we mainly conduct experimental comparisons with EXPOSURE. Figure 3 shows the comparisons of effectiveness across our approach and EXPOSURE. Concretely, DR comparison and FPR comparison are depicted in Figure 3(a) and Figure 3(b) respectively. From Figure 3(a), we can see that the DRs of our max and weight strategies are higher than that of EXPOSURE, and the FPRs of our low_quartile and min strategies are lower than that of EXPOSURE as seen in Figure 3(b). In general, the performance of our approach is slightly less excellent than that of EXPOSURE, however, our method has obvious advantages such as not requiring labor-consuming and boring feature engineering. Specifically, EXPOSURE required to extract 15 features (i.e., 4 time-based features, 4 DNS answer-based features, 5 TTL value-based features and 2 domain name-based features) for DNS traffic to feed a J48 decision algorithm. From this point, compared to EXPOSURE, BotCensor only needs to gain the linguistic information of a domain name without any other feature extraction operations.



(a) (b)

Fig. 3: Comparison between EXPOSURE and our approach in DR and FPR



(a) (b)

Fig. 4: Comparison between Phoenix and our approach in DR and FPR

Another popular solution to malicious domains detection is to model the detection problem as a graph analysis problem [36], [37], [38]. The main idea of these proposals is that they discover and analyze the global associations among domains, especially the relationship between hosts and domains, and employ the associated historical data to construct machine-domain behavior graph, afterwards some graph mining algorithms are utilized to search suspicious malware domains.

We think there are two shortcomings in the above-mentioned approaches, the one is difficult to acquire training data, specifically, like Segugio [36] requires to know which machines are malicious or legitimate beforehand, so that it confirms the legality or malice of the unknown domains. The another one is malicious domains may be queried by legitimate hosts, similarly malware-infected hosts may sent DNS requests to legitimate domains, thus graph pruning (de-noising) is also a complex task. Compared to these works, our approach doesn't require other information except textual information of the tested domains.

2) Comparison with other DGAs detection methods: An interesting research on DGAs is Phoenix [20]. It not only focused on detecting DGA-domains, but also it extracted the fingerprints of the DGAs as the intelligence to facilitate tracking and characterizing botnets. Hence the DGA discovery module was particularly important, and anomaly detection technique was also exploited to identify DGAs. Based on this, we compare our approach with Phoenix experimentally. Comparison between Phoenix and our approach in DR and FPR is depicted in Figure 4(a) and Figure 4(b) respectively. From Figure 4(a), we can see that three anomaly detection algorithms used in our approach have significant advantage over Phoenix in DR, and the FPRs of One-class SVM and Multivariate Gaussion are lower than that of Phoenix as shown in Figure 4(b). An interesting phenomenon we found is that the DRs of Phoenix are changeable for our datasets as seen in Figure 4(a). We speculate the reason may be that merely linguistic features fail to distinguish DGA-domains from legitimate ones, for instance, some DGA-domains may have pronounceable fields or large meaningful characters ratios for the sake of avoiding detection.

Pleiades [16] first exploited NXDomains clustering to automatically identify DGA-bots. However, we have no idea why Pleiades needs a DGA classification module, and it is unreasonable to assign the subset of NXDomains to a specific DGA label for NXDomains may be produced by unknown DGAs. Compared to Pleiades, our approach just safely detects DGAs with anomaly detection.

Besides there are also works such as [39], [40] relying on supervised learning to classify DGAs. Our work is different from these approaches because we require no labeled datasets of AGDs to be bootstrapped, and we can identify unknown AGDs.

D. Discussions

Our experimental results demonstrate the feasibility and effectiveness of the proposed method in automatically identifying DGA-based botnets. We think it might be caused by the following reasons.

First, in the first-stage anomaly detection, a Markov model is used to detect malicious domains. Ultimately, the probabilities of most malicious domains are less than those of legitimate domains. The reasons are intuitive, the one is some malicious domains are very long, resulting in the probabilities will be small (cf. 5), the other one is some infrequent two-adjacent characters appear in malicious domains, and the values of these in transition matrix A (cf. 6) are very little, leading to the probabilities of malicious domains will be also small (cf. 5). Second, in the second-stage anomaly detection, we extract three extraordinarily discriminant features (cf. 7-9) from DNS traffic. Vast of experiments verify the effectiveness of our method.

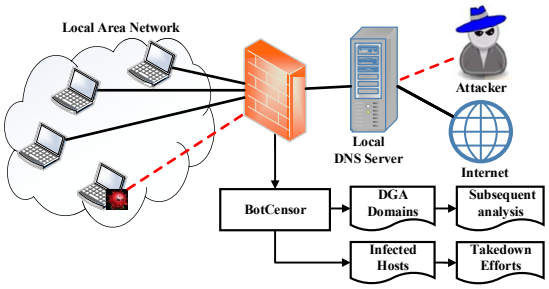


Fig. 5: BotCensor deployment overview

Note that the DR of our approach is satisfactory, however, the FPR is a little bad, we believe that such huge difference between legitimate DNS traffic and DGA-based traffic makes it easier to distinguish one of them from another. Furthermore, the randomness of behaviors of legitimate users is too strong to completely differentiate from themselves.

E. BotCensor deployment environment

Figure 5 shows the BotCensor deployment overview. BotCensor can be deployed at the edge of a local area network practically. On the one hand, BotCensor enables security administrators to discover DGA domain names, and conduct subsequent further analysis; on the other hand, DGA-based bots i.e., infected hosts can be found and monitored with BotCensor, and takedown efforts could be performed.

Besides, from the perspective of practical operations, BotCensor can also be regarded as an effective security forensics tool that focuses on inspecting single hosts in small monitored network.

F. Limitations

Despite the nice results, BotCensor has some limitations. First, if an attacker knows the rationale of the first-stage anomaly detection of BotCensor, he or her may use more domains that are similar to legitimate ones as DNS mapping objects. However, if so, domain name space that an attacker can choose will be greatly reduced. Second, to escape the detection of BotCensor, an attacker may design a new DGA-botnet and the regularity of DNS requests by bots becomes week. Concretely, bots randomly sent DNS queries to outside, in this way, BotCensor may falsely determine a bot as a legitimate host. But these botnets lack stability and reliability, and the botmaster may lose control of the botnets intermittently. Third, despite BotCensor can detect unknown DGA, it is unable to learn or reconstruct the exact DGA, from this point, the limitation of BotCensor is similar to other literatures [16], [20], [41]. This puzzle seems to be very

challenging. Finally, BotCensor fails to speculate and assess the population distribution of DGA-bots. This issue may be our future research direction.

VI. Related work

Botnet confrontation is a broad and continuous topic in security community. Surveys we can refer to on botnet phenomenon, botnet detection, and botnet mitigation include [42], [43], [44], [45]. Because there are quite a large number of researches on this area, we just enumerate the representative related works. Yadav et al. [23] proposed a method to discover DGA-domains by leveraging the fact that randomization of AGDs is more strong than that of human generated domains (HGDs) to distinguish them from HGDs. Antonakakis et al. [16] presented a detection system i.e., Pleiades that leveraged NXDomains clustering to identify botnets members that ran the same DGA malware. Sharifnya et al. [46] proposed a reputation system to detect DGA-based botnets by automatically assigning a high negative reputation score to hosts that were involved in suspicious bot activities. Schiavoni et al. [20] only leveraged linguistic features to identify DGA domain names. Haq et al. [47] proposed a system that generated network fingerprints of DGA-bots to automatically detect its DGAs. Grill et al. [48] proposed a statistical approach and model the ratio of DNS requests and visited IPs for every host, and label the deviations from this model as DGA-bots. Thomas et al. [49] analyzed NXDomains at several TLD authoritative name servers to identify groups of malicious domains sharing similar lookup patterns. Woodbridge et al. [40] leveraged deep learning to build a DGA classifier for real-time prediction of DGAs without the need for contextual information or manually created features.

There exist also other interesting works focusing on DGA analysis. Daniel Plohmann et al. [10] performed a comprehensive measurement study of DGAs by analyzing 43 DGA-based malware families and variants. Aditya K. Sood et al [9] introduced a detailed taxonamy of DGAs and facilitated us to understand various attack techniques and their potential trends. Yu Fu et al. [50] designed two new DGAs that could escape the detection of several DGA detection systems. Hyrum S. Anderson et al. [51] leveraged the concept of generative adversarial networks to construct a deep learning based DGA that is designed to intentionally bypass a deep learning based detector. Ting Wang et al. [52] presented a novel tool called BotMeter that accurately assess the DGA-bot population landscapes in large networks by analyzing DNS traffic.

VII. Conclusion

In this paper, we propose a system called BotCensor that effectively and efficiently detect unknown DGA-based botnets with two-stage anomaly detection without the need for malware reverse-engineering. Specifically, in the first-stage anomaly detection, we preliminarily exploit the Markov model to identify malicious domain names, and then we re-examine the hosts that requested aforementioned malicious domains. And the first anomaly detection stage plays a role in reducing the complexity of computation tasks in the whole DGA-based botnets detection procedure. Note that in the second-stage anomaly detection process, we extract three fine-grained discriminant features from DNS traffic, and then three novelty detection algorithms are utilized to discern the DGA-based traffic from legitimate traffic. To validate BotCensor, a large number of experiments are conducted on several public datasets and private DNS traffic. Our evaluation results and experimental comparisons with previous related works reveal that our method performs some advantages over existing approaches. From a practical point of view, we believe that our approach not only can be regarded as a security forensics tool, but also can be used to prevent malware infections and spread.

VIII. Acknowledgement

We thank our anonymous reviewers for their invaluable feedback. This material is based upon work supported in part by the National Key Research and Development Program of China grant no. 2016YFB0801001 and 2016YFB0801004.

References

[1] Z. Zhu, G. Lu, Y. Chen, Z. J. Fu, P. Roberts, and K. Han, "Botnet research survey," in *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*. IEEE, 2008, pp. 967–972.

[2] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on*. IEEE, 2009, pp. 268–273.

[3] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," 2008.

[4] S. Saroiu, S. D. Gribble, and H. M. Levy, "Measurement and analysis of spyware in a university environment." in *NSDI*, 2004, pp. 141–153.

[5] J. P. John, A. Moshchuk, S. D. Gribble, A. Krishnamurthy *et al.*, "Studying spamming botnets using botlab." in *NSDI*, vol. 9, 2009, pp. 291–306.

[6] H. Haddadi, "Fighting online click-fraud using bluff ads," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 21–25, 2010.

[7] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *USENIX Security Symposium*, 2017.

[8] F. C. Freiling, T. Holz, and G. Wicherski, "Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks," in *European Symposium on Research in Computer Security*. Springer, 2005, pp. 319–335.

[9] A. K. Sood and S. Zeadally, "A taxonomy of domain-generation algorithms," *IEEE Security & Privacy*, vol. 14, no. 4, pp. 46–53, 2016.

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

[11] B. Stone-Gross, M. Cova, B. Gilbert, R. Kemmerer, C. Kruegel, and G. Vigna, "Analysis of a botnet takeover," *IEEE Security & Privacy*, vol. 9, no. 1, pp. 64–72, 2011.

[12] S. Shin and G. Gu, "Conficker and beyond: a large-scale empirical study," in *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 2010, pp. 151–160.

[13] S. Shevchenko, "Domain name generator for murofet," *threatexpert.com*, 2010.

[14] D. Schwarz, "Bedeps dga: Trading foreign exchange for malware domains," 2016.

[15] I. Van der Elzen and J. van Heugten, "Techniques for detecting compromised iot devices," *University of Amsterdam*, 2017.

[16] 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 symposium*, vol. 12, 2012.

[17] B. Ager, G. Smaragdakis, and S. Uhlig, "Comparing dns resolvers in the wild," in *ACM SIGCOMM Conference on Internet Measurement 2010, Melbourne, Australia - November*, 2010, pp. 15–21.

[18] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis." in *Ndss*, 2011.

[19] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, "Detecting malware domains at the upper dns hierarchy." in *USENIX security symposium*, vol. 11, 2011, pp. 1–16.

[20] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: Dga-based botnet tracking and intelligence," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2014, pp. 192–211.

[21] M. Sharif, A. Lanzi, J. Giffin, and W. Lee, "Automatic reverse engineering of malware emulators," in *Security and Privacy, 2009 30th IEEE Symposium on*. IEEE, 2009, pp. 94–109.

[22] T. Barabosch, A. Wichmann, F. Leder, and E. Gerhards-Padilla, "Automatic extraction of domain name generation algorithms from current malware," in *Proc. NATO Symposium IST-111 on Information Assurance and Cyber Defense, Koblenz, Germany*, 2012.

[23] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *ACM SIGCOMM Conference on Internet Measurement 2010, Melbourne, Australia - November*, 2010, pp. 48–61.

[24] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, Usa, February - February*, 2008, pp. 487 – 492.

[25] M. Karir, M. Karir, and Z. M. Mao, "Characterizing dark dns behavior," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2007, pp. 140–156.

[26] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: Mining for unwanted p2p traffic," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2013, pp. 62–82.

[27] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc of the IEEE*, vol. 77, no. 2, pp. 267–296, 1990.

[28] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Tracking and characterizing botnets using automatically generated domains," *Computer Science*, vol. cmp-lg/9503011, no. 2, pp. pgs. 217–248, 2013.

[29] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of Machine Learning Research*, vol. 2, no. 1, pp. 139–154, 2002.

[30] F. T. Liu, M. T. Kai, and Z. H. Zhou, "Isolation forest," in *Eighth IEEE International Conference on Data Mining*, 2009, pp. 413–422.

[31] G. G. Hazel, "Multivariate gaussian mrf for multispectral scene segmentation and anomaly detection," *IEEE Transactions on Geoscience & Remote Sensing*, vol. 38, no. 3, pp. 1199–1211, 2000.

[32] "Abuse.ch," https://abuse.ch/.

[33] "Malwaredomainlist.com," https://www.malwaredomainlist.com/mdl.php.

[34] "Dga domain feeds," http://osint.bambenekconsulting.com/.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] B. Rahbarinia, R. Perdisci, and M. Antonakakis, "Segugio: Efficient behavior-based tracking of malware-control domains in large isp networks," in *Ieee/ifip International Conference on Dependable Systems and Networks*, 2015, pp. 403–414.

[37] S. Yadav, S. Yadav, P. Rao, and W. Horne, "Detecting malicious domains via graph inference," in *European Symposium on Research in Computer Security*, 2014, pp. 59–60.

[38] I. Khalil, T. Yu, and B. Guan, "Discovering malicious domains through passive dns data graph analysis," in *ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 663–674.

[39] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with dns traffic analysis," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1663–1677, 2012.

[40] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," 2016.

[41] T. S. Wang, H. T. Lin, W. T. Cheng, and C. Y. Chen, "Dbod: clustering and detecting dga-based botnets using dns traffic analysis," *Computers & Security*, vol. 64, pp. 1–15, 2016.

[42] M. Bailey, E. Cooke, F. Jahanian, Y. Xu, and M. Karir, *A Survey of Botnet Technology and Defenses*, 2009.

[43] R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks the International Journal of Computer & Telecommunications Networking*, vol. 57, no. 2, pp. 378–403, 2013.

[44] N. Goodman, "A survey of advances in botnet technologies," 2017.

[45] G. Vormayr, T. Zseby, and J. Fabini, "Botnet communication patterns," *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, pp. 1–1.

[46] R. Sharifnya and M. Abadi, "A novel reputation system to detect dga-based botnets," in *International Econference on Computer and Knowledge Engineering*, 2013, pp. 417–423.

[47] O. Haq, W. Ahmed, and A. A. Syed, "Titan: Enabling low overhead and multi-faceted network fingerprinting of a bot," in *Ieee/ifip International Conference on Dependable Systems and Networks*, 2014, pp. 37–44.

[48] M. Grill, I. Nikolaev, V. Valeros, and M. Rehak, "Detecting dga malware using netflow," in *Ifip/ieee International Symposium on Integrated Network Management*, 2015, pp. 1304–1309.

[49] A. Mohaisen and A. Mohaisen, "Kindred domains: detecting and clustering botnet domains using dns traffic," in *Companion Publication of the International Conference on World Wide Web Companion*, 2014, pp. 707–712.

[50] Y. Fu, L. Yu, O. Hambolu, I. Ozcelik, B. Husain, J. Sun, K. Sapra, D. Du, C. Beasley, and R. Brooks, "Stealthy domain generation algorithms (dgas)," *IEEE Transactions on Information Forensics & Security*, vol. PP, no. 99, pp. 1–1, 2017.

[51] H. S. Anderson, J. Woodbridge, and B. Filar, "Deepdga: Adversarially-tuned domain generation and detection," pp. 13–21, 2016.

[52] T. Wang, X. Hu, J. Jang, S. Ji, M. Stoecklin, and T. Taylor, "Botmeter: Charting dga-botnet landscapes in large networks," in *IEEE International Conference on Distributed Computing Systems*, 2016, pp. 334–343.