Computer Networks 56 (2012) 20–33

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Identifying botnets by capturing group activities in DNS traffic

Hyunsang Choi, Heejo Lee *

Department of Computer Science and Engineering, Korea University, Seoul 136-713, Republic of Korea

ARTICLE INFO

Article history:
Received 13 October 2010
Received in revised form 21 June 2011
Accepted 22 July 2011
Available online 30 July 2011

Keywords:
Botnet
Group activity
DNS

ABSTRACT

Botnets have become the main vehicle to conduct online crimes such as DDoS, spam, phishing and identity theft. Even though numerous efforts have been directed towards detection of botnets, evolving evasion techniques easily thwart detection. Moreover, existing approaches can be overwhelmed by the large amount of data needed to be analyzed. In this paper, we propose a light-weight mechanism to detect botnets using their fundamental characteristics, i.e., group activity. The proposed mechanism, referred to as BotGAD (botnet group activity detector) needs a small amount of data from DNS traffic to detect botnet, not all network traffic content or known signatures. BotGAD can detect botnets from a large-scale network in real-time even though the botnet performs encrypted communications. Moreover, BotGAD can detect botnets that adopt recent evasion techniques. We evaluate BotGAD using multiple DNS traces collected from different sources including a campus network and large ISP networks. The evaluation shows that BotGAD can automatically detect botnets while providing real-time monitoring in large scale networks.

1. Introduction

A botnet is a network of computers compromised by malicious software. The botnet is operated by a criminal entity to perform Internet attacks, such as identity theft, spam distribution, and DDoS attack. All of these infected hosts are unwilling victims, performing malicious tasks unbeknownst to their owners.

Researchers have focused on bot traffic detection using incidental traits of prevalent bots. However, the detection approaches can be quickly overcome by evasion techniques. Moreover, some approaches need to run with an overwhelming amount of data, which becomes ineffective for high speed networks.

We have proposed a botnet detection mechanism using a fundamental property of botnets [1,2]. We focused on an underlying common association among infected hosts, command and control (C&C) servers and victims, and found that the botnet generally acts as a coordinated group. Using this "group activity" property, it is possible to detect unknown botnets, irrespective of their communication protocol and structure. BotGAD detects botnets using DNS traffic, since it is possible to capture botnet group activities by monitoring the DNS traffic and monitoring DNS traffic has less overhead than monitoring the entire network traffic. Moreover, DNS monitoring enables botnet detection at their early stages, since botnet DNS traffic is often sent prior to performing attacks.

However, our previous mechanism has three limitations as follows:

- The mechanism may generate false negatives, when a set of infected hosts in a botnet is changed frequently. For example, a part of a botnet can appear only for a short time because they are removed by users or temporally deactivated. The changes in the botnet can decrease detection accuracy of our previous mechanism.

* Corresponding author.
E-mail addresses: realchs@korea.ac.kr (H. Choi), heejo@korea.ac.kr (H. Lee).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.07.018

- The previous mechanism is too sensitive against detection parameters. For example, if a time window parameter of the previous mechanism is misconfigured, the mechanism can generate a significant number of false positives or false negatives.
- The mechanism cannot detect recently introduced evasive botnets that utilize a domain generation algorithm for C&C. For example, Kraken/Bobax [3], Srizbi [4], Torpig [5] and Conficker [6] use the domain generation algorithm (DGA) to evade detection.

To overcome the limitations, we improve the mechanism by applying three methods: error correction, cluster analysis, and hypothesis test.

- *Error correction.* We develop the error correction method to alleviate errors caused when analyzing group activities. Error correction can decrease false alarms caused by unexpected changes in a botnet or by misconfigured detection parameters. We devise column filtering and row filtering operations to correct the errors.
- *Cluster analysis.* We develop a clustering method using unsupervised machine learning to detect a set of correlated botnets. Several features are devised to classify correlated clusters. Each cluster is analyzed to detect botnet clusters.
- *Hypothesis test.* We adopt Sequential Probability Ratio Testing (SPRT) [7], as a hypothesis test for sequential analysis, where a decision is made within a small number of rounds with bounded false alarm rates. The SPRT method guarantees a higher level of confidence to make decisions than simple threshold based detection used in our previous mechanism.

The three methods are added in BotGAD to enhance accuracy and robustness against evasions.

We evaluate BotGAD using real-life DNS traces collected from several networks, such as a campus network and a large ISP network. BotGAD can report hundreds of botnet domains and correlated botnet domain clusters. It takes only a few minutes to analyze an hour's DNS trace of a large ISP network. The evaluation shows BotGAD can automatically detect botnets in real-time, even though they apply evasion techniques, such as the DGA algorithm.

The remainder of this paper is organized as follows. Section 2 reviews related work. We describe the botnet group activity, detection algorithms and a framework of BotGAD in Section 3. We evaluate BotGAD performance in Section 4 and analyze results. We also discuss possible evasion techniques in Section 4. We draw conclusions in Section 5.

2. Related work

In this section, we review several network based botnet detection approaches that can be classified machine learning approaches and non-machine learning approaches. We further classify the approaches according to their detection object such as botnet traffic, cooperative behavior, and spamming botnets. We also distinguish the approaches by the source data they analyze (DNS traffic or other network traffic).

2.1. Non-machine learning approaches

2.1.1. Botnet traffic detection

DNS-based mechanisms. Ramachandran et al. [8] developed techniques and heuristics derived from an idea that detects DNSBL (DNS-based block list) reconnaissance activity of the botmaster whereby the botmasters perform lookups against the DNSBL to determine whether their spam bots have been blacklisted. However, it is easy to design evasion strategies. Salomon et al. [9] proposed and evaluated a Bayesian approach for bot detection based on the similarity of their DNS traffic to that of known bots. The hypothesis of the proposed approach is that bots in the same botnet have similar DNS traffic that can be distinguished from legitimate DNS traffic. However, the approach may generate false positives when a domain name is queried by one infected host and a few uninfected hosts. Sato et al. [10] also proposed a similar approach to detect botnets. Brustoloni et al. [11] described DNS Flagger, a device for ISP bot detection. DNS Flagger matches subscribers' DNS traffic against IP and DNS signatures with the IP addresses and domain names of blacklisted C&C servers, respectively.

Network traffic-based mechanisms. BotHunter [12] modeled the botnet infection life cycle as sharing common steps. It then detects botnets employing IDS-driven dialog correlation according to the bot infection life-cycle model. Karasaridis et al. [13] also proposed a similar approach using IDS-driven dialog correlation according to a defined bot infection dialog model. These bot infection model-based approaches are useful to detect botnets with low false positives. However, malware not conforming to these models would seemingly go undetected. BotCop [14] is a botnet traffic detection system in which the network traffic is fully classified into different application communities using payload signatures and a decision tree model. However, the very nature of signature-based detection renders it easy to evade. RB-Seeker [15] can automatically detect redirection botnets. RB-Seeker gathers information about bots redirection activities. Then it utilizes the statistical methodology and DNS query probing technique to detect botnet redirection domains. However, RB-Seeker only focused on the redirection botnets. Zeidanloo and Manaf [16] proposed a general detection framework that focused on P2P and IRC based Botnets. The framework is based on the definition of botnets that is a group of bots that perform similar communication and malicious activity patterns.

2.1.2. Cooperative network behavior detection

The approaches in this category are closely related to BotGAD since they have a similar concept of capturing the synchronized botnet communication.

DNS-based mechanisms. Manasrah and Hasan [17] proposed a DNS-based mechanism that is similar to our previous mechanism [1,2] since they capture botnet group activities from DNS traffic. However, their approach has limited coverage because they use a MAC address as an

identifier of a host rather than an IP address. The MAC address is visible only to hosts on the same subnet. Therefore, it is not appropriate for monitoring large-scale networks.

Network traffic-based mechanisms. Reiter et al. proposed TAMD [18], a system to detect botnets by aggregating traffic that shares the same external destination, similar payload, and that involves internal hosts with similar OS platforms. BotSniffer [19] is designed to detect IRC or HTTP botnets using a spatial–temporal correlation of botnets. It relies on the assumption that all botnets, unlike humans, tend to communicate in a highly synchronized fashion. BotSniffer performs string matching to detect similar responses from botnets, in contrast to BotGAD. Botnet can encrypt their communication traffic or inject random noise packets for evasion. Yu et al. [20] proposed a real-time based botnet activity monitoring mechanism by using network features such as bps, pps and bytes to detect botnet. However, adversaries can easily manipulate the features by adding a noise to their network traffic.

2.1.3. Spam bot detection

Most recent botnet detection approaches focus on spam bot detection because botnets mainly perform spam distribution. Husna et al. [21] investigated the behavior patterns of spammers based on their underlying similarities in spamming. Zhuang et al. [22] developed techniques to map botnet membership by grouping bots into botnets they look for multiple bots participating in the same spam email campaign. SPOT [23] is a spam zombie detection system for monitoring outgoing messages of a network. SPOT is designed based on a powerful statistical tool, Sequential Probability Ratio Test. Botgraph [24] used graph algorithms to detect web provider email accounts used by botnets to send spam. This analysis helped identify accounts registered by bots during an interval when CAPTCHAs were subverted, allowing automatic bot registration. Spam bot detection obtains high accuracy with low false positives. However, the approaches cannot detect zombie machines or C&C servers that are important to disarm botnets. Only spam relays or proxy servers (or some infected hosts when they directly send spam) are detectable by spam bot detection approaches. Moreover, spam bot detection cannot provide early detection because they are post-mortem methods that can detect botnets only after sending spam mails.

2.2. Machine learning approaches

2.2.1. Bot traffic detection

DNS-based mechanisms. Antonakakis et al. [25] proposed Notos, a dynamic reputation system for DNS that uses passive DNS query data and analyzes the network and zone features of domains. It builds models of known legitimate domains and malicious domains, and uses these models to compute a reputation score for a new domain indicative of whether the domain is malicious or legitimate.

Network traffic-based mechanisms. BotMiner [26] presented a botnet detection method that clusters botnet's communication traffic and activity traffic. Clustering algorithms are applied and performed cross-plane correlation to detect botnets. Yu et al. [27] proposed a technique to detect botnet activities. They transform network traffic flows into multi-dimensional feature, adopt the sliding window to retain the continuous network traffic and select correlation analysis as the similarity measurement. Hosts whose feature streams belong to the same cluster with high similarities will be regarded as suspected bot hosts. Lu et al. [28] proposed an approach for detecting and clustering botnet traffic on large scale network application communities. They classified the network traffic into different applications using traffic payload signatures, and used a decision tree model to classify the traffic to be unknown by the payload content into known application communities to differentiate the malicious botnet traffic from normal traffic on each specific application.

2.2.2. Spam bot detection

SNARE [29] investigates ways to infer the reputation of an email sender based solely on network-level features that enable it to distinguish spammers from legitimate senders.

Even though several approaches have been proposed to detect the botnets, they often suffer from several tactics to evade the detection methods. Stinson and Mitchell [30] proposed a systematic framework to evaluate the evadability of a detection method to assess the fitness of a detection method. We discuss possible evasion tactics and evaluate our mechanism using their systematic framework in Section 4.3.3.

3. Botnet group activity and detection scheme

In this section, we illustrate the concept of our mechanism and a botnet detection scheme.

3.1. Botnet group activity

The main characteristic of a botnet is embedded C&C (Command and Control) systems that allow an attacker (botmaster) to control a pool of compromised machines. Bots communicate through the C&C systems and perform malicious behaviors in a coordinated manner. We start with this fundamental property of a botnet defined as a "group activity". Fig. 1 shows an example of the botnet group activity which has a centralized C&C. Botnet group activities are frequently shown in a botnet life cycle,

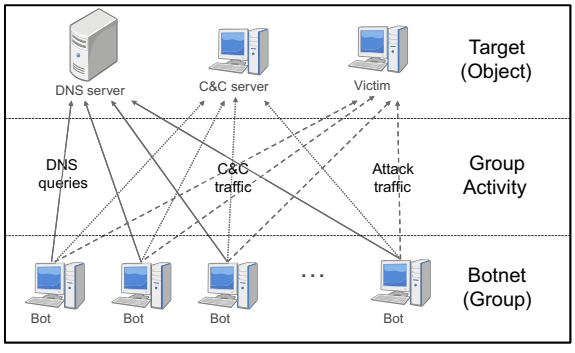

Fig. 1. Botnet group activities (centralized C&C).

particularly more often in centralized botnets since they continuously communicate with their C&C servers. Group activities can be also observed in P2P botnets that have a decentralized architecture [31] (e.g., Bots within the Storm P2P botnet frequently contact the NTP (Network Time Protocol) server as a group to synchronize themselves.

In this study, we use DNS data to capture the botnet group activities. We use the DNS data for three reasons. First, DNS queries are frequently generated during the operation of botnets. Second, DNS occupies only a small portion of network traffic so that we can greatly reduce the amount of data to be handled. Third, DNS monitoring enables botnet detection at their early stages, because the DNS traffic is often sent when bots find C&C servers prior to performing attacks.

In general, the two main purposes of DNS lookup in botnets are (1) rendezvous points lookup (C&C servers or update download servers) and (2) victim lookup.

Rendezvous point lookup. Botnets send DNS queries when they look up C&C servers or update servers. Once a vulnerable machine has been infected, the machine connects to C&C servers to receive orders, and finds update servers to download new binaries. Generally, botnets use DNS to find IP addresses of the rendezvous point [32]. IRC protocol based bots frequently send PING/PONG messages to keep their connection with a C&C server and HTTP protocol based bots periodically/sporadically send HTTP requests to deliver commands from C&C servers [33]. The rendezvous point access and the connection maintenance are repeatedly observed in a botnet lifecycle and can be considered as group activities (accompanied with DNS queries sent in a similar fashion). Some botnets apply a dynamic DNS [34] service to migrate C&C servers frequently.

Victim lookup. Botnets send DNS queries when they perform malicious behavior, such as DDoS attacks, spam distribution and click frauds. For example, when the botnet sends spam, the botnets look up domains in spam recipient lists [35]. Recent spam bots such as Rustock [36] periodically obtain a chunk of recipients and send spam to the recipients. The spam sending behavior of a botnet can induce massive DNS queries for the victim lookup.

Consequently, the coordinated DNS transmission is one of the most frequently observed group activities in a botnet lifecycle. Group activities can be monitored in normal communication as well (e.g., flash crowds). However, group activities of botnet have discriminative characteristics as shown in Table 1. Members of a botnet (i.e., bots) are relatively stable when they perform group activities (consistent group). Conversely, members in a normal group (i.e., benign hosts) generally keep changing over time (inconsistent group). Botnet group activities generally appear intensively having a periodic/sporadic pattern, whereas benign group activities appear at random. Our detection mechanism uses these characteristics to distinguish botnets from legitimate groups. The dynamics of IP addresses can affect the uniformity of the botnet group. The issue of IP dynamics will be discussed in Section 4.3.2.

3.2. BotGAD framework

In this section, we describe the framework of BotGAD. It consists of five main parts: (1) data collector, (2) data mapper, (3) correlated domain extractor, (4) matrix generator, and (5) similarity analyzer (see Fig. 2). The data collector receives and aggregates DNS traffics from the sensors. The data mapper parses the DNS traffic and inserts DNS information into the hash map data structure. The hash map data structure includes a domain map that has a domain name as a key and an IP map as a value, and IP maps that have an IP address number as a key and the information list as a value. The information list has timestamps of each DNS query and DNS based feature values. The matrix generator builds a matrix to measure a similarity score. The correlated domain extractor classifies domain sets using the DNS based features stored in the hash maps. The similarity analyzer calculates the similarity score of generated matrixes. It also performs a hypothesis test to

Table 1
Differences between botnet and normal group activities.

	Group uniformity	Activity periodicity	Activity intensity
Botnet	Consistent	Periodic/ sporadic	Intensive
Normal group	Fluctuate	Irregular	Moderate

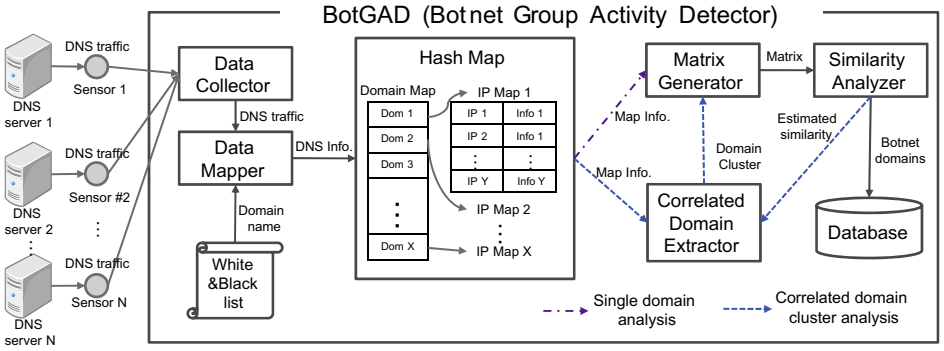

Fig. 2. BotGAD framework.

make a decision to detect botnet domains. Detected botnet domains are summarized in a database.

3.2.1. Matrix generation and error correction

Conventional vector based similarity is one of the most widely used methods to measure similarity [37]. The vectors are constructed to represent original data and a coefficient function operating on the vectors (e.g., cosine coefficient) is used to output the similarity score. We used a binary matrix representation to measure the temporal similarity of group activities. The matrix has column vectors that can be regarded as temporal vectors of a group. Temporal similarity scores can be obtained using the vector based similarity method.

We first translate DNS queries into the form of a binary matrix. In designing the binary matrix construction scheme, we use a domain, a set of IP addresses which queries the domain and a timestamp when each IP address queries the domain. Assume there is an m by n matrix for a domain D. Rows of the matrix represent unique IP addresses that send DNS queries for a domain D and columns correspond to time windows that are evenly distributed time intervals. For example, if an IP_1 queries the domain D within a time window w_1, the matrix generator marks 1 at the matrix element (1,1). After marking all elements in the binary matrix, the similarity analyzer computes the similarity of each neighbor column vector of the matrix.

The matrix representation is useful to measure temporal similarity and our previous mechanisms [1,2] measure similarities in this way. However, we observed two types of errors that can cause false detection. (1) If a time window parameter is misconfigured, especially when there is a relatively loose/random C&C lookup, the matrix is likely to have column vectors that decrease similarity values between neighbor column vectors. (2) If a part of a botnet is removed by users or temporally deactivated, the matrix may have erroneous rows that also decrease similarity values. We devise two filtering operations (column filtering and row filtering) to eliminate such errors. The filtering operation is designed as a pre-processing method to decrease errors incurred by erroneous vectors of a matrix.

Column filtering. Columns of a botnet domain matrix tend to have same (or similar) vector lengths since bots behave as a coordinated group and botnet domains are periodically queried. However, columns of a normal domain matrix tend to have random vector lengths, because the user web access model is known to follow the Poisson model [38].[1] Therefore, we use a vector length difference to find erroneous columns for a botnet domain matrix. In column filtering, column vector $\vec{w}_x$ is deleted when it satisfies

$$\frac{1}{m}\left(\frac{\sum_{i=1}^{n}\|\vec{w}_i\|^2}{n}-\|\vec{w}_x\|^2\right)>\eta_c,$$

where m is the number of rows (i.e., the number of unique IPs seen) and n is the number of columns (i.e., the number of time windows). The equation yields a squared vector length difference between $\vec{w}_x$ and the average length of column vectors in a matrix. If the difference is larger than the column filter threshold η_c, the matrix generator removes the column vector as an error. The pre-defined threshold determines how aggressively to filter erroneous columns of the matrix.

Row filtering. Periodicity difference and row vector length difference metrics are used to delete erroneous row vectors (errors by the bot deviation or deletion). The periodicity metric is measured to find a row that has same temporal patterns as the others and the difference of vector lengths is used similarly as used in the column filtering operation. We choose an erroneous row when a row has a small periodicity difference (shows temporal property of botnets) but has a large row vector length difference. We first measure a time interval between a successive pair of timestamps to estimate the periodicity. Assume the time interval for the xth row is $T_x = t_1, t_2, t_3, \ldots, t_k$ and $\bar{t}$ is a mean value of T_x. Then, the periodicity for x can be measured by the following equation.

$$P(x)=\sqrt{\frac{1}{k}\sum_{i=1}^{k}(t_i-\bar{t})^2}.$$

The periodicity (i.e., a standard deviation of time intervals) is measured for each row in the matrix. The difference between the periodicity and the mean value of the periodicity ($\overline{P}$) is used to measure how similar the periodicity of a row is. We also measure a vector length difference between a row vector $\vec{ip}_x$ and an average length of row vectors in a matrix. If the length difference is larger than the row filter threshold η_r and the periodicity difference is larger than a periodicity difference threshold η_p, the row is removed as an error. In summary, the row filter operation removes a row vector $\vec{ip}_x$ that follows

$$\frac{1}{n}\left(\frac{\sum_{i=1}^{m}\left\|\vec{ip}_i\right\|^2}{m}-\left\|\vec{ip}_x\right\|^2\right)>\eta_r,\quad\text{and}\quad\frac{|P(x)-\overline{P}|}{\overline{P}}<\eta_p.$$

The pre-defined thresholds η_c, η_r and η_p are decided considering the detection rate and false positive/negative rates (discussed in Section 4.2.3).

3.2.2. Correlated domains clustering

Our previous mechanism [1,2] cannot detect a botnet if the botnet utilizes multiple domains at random. The multiple domains can be hard-coded in a bot code or generated by an embedded algorithm. We found three typical cases from real-world botnets: (1) botnets query a domain using a domain generation algorithm, (2) botnets randomly query domains from hard-coded C&C domains and (3) botnets query a domain from a spam recipient list.

Domain generation algorithm. Adversaries have recently developed a flexible and robust channel lookup mechanisms against C&C break down. Kraken/Bobax [3], Srizbi [4], Torpig [5] and Conficker [6] are examples that use a domain generation algorithm to be robust against a sinkhole

[1] Roughly, more than 60% of normal domains have random column lengths that follow the Poisson distribution in our datasets. We used a matrix rank value to measure randomness of the matrix [39].

defense mechanism [40]. Each bot independently uses the generated domains and queries the domains frequently. The bots keep contacting the domains until one of them succeeds.

Multi-domain C&C. Some botnets have a list of multiple domains for C&C and query domains sequentially/randomly.

Botnet spamming. When botnets distribute spam using recipient lists, bots query the mail server domains in the recipient lists. The spam bots share their recipient lists and choose victims randomly. The spam distribution corresponds to overlapped multi-domain lookups.

We devise a cluster analysis method using a machine learning algorithm, *X*-means [41], to detect these correlated domains.

Features. We develop a feature set obtainable from DNS traffic for the cluster analysis. Selection of discriminative features plays a critical role for machine learning based approaches. Therefore, we carefully choose 13 features from three different aspects: (1) DNS lexicology, (2) DNS query information and (3) DNS answer information. These three groups of features can effectively represent the properties of botnet domains to classify the correlated domains.

- *DNS lexicology features.* DNS lexicology features are the textual properties of a botnet domain itself. Domains queried by botnets may have distinguishable textual patterns. For example, spam [42] and phishing [43] domains have different textual patterns from benign domains. Algorithmically generated botnet domains tend to have similar length and number of domain tokens [6]. Therefore, we adopt the token count, the average length of tokens and the longest length of tokens as lexicology features. Moreover, the correlated botnet domains are likely to be hosted by the same provider who often serves malicious domains (e.g., 3322.org); therefore, we employ a binary feature, "blacklisted SLD presence" to check whether a SLD (Second Level Domain) of a domain is matched to a SLD contained in a blacklist. Consequently, four lexical features listed in Table 2 are used in our method (domains are delimited by '·', '/', '?', '=', '–', '_' to obtain the domain tokens).
- *DNS query features.* A set of domains queried by a botnet are sent in a similar way. Bots send a similar number of DNS queries that have the same query type (e.g., A, NS, CNAME, MX and PTR). Therefore, we employ the number of queries sent and the query type as DNS query features. DNS queries from botnets also have similar temporal patterns so we adopt the measured similarity as a DNS query feature (even if it has a small value). The bots are usually distributed over different networks. Thus, we adopt the number of distinct sender IPs and ASNs (Autonomous System Numbers) as DNS query features.
- *DNS answer features.* A DNS answer packet returned by a DNS server usually has several DNS A (address) records. Attackers typically use botnet domains that map to multiple IP addresses that reside in different ASNs, countries and regions. With this insight, we extract four features from the DNS answer data. The number of unique IP addresses, ASNs and countries that are resolved for a given domain are used as DNS answer features. AS numbers and country codes are extracted using MaxMind's database [44]. Every DNS record has a TTL (Time To Live) value that specifies how long the answer for a domain should be cached. Setting lower TTL value is useful for the attackers to achieve higher availability and resistancy against take downs. In particular, botnet domains usually have a very low TTL value when they use DDNS [34] or FFSN [45] service. Thus, we select the TTL value as a DNS answer feature.

Using the features, we apply *X*-means in order to cluster correlated domains.

X-means clustering. *X*-means is a clustering algorithm based on a very popular *K*-means clustering algorithm. Different from *K*-means, the algorithm *X*-means does not have to choose the number of final clusters in advance. It tries to incorporate a search for the best *K* in the process itself. While more comprehensive criteria for finding optimal *K* require running independent *K*-means and then comparing the results, *X*-means tries to split a part of the previously constructed cluster based on the outcome of Bayesian Information Criterion [41]. This gives a much better initial guess for the next iteration and covers a user specified range of admissible *K*.

3.2.3. Similarity analysis

Numerous similarity measures in use, differ primarily in the way they normalize the intersection value. We use the cosine similarity coefficient to measure the domain/cluster similarity. The cosine similarity coefficient is based on the binary term vectors and normalizes the similarity value between 0 and 1. The similarity yields the probability of how many members are presented in a single group and presented in the other group simultaneously. Therefore, the cosine similarity coefficient between column vector w_1 and w_2 indicates measured similarity of the group between w_1 and w_2. The cosine similarity S_{Cos} is measured as the following equation,

$$S_{Cos}(\vec{w}_i, \vec{w}_{i+1}) = \frac{\vec{w}_i \cdot \vec{w}_{i+1}}{\|\vec{w}_i\| \|\vec{w}_{i+1}\|}.$$

In the case of a domain cluster, we measure two values: cosine similarity and intensity of a cluster matrix. We generate a matrix for a cluster using every DNS queries involved in the cluster. The intensity value is used to estimate periodic/sporadic pattern of a cluster. Assume that a cluster has n domains and $\|m_i\|^2$ is the number of elements that have a value of 1 in a domain matrix i, and $\|M\|^2$ is the number of elements that have a value of 1 in a cluster matrix. Then, the matrix intensity is calculated as

$$I = \frac{\|M\|^2}{\sum_{i=0}^{n} \|m_i\|^2}.$$

The similarity analyzer detects botnet domain clusters using a metric of $\frac{S_{Cos}+I}{2}$. The metric is used for a hypothesis test of a cluster instead of a similarity value.

Sequentialprobability ratio test. Our previous mechanism calculates an average value of the similarity and uses a

Table 2
Selected features for clustering.

Feature type	Feature name
DNS lexicology features	Number of domain tokens Average length of domain tokens Longest length of domain token Blacklisted SLD presence
DNS query features	Number of queries sent Number of distinct sender IPs Number of distinct sender ASNs Query type (A, NS, CNAME, MX, PTR) Estimated similarity of a domain
DNS answer feature	Number of distinct resolved IPs Number of distinct ASNs of resolved IPs Number of distinct countries of resolved IPs TTL value in a DNS answer packet

fixed threshold to detect botnet domains. However, the simple threshold based method can make an incorrect decision, particularly when botnets loosely/randomly look up domains or a time window parameter is misconfigured. Therefore, we apply a hypothesis test that utilizes multiple observations for decision making can guarantee a higher level of confidence than the simple threshold method. We use the Sequential Probability Ratio Testing (SPRT) which is a statistical method for testing a hypothesis with a bounded false positive rate and false negative rate. BotGAD measures similarity and after a number of tests, BotGAD produces the acceptance or rejection of the hypothesis.

We consider two hypotheses: H_0 denotes the domain (or the cluster) is detected as a benign domain and H_1 denotes the domain (or the cluster) is detected as a botnet domain.

$$Pr(X_i|H_0) = \theta_0, \quad Pr(X_i|H_1) = \theta_1.$$

Let θ_0 and θ_1 denote the probability of a domain being a botnet domain and a normal domain, respectively. Let S_1, S_2,...,S_n be an observed samples sequence of a domain (or a cluster), then we have the likelihood ratio Λ_n as

$$\Lambda_n = ln\frac{Pr(S_1, S_2, \ldots, S_n|H_1)}{Pr(S_1, S_2, \ldots, S_n|H_0)} = \prod_{k=1}^{n} ln\frac{Pr(S_k|H_1)}{Pr(S_k|H_0)}.$$

The equation represents the step in the hypothesis test that generates the accumulated likelihood ratio. The hypothesis test is then defined as follows. Given two user specified threshold λ_0 and λ_1 where $\lambda_0 < \lambda_1$, at each test we compute the likelihood ratio and check the stopping rule as follows:

$$\Lambda_n \leqslant \lambda_0, \quad \text{then accept } H_0,$$
$$\Lambda_n \geqslant \lambda_1, \quad \text{then accept } H_1,$$
$$\lambda_0 < \Lambda_n < \lambda_1, \quad \text{then pend the decision.}$$

The thresholds λ_0 and λ_1 can be set according to the user-chosen false positive rate α and false negative rate β. By setting the threshold $\lambda_0 = \frac{1-\beta}{\alpha}$ and $\lambda_1 = \frac{\beta}{1-\alpha}$, the true bounded false positive rate and false negative rate can be acquired.

4. Evaluation result and analysis

We tested BotGAD on three different real-life traces to evaluate its performance. We also evaluate the performance of the three modules, i.e., error correction, correlated cluster analysis and hypothesis test. We discuss evadability and compare BotGAD to other mechanisms.

4.1. Datasets and result verification

We obtained three DNS traces from three different networks:

- *Trace #1*: This DNS traffic was tapped from the gateway router of a /16 campus network on December 24th, 2008. There are 4.6 million DNS queries in 1.48 GB of captured DNS traffic.
- *Trace #2*: This DNS traffic was collected from ISP DNS servers on July 7th, 2009. The network size is much larger than the campus network (453.6 million DNS queries were captured).
- *Trace #3*: This data has DNS logs tapped from ISP DNS servers on June 15th 2010. The data consists of dynamic DNS logs that have more than 15 million DNS queries.

We need a method to ensure the result for the evaluation since it is difficult to know all hidden botnets in the real-life traces. Hence, we design a method to verify experimental results with the help of third parties such as search engines. A combination of the following approaches is used to verify the results.

1. *Blacklist matching*: We match detected domains with a blacklist collected from several resources (Korea Information Security Agency (KISA) sinkhole domain list [46], DNS-BH list [47] and Cyber-TA list [48]). The blacklists include more than 200,000 domain names.
2. *Web reputation search*: We use online Web reputation search tools such as McAfee SiteAdvisor [49] and WOT (Web of Trust) [50] that provide the reputation of a submitted website domain including the detailed categories to which it belongs.
3. *IP address resolution*: Resolved IP information is used to check whether the resolved IP address is abnormal or inaccessible.
4. *Domain information investigation*: We look up the domain using Google and a domain crawler [51] to determine whether it is a domain for a name server, mail server, or web server and to find that resolved IP addresses of the domain is listed on RBLs such as Spamhaus SBL.

We categorize the detected result into known botnets, unknown botnets and false positives. The known botnets are revealed in steps 1 and 2. If a domain is listed in the blacklist or reported as a malicious domain by the Web reputation tools, the domain is classified as a known botnet domain. We verify unknown botnets from steps 3 to 4. If a domain is determined as suspicious at these steps, we regard the domain as an unknown suspicious domain. For

example, if a domain has a resolved IP address listed on RBL, the domain is classified into the unknown suspicious. Remainders are considered as false positives. Knowing a complete set of hidden botnets can be hardly done since we evaluate them with the real-life traces.[2] Therefore, the exact number of false negatives is difficult to acquire. In the evaluation of our study, the false negatives are approximately estimated by comparing the detection result to our blacklist. When a domain is listed in the blacklist but not detected by BotGAD, it is counted as a false negative. The detection rate and false negatives in our evaluation are likely to be altered depending on the blacklist. However, they can show the improvements of relevant metrics.

We use three metrics in the evaluation: (1) detection rate: the proportion of true detected botnet domains by BotGAD over a union of true detected botnet domains by BotGAD and by the blacklist. (2) false positive rate: the number of false positive domains divided by the total number of distinct normal domains. (3) false negative rate: the number of false negative domains divided by the total number of true botnet domains.

4.2. Evaluation results

4.2.1. Single domain analysis result

This section describes the evaluation results of the single domain analysis. We observe 142,045, 16,420,531 and 1,301,919 unique queried domains in the trace #1, #2, and #3, respectively. More than 80% of the domains are queried by only a single host. In the single domain analysis, BotGAD measures the similarity of a domain that is queried by more than three hosts (three unique IPs). BotGAD analyzes 7,850, 984,000 and 151,326 distinct domains as shown in Table 3.

We measure the similarity coefficients (with a 10 min time window parameter). 72% of domains had similarities estimated to 0.

We obtain 28.9%, 17.6% and 28.7% of detection rate using the trace #1, #2 and #3, respectively. The detection rates are very low and the false negative rates are very high because many correlated botnet domains (e.g., Conficker domains) are not detected in the single domain analysis. Moreover, BotGAD cannot detect a single infected client in the single domain analysis, e.g., asp.ircdevilz.net**. The result shows the necessity of the cluster analysis.

Examples of detected botnet domains and false positives are listed in Table 4 (trace #1). Most false positives are related to update domains such as antivirus software updates, installshield updates and toolbar updates. The update related domain groups occur frequently and periodically, similar to botnets. BotGAD reports time.nist.gov* as a botnet domain. We find that 20 of the 89 hosts who query the domain generate excessive queries, estimated at 4.2 queries/s. Obviously, they are abnormal because NIST does not allow queries being sent more frequently than once every four seconds. The hosts had been infected by the Storm botnet, known as the largest P2P botnet. Storm synchronizes the system time of the infected machine with the help of the Network Time Protocol (NTP) using time server domains (time.nist.gov and time.windows.com) [52]. The result implies BotGAD can capture not only IRC and HTTP botnets but also P2P botnet (Storm) synchronization activities.

[2] Thus, network anomaly detection approaches commonly use synthetic network traffic data for evaluation, e.g., MIT/DARPA data sets.

Table 3
Single domain analysis result.

	Trace #1	Trace #2	Trace #3
DNS queries	4.6 M	453.6 M	15 M
No. of distinct domains	7,850	984,000	151,326
Detected domains	43	512	486
Unknown suspicious	12	250	191
Known botnets	10	135	170
False positives	21	127	125
False negatives	54	1805	897
Detection rate (%)	28.9	17.6	28.7
False positive rate (%)	0.27	0.01	0.08
False negative rate (%)	71.1	82.4	71.3

Table 4
Detected botnet domains and false positives.

Result type	Domain name	Group size	Average similarity
Suspicious domains	bosam.gnway.net	13	0.99
	shiyansend.zyns.com	33	0.98
	shiyansend.solaris.nu	33	0.97
	shiyansend.servebbs.org	29	0.87
Known botnet domains	tzhen.3322.org	14	0.92
	proxima.ircgalaxy.pl	33	0.87
	proxim.ntkrnlpa.info	4	0.85
	roon.shannen.cc	11	0.83
	time.nist.gov*	50	0.91
False positives	updates.installshield.com	22	0.94
	us.update2.toolbar.yahoo.com	33	0.93
	asp.ircdevilz.net**	1	0

4.2.2. Cluster analysis result

This section shows the evaluation result of the cluster analysis. Several machine learning based clustering [26–28] approaches are proposed to detect botnets. However, the clustering approaches differ from our clustering method. We devise DNS based features using domain lexicons, DNS query and answer information, whereas other approaches use network traffic based features such as pps (packets per second) and bps (bytes per second). Generally, those network traffic features are easy to manipulate for evasion.

By analyzing each trace (#1, #2 and #3), BotGAD classifies 144, 320, 208 clusters, respectively. Table 5 lists the clustering result for each trace. As expected, most detected botnet clusters are: (1) algorithmically generated domains by botnets (Conficker's domain, e.g., yrxhwjlzg.org, ylehaairwse.biz, ykywftwssjc.com), (2) randomly used multiple C&C domains or binary download domains (e.g., *.jiangmin.com, *.duba.net, *.http://www.duba.net), and (3) domains for sending spam mails (mail server domains, e.g., mail.global.frontbridge.com, mail.global.sprint.com, mail.global.mas.att.com). The result shows the cluster

Table 5
Cluster analysis result.

	Trace #1	Trace #2	Trace #3
Clusters	144	320	208
Detected clusters	12	42	7
Botnet clusters	9	27	5
False positive clusters	3	15	2

analysis enables BotGAD to detect all three types of correlated botnet domains.

Most false positive clusters are related to consequently generated domains when accessing a website. Different domains are queried all together to get each content of the Web page. In particular, a website served by Content Delivery Network (CDN) often uses an array of domains and the domains have temporal properties of a botnet group activity. Therefore, the domains of the website are likely to be detected as a botnet domain cluster. These false positives can be decreased by whitelisting popular domains[3] that are usually served by CDNs.

Feature analysis. We evaluate each individual feature to know the effectiveness of each feature. We use the precision metric[4] to measure clustering accuracies. First, we perform clustering with all features. About 97% precision is obtained (averaged precisions using the three traces). We then exclude a feature one-by-one and measure a precision improvement (leave-one-out approach). Table 6 shows the results.

The number of queries sent, the number of distinct sender IPs, and the estimated similarity of a domain are the top three features that contribute the most to the clustering precision. We find out that the three features have different distributions for each type of cluster (i.e., normal clusters, generated botnet domain clusters, multiple C&C domain clusters, and spam domain clusters). The DNS lexicology features have different distributions for the generated botnet domain clusters and the multiple C&C domain clusters, but not for the spam domain clusters. The DNS answer features are effective to cluster the multiple C&C domain clusters. However, the features were not effective in distinguishing the other types of clusters because most of generated botnet domains do not have answer records (a few of them had the answer records) and spam domains do not have similar answer record patterns. If we consider the reliability of the features, DNS query features are the most effective features, since they are more difficult to manipulate than the other two types of feature. Therefore, the top three features, i.e., the number of queries sent, the number of distinct sender IPs, and the estimated similarity of a domain, are the most effective features that contribute to both precision and reliability.

Performance improvement. Table 7 lists the detection results using both the single domain analysis and the cluster analysis. The result shows the improvements of the cluster analysis that was not applied in our previous mechanism. [1,2]. The cluster analysis significantly increases the detection rates and the false negative rates, e.g., 71% improvement of the detection rates on average.

[3] Whitelist domains are taken from the Alexa top 500 site list.

[4] Precision = number of true positive classifications/number of positive classifications.

Table 6
The precision gain for each feature (%).

Type	Feature	Precision+
DNS lexicology feature	Number of domain tokens	4.2
	Average length of domain tokens	4.7
	Longest length of domain token	3.5
	Blacklisted SLD presence	6.3
DNS query feature	Number of queries sent	8.8
	Number of distinct sender IPs	9.2
	Number of distinct sender ASNs	6.7
	Query type (A, NS, CNAME, MX, PTR)	4.5
	Estimated similarity of a domain	9.6
DNS answer feature	Number of distinct resolved IPs	6.7
	Number of distinct ASNs of resolved IPs	3.3
	Number of distinct countries of resolved IPs	2.7
	TTL value in a DNS answer packet	3.9

Table 7
Result summary using both single domain analysis and cluster analysis (%).

	Trace #1	Trace #2	Trace #3
Detection rate	97.2	95.4	96.1
False positive rate	0.31	0.11	0.05
False negative rate	2.8	4.6	3.9

4.2.3. Error correction result

This section shows evaluation results of the error correction method. As mentioned in Section 3.2.1, we determine the thresholds η_c, η_r and η_p to minimize false positive/negative rates.

Figs. 3 and 4 show false positive and negative rates using trace #1. In the experiment, we set the time window $w = 10$ min. When both row filtering threshold (η_r) and column filtering threshold (η_c) are increased, both filtering operations coarsely eliminate errors. Therefore, high filtering thresholds result in low false positive rates but high false negative rates as shown in the figures. Conversely, low filtering thresholds result in low false negative rates

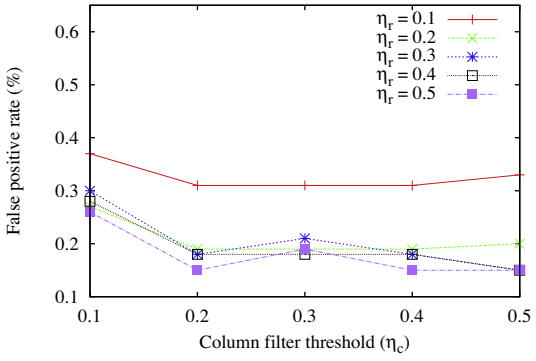

Fig. 3. False positive rate of trace #1.

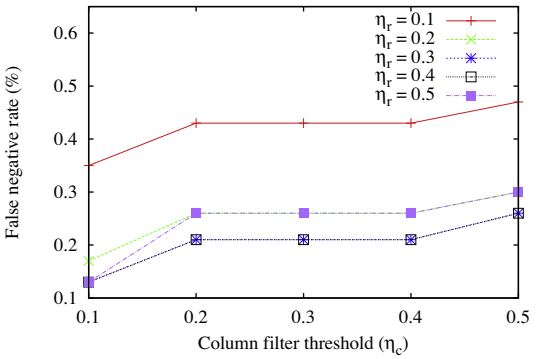

Fig. 4. False negative rate of trace #1.

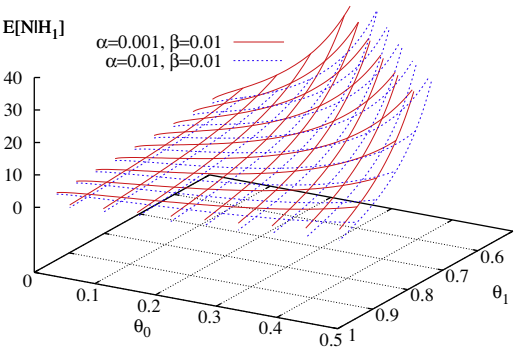

Fig. 5. Average number of observation rounds ($E[N|H_1]$).

but high false positive rates because the filtering operations aggressively delete not only erroneous rows/columns of a botnet domain matrix but also scarce rows/columns of a normal domain matrix. When monitoring networks generate a small amount of traffic (DNS queries), aggressive filtering can affect the detection accuracy critically. Thus, high thresholds should be designated for small networks.

When the filtering thresholds are $\eta_r = 0.3$ and $\eta_c = 0.2$, we have the best results with an optimal trade-off between false positives and negatives on trace #1. Similarly, when $\eta_r = 0.2$ and $\eta_c = 0.1$, best results are obtained on trace #2 and trace #3. We set periodicity threshold $\eta_p = 0.3$ since it produced the best results for all cases. It is possible to decrease 12% of false positives and 28% of false negatives on average. In this analysis, we do not count the number of detected correlated botnet domains that are only detectable by cluster analysis as false negatives to know the exact improvement in the error correction method.

4.2.4. Hypothesis test

BotGAD uses SPRT to determine botnet domains/clusters so that the average number of observation rounds ($E[N|H_1], E[N|H_0]$) to reach the decision are represented as the two following equations [7]:

$$E[N|H_1] = \frac{\beta \ln \frac{\beta}{1-\alpha} + (1-\beta)\ln \frac{1-\beta}{\alpha}}{\theta_1 \ln \frac{\theta_1}{\theta_0} + (1-\theta_1)\ln \frac{1-\theta_1}{1-\theta_0}},$$

$$E[N|H_0] = \frac{(1-\alpha)\ln \frac{\beta}{1-\alpha} + \alpha \ln \frac{1-\beta}{\alpha}}{\theta_0 \ln \frac{\theta_1}{\theta_0} + (1-\theta_0)\ln \frac{1-\theta_1}{1-\theta_0}},$$

where α and β are user-chosen false positive and false negative probabilities, respectively.

Fig. 5 shows the value of $E[N|H_1]$ as a function of θ_0 and θ_1 with fixed $\alpha = 0.01$, $\beta = 0.01$ and $\alpha = 0.001$, $\beta = 0.01$. Only small observations are needed to reach the decision for SPRT as shown in the figure. For instance, if the user-defined false positive and negative rates are 0.01 and $\theta_0 = 0.2$, $\theta_1 = 0.8$, then BotGAD needs six observations to make the decision. BotGAD needs a smaller number of $E[N|H_1]$ if we apply smaller θ_0, α, β, and larger θ_1. The relationship between false alarm rates (α, β) and observation rounds ($E[N|H_1]$) illustrates the trade-offs between effectiveness and efficiency of the detection algorithm.

4.3. BotGAD analysis

4.3.1. Deployment and performance analysis

BotGAD can be used to detect botnet domains at any network location that sees a collective DNS query sent by users. If a monitoring network is large, more infected hosts can be found. Therefore, it is recommended to setup BotGAD to monitor large scale network or to deploy sensors in distributed networks. There are good options for receiving DNS data, e.g., from anti-virus softwares, browser toolbar, or such types of products.

We now present system performance to justify the real-time nature of BotGAD. This is a very important aspect for two reasons. First, adversaries continuously change their domains and IP addresses of the domains. Therefore, real-time blocking of such domains is needed to alleviate them effectively. Second, it is important to identify suspicious botnet domains that need a real-time inspection to ensure the result. We run BotGAD on a machine that has a 3.0 GHz dual core CPU with a 4 GB RAM. Coarsely speaking, BotGAD can run as a real-time system since it took about 20, 318 and 58 min to process the day traces #1, #2 and #3, respectively. For an hour DNS trace analysis, it even took less than 1, 5 and 2 min.

4.3.2. Parameter analysis

This section describes an analysis of a similarity measure along with relevant parameters. Through the parameter analysis, we can estimate appropriate values of parameters to measure accurate similarities. We divide the parameters into two types: the botnet related parameters and the BotGAD related parameters as shown in Table 8. If $\Delta n = 0$, it is clear that we can detect a botnet domain group. Suppose the worst case that a botnet forms a group appeared randomly. We derive a similarity equation using Poisson distribution.

$$S = \frac{n}{n + |\Delta nt|}\left(1 - \frac{1}{e^{\gamma(t-x)}}\right), \quad x = \begin{cases} w, & w \geqslant T_L, \\ T_L, & w < T_L. \end{cases}$$

The derived equation consists of four parts: TTL in DNS resource record T_L, group size parameter n and Δnt, parameters from the detection mechanism t and w, and DNS querying ratio γ.

Table 8
Parameters related with BotGAD.

	Parameter description	Variable
Botnet parameters	TTL in DNS resource record	T_L
	DNS querying ratio	γ
	Group size and change in the group size	n,Δn
BotGAD parameters	Monitoring time	t
	Time window	w

- *TTL in DNS resource record* T_L: Most operating systems, including Windows, have a DNS resolver cache. For example, when the Windows resolver receives a positive or negative response to a query, it adds that positive or negative response to its cache, and as a result, creates a DNS resource record. If a DNS resource record is in the cache, the resolver uses the record from the cache instead of querying. After a period specified in TTL in the DNS resource record, the resolver discards the record from the cache. The cache can decrease botnet DNS queries as well as normal queries. Botnets commonly use DDNS [53]. Therefore, it is possible to roughly estimate the value of TTL applied in botnet domains.
- *DNS querying ratio* γ: We set w to 10 min (<1 h) in our experiment, so we can assume $\Delta n = 0$. $T_L < w$. The derived equation can be approximated to

$$S \approx 1 - \frac{1}{e^{\gamma(t-w)}}.$$

If $t = 5w$, similarity S will be approximated as shown in Fig. 6. The graph implies that γ is also one of the most important parameters of BotGAD.

- *Change in the group size* Δn: We can estimate Δn from the botnet propagation model [32]. In addition, dynamics of IP address should be considered. The authors in [54] address that more than half (61.4%) of the IP addresses were observed as dynamic IP addresses. However, over 95% of IP addresses have inter-user durations longer than an hour. Errors derived from dynamics of IP addresses can be ignored when we choose the w within an hour.
- *Time window* w: To set an appropriate value of a time window w, relevant parameters, i.e., γ,T_L and false positive/negative rates should be considered. (1) If γ is larger than w, false negatives will be increased because too small w will decrease the estimated similarity. (2) If T_L is larger than w, false negatives will be increased due to the DNS resolver cache effect. (3) If w is too large, false positives will be increased since many normal groups are possibly selected as suspicious groups. Consequently, the lower bound of w should be larger than both γ and T_L($w \geqslant max(\gamma, T_L)$). The γ and T_L can be estimated, considering existing bot implementations. The upper bound of w should be determined to have the smallest number of false positives possible.

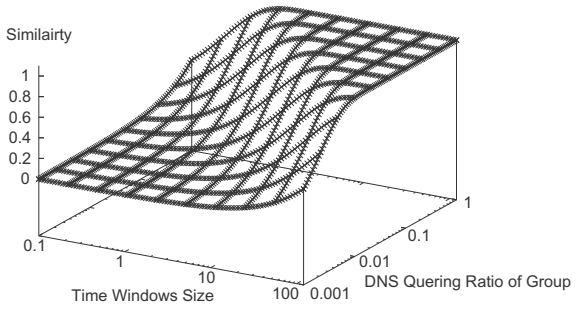

Fig. 6. Relationship among similarity S, time window w, and DNS querying ratio γ.

4.3.3. Evadability analysis

Even though several automated botnet detection mechanisms have been proposed, botnets can evade them with a high level of attacker power. Our mechanism is to be evaded to a certain degree, as for the existing mechanisms. However, it is an improvement if we can raise a bar of evasion difficulty by either increasing the evasion cost or decreasing the effectiveness of botnets. We refer to Stinson's [30] systematic evaluation which demonstrates an evasion tactic in respect of two associated costs: implementation complexity and effect on botnet utility. An evasion tactic's implementation complexity is based on the ease with which bot writers can incrementally modify current bots to evade detection. If it takes a high implementation cost, the evasion tactic is less useful. Affecting on botnet utility is also important to the adversaries, because an evasion tactic is less effective if the tactic restricts the botnet utility. Stinson et al. list the botnet utility including diversity of attacks, lead time required to launch an attack, botnet size, attack rate, and synchronization level. We address likely evasion techniques of BotGAD and evaluate the evadability as shown in Table 9.

1. *Evasion by threshold attacks*: BotGAD relies on time related variables such as the time window parameter. Therefore, a botmaster can control the frequency of botnet behaviors to decrease similarities. We add the error correction and hypothesis test method to BotGAD to be more robust against such threshold attacks. Moreover, the implementation complexity is high and applying

Table 9
Evasion tactics used for defeating BotGAD, the tactic's implementation complexity and effects on botnet utility.

Evasion tactic	Implementation complexity	Effects on botnet utility
1. Threshold attacks	High	↓Attack rate
2. Botnet subgrouping	Low	↓Botnet size
3. Minimize the synchronicity	High	↓Attack diversity
4. Induce IP churn	Unknown	None
5. Generate fake DNS queries	Low	None

this tactic reduces attack rates of the botnet. Therefore, the threshold attack is less effective than the other tactics to evade BotGAD.

2. *Evasion by the botnet subgrouping*: A botmaster can apply multi-purpose time sharing botnets where a subset of bots is used for a single purpose (e.g., DDoS) and others for another purpose (e.g., spamming). If elements (bots) of each subset are not changed, BotGAD can detect each subset independently. Even if a botmaster randomly changes subsets, BotGAD can detect the subsets with cluster analysis. Therefore, the subgrouping tactic is also less useful for attackers.
3. *Evasion by minimizing the synchronicity*: Bots can delay communication time and do not perform any synchronized attacks to evade BotGAD. For example, many new botnets have adopted a P2P architecture, where bots are coordinated in a distributed fashion. BotGAD cannot capture the P2P bots if bots have a long time delay for communications and attacks. However, botnet utilities including botnet's synchronization level and diversity of attacks will decrease significantly in such a case.
4. *Evasion by inducing IP churn*: BotGAD uses an IP address as an identifier of a host. Therefore, if bots can change their IP address on demand, BotGAD will be defeated. The implementation complexity of this technique is unknown.
5. *Evasion by generating Fake DNS queries*: If bots intentionally generate fake DNS queries using a source spoofing method, the queries can poison our previous mechanism.

If botnets avoid using DNS, BotGAD cannot detect the botnets. Some botnets have been observed to use hard-coded IP addresses for C&C even though the IP based C&C is easy to take down.[5] Group activities of the IP based C&C cannot be detected by BotGAD, but other group activities (e.g., infection, binary download and spamming) are detectable if the activities utilize DNS. To be sure, botnets can never use DNS during every part of its lifecycle. In such a case, however, the botnet utility will be restricted.

4.3.4. Comparison to other mechanisms

This section lists advantages and disadvantages of the BotGAD based on key features including: (1) ability to detect unknown bots, (2) capability of botnet detection regardless of botnet protocol and structure, (3) robust against evasions including channel encryption, packing, traffic manipulation and random domain generation, (4) capability of real-time detection and ability to monitor large scale networks, and (5) capability of early detection. These features are used in the study by Feily et al. [55].

- *Unknown botnet detection.* BotGAD can detect unknown botnets, whereas some other works, such as signature-based techniques, are unable to detect them.
- *Protocol & structure independent.* BotGAD can detect botnets regardless of botnet protocol if the botnets use DNS traffic. However, botnets that do not use DNS are undetectable by BotGAD. Moreover, BotGAD cannot detect C&C of P2P botnets since it is not a group activity according to our definition. Only malicious attacks by P2P botnets are able to be captured by BotGAD. These limits are the disadvantages of BotGAD compared to other approaches [26–28] that provide protocol & structure independent botnet detection.
- *Robust against evasions.* BotGAD is robust to channel encryption, packing, traffic manipulation (by adding a noise in communications) and random domain generation for C&C. However, many approaches introduced in Section 2 are weak to such evasions. Therefore, the robustness to evasions is one of the most important benefits of this study.
- *Real-time detection and scalability.* BotGAD is designed as a light-weight system to provide real-time detection and large coverage of monitoring networks. However, there are many approaches that cannot detect botnets in real-time or are only deployable to small networks. We regard the abilities of real-time detection and large coverage as major benefits of this study.
- *Early detection.* BotGAD can detect botnets in their early stages, since BotGAD can capture botnet DNS queries that are often sent when bots find C&C servers prior to performing attacks. The early detection is helpful for network operators to clean up compromised computers inside their networks, alleviating further infections and losses from attacks. However, many other mechanisms (e.g., spam bot detection approaches) do not provide early detection.

In summary, BotGAD has advantages in four aspects: (1) ability to detect unknown botnet detection, (2) robustness to evasions, (3) ability to monitor large scale networks in real-time and (4) ability to detect botnets at their early stage. Our mechanism can *effectively* and *efficiently* detect unknown botnets in large scale networks due to these advantages.

5. Conclusion

Botnets are the major threats to network security and major contributors to unwanted network traffic. Thus, it is necessary to provide appropriate countermeasures to botnets. We propose BotGAD to reveal both unknown domain names of C&C servers and IP addresses of hidden infected hosts. We define an inherent property of botnets, termed group activity. Using this property, we propose a light-weight mechanism to detect botnets. Our mechanism needs a small amount of data from DNS traffic, not all the traffic content or known signatures. BotGAD can detect botnets from a large-scale network in real-time even though the botnet performs encrypted communications. Moreover, BotGAD can detect not only individual botnets but also correlated evasive botnets, using unsupervised machine learning. Our method provides over 95% detection rates while generating less than 0.4% false positive rates and 5% false negative rates based on experiments with

[5] DNS based C&C is harder to take down since they can circumvent botnet defense systems by changing resolved IP addresses continuously.

real-life campus and ISP DNS traces. It takes only a few minutes to analyze an hour-long DNS trace of a large ISP network. The evaluation results prove that BotGAD can automatically detect botnets in large scale networks.

Acknowledgments

This research was supported by the MKE, Korea, under the ITRC support program supervised by the NIPA (NIPA-2011-C1090-1131-0005) and the Seoul R&BD Program (WR080951). The preliminary version of this paper was presented in IEEE CIT [1] and COMSWARE [2].

References

[1] H. Choi, H. Lee, H. Lee, H. Kim, Botnet Detection by monitoring group activities in dns traffic, in: Proceedings of the IEEE International Conference on Computer and Information Technology (CIT), 2007.
[2] H. Choi, H. Lee, H. Kim, BotGAD: detecting botnets by capturing group activities in network traffic, in: Proceedings of International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE), 2009.
[3] S. Shevchenko, Kraken changes tactics. <http://www.blog.threatexpert.com/2008/04/kraken-changes-tactics.html>, 2009.
[4] J. Wolf, Technical details of Srizbi's domain generation algorithm. <http://www.blog.fireeye.com/research/2008/11/technical-details-of-srizbis-domain-generation-algorithm.html>, 2008.
[5] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, G. Vigna, Your Botnet is My Botnet: Analysis of a Botnet Takeover, 2009.
[6] P. Porras, H. Saidi, V. Yegneswaran, A foray into Conficker's logic and rendezvous points, in: Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), 2009.
[7] A. Wald, Sequential tests of statistical hypotheses, The Annals of Mathematical Statistics (1945) 117–186.
[8] A. Ramachandran, N. Feamster, D. Dagon, Revealing botnet membership using DNSBL counter-intelligence, in: Proceedings of the Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI), 2006.
[9] R.V.-Salomon, J.C. Brustoloni, Bayesian bot detection based on DNS traffic similarity, in: Proceedings of the ACM Symposium on Applied Computing (SAC), 2009.
[10] K. Sato, K. Ishibashi, T. Toyono, N. Miyake, Extending black domain name list by using co-occurrence relation between dns queries, in: Proceedings of the workshop on Large-scale Exploits and Emergent Threats (LEET), 2010.
[11] J. Brustoloni, N. Farnan, R. Villamarin-Salomon, D. Kyle, Efficient detection of bots in subscribers' computers, in: Proceedings of IEEE International Conference Communications (ICC), 2009.
[12] G. Gu, P. Porras, V. Yegneswaran, M. Fong, W. Lee, BotHunter: detecting malware infection through ids-driven dialog correlation, in: Proceedings of the USENIX Security Symposium (Security), 2007.
[13] A. Karasaridis, B. Rexroad, D. Hoeflin, Wide-scale botnet detection and characterization, in: Proceedings of the Workshop on Hot Topics in Understanding Botnets (HotBots), 2007.
[14] W. Lu, M. Tavallaee, G. Rammidi, A.A. Ghorbani, BotCop: an online botnets traffic classifier, in: Proceedings of the Conference on Communication Networks and Services Research (CNSR), 2009.
[15] X. Hu, M. Knyz, K.G. Shin, RB-Seeker: auto-detection of redirection botnets, in: Proceedings of the Annual Network and Distributed System Security Symposium (NDSS), 2009.
[16] H.R. Zeidanloo, A.B.A. Manaf, Botnet Detection by Monitoring Similar Communication Patterns, International Journal of Computer Science and Information Security (IJCSIS) (2010).
[17] A.M. Manasrah, A. Hasan, O.A. Abouabdalla, S. Ramadass, Detecting Botnet Activities Based on Abnormal DNS traffic, International Journal of Computer Science and Information Security (IJCSIS) (2009).
[18] M. Reiter, T. Yen, Traffic aggregation for malware detection, in: Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), 2008.
[19] G. Gu, J. Zhang, W. Lee, BotSniffer: detecting botnet command and control channels in network traffic, in: Proceedings of the Annual Network and Distributed System Security Symposium (NDSS), 2008.
[20] X. Yu, X. Dong, G. Yu, Y. Qin, D. Yue, Y. Zhao, Online botnet detection by continuous similarity monitoring, in: Proceedings of International Symposium Information Engineering and Electronic Commerce IEEC'09, 2009.
[21] H. Husna, S. Phithakkitnukoon, S. Palla, R. Dantu, Behavior analysis of spam botnets, in: Proceedings of the International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE), 2008.
[22] L. Zhuang, J. Dunagan, D.R. Simon, H.J. Wang, I. Osipkov, G. Hulten, J.D. Tygar, Characterizing botnets from email spam records, in: Proceedings of the workshop on Large-scale Exploits and Emergent Threats (LEET), 2008.
[23] Z. Duan, P. Chen, F. Sanchez, Y. Dong, M. Stephenson, J. Barker, Detecting spam zombies by monitoring outgoing messages, in: Proceedings of the Annual IEEE Conference on Computer Communications (INFOCOM), 2009.
[24] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, E. Gillum, Botgraph: Large scale spamming botnet detection, in: Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2009.
[25] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, N. Feamster, Building a dynamic reputation system for DNS, in: Proceedings of the USENIX Security Symposium (Security), 2010.
[26] G. Gu, R. Perdisci, J. Zhang, W. Lee, BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection, in: Proceedings of the USENIX Security Symposium (Security), 2008.
[27] X. Yu, X. Dong, G. Yu, Y. Qin, D. Yue, Data-adaptive clustering analysis for online botnet detection, in: Proceedings Third International Computational Science and Optimization (CSO) Joint Conference, 2010.
[28] W. Lu, G. Rammidi, A.A. Ghorbani, Clustering botnet communication traffic based on n-gram feature selection, Computer Communications (2010).
[29] S. Hao, N. Syed, N. Feamster, A.G. Gray, S. Krasser, Detecting spammers with snare: spatio-temporal network-level automatic reputation engine, in: Proceedings of the USENIX Security Symposium (Security), 2009.
[30] E. Stinson, J.C. Mitchell, Towards systematic evaluation of the evadability of bot/botnet detection methods, in: Proceedings of the USENIX Workshop on Offensive Technologies (WOOT), 2008.
[31] J. Grizzard, V.Sharma, C. Nunnery, B. Kang, D. Dagon, Peer-to-peer botnets: overview and case study, in: Proceedings of the Workshop on Hot Topics in Understanding Botnets (HotBots), 2007.
[32] D. Dagon, G. Gu, C. Lee, W. Lee, A taxonomy of botnet structures, in: Proceedings of the Annual Computer Security Applications Conference (ACSAC), 2007.
[33] L. Liu, S. Chen, G. Yan, Z. Zhang, BotTracer: execution-based bot-like malware detection, in: Proceedings of the Information Security Conference (ISC), 2008.
[34] P. Vixie, S. Thomson, Y. Rekhter, J. Bound, Dynamic updates in the domain name system (DNS update). <http://www.faqs.org/rfcs/rfc2136.html/>, 1997.
[35] J. John, A. Moshchuk, S. Gribble, A. Krishnamurthy, Studying spamming botnets using botlab, in: Proceedings of the Usenix Symposium on Networked Systems Design and Implementation (NSDI), 2009.
[36] K. Chiang, L. Lloyd, A case study of the rustock rootkit and spam bot, in: Proceedings of the Workshop on Hot Topics in Understanding Botnets (HotBots), 2007.
[37] W. Yih, C. Meek., Learning vector representations for similarity measures, Technical Report MSR-TR-2010-139, Microsoft Research.
[38] S. Gündüz, M.T. Özsu, A Poisson model for user accesses to web pages, in: ISCIS: Computer and Information Sciences, 2003.
[39] G. Marsaglia, L.-H. Tsay, Matrices and the structure of random number sequences, Linear Algebra and its Applications 67 (1985) 147–156.
[40] D. Dagon, C. Zou, W. Lee, Modeling botnet propagation using time zones, in: Proceedings of the Annual Network and Distributed System Security Symposium (NDSS), 2006.
[41] D. Pelleg, A.W. Moore, *X*-means: extending *k*-means with efficient estimation of the number of clusters, in: Proceedings of the International Conference on Machine Learning (ICML), 2000.
[42] J. Ma, L.K. Saul, S. Savage, G.M. Voelker, Beyond blacklists: learning to detect malicious web sites from suspicious URLs, in: KDD: Proceedings of the international conference on Knowledge Discovery and Data mining, 2009.

[43] D.K. McGrath, M. Gupta, Behind Phishing: An Examination of Phisher Modi Operandi, in: LEET: Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats, 2008.
[44] GeoIP API, MaxMind, LLC, Open source APIs and database for geological information. <http://www.maxmind.com/app/api>, 2002.
[45] T. Holz, C. Gorecki, K. Rieck, F.C. Freiling, Detection and mitigation of fast-flux service networks, in: Proceedings of the Annual Network and Distributed System Security Symposium (NDSS), 2008.
[46] Korea Information Security Agency (KISA), Botnet C&C server domain list, 2009.
[47] DNS-BH, Malware Prevention through Domain Blocking (Black Hole DNS Sinkhole). <http://www.malwaredomains.com/>, 2007.
[48] Cyber-TA, SRI Honeynet and BotHunter Malware Analysis Automatic Summary Analysis Table. <http://www.cyber-ta.org/releases/malware-analysis/public/>, 2005.
[49] McAfee SiteAdvisor, Service for reporting the safety of web sites. <http://www.siteadvisor.com/>, 2006.
[50] WOT, Web of Trust Community-based safe surfing tool. <http://www.mywot.com/>, 2006.
[51] Domaincrawler, Domain Information Services. <http://www.domaincrawler.com/>, 2006.
[52] T. Holz, M. Steiner, F. Dahl, E. Biersacky, F. Freiling, Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm, in: Proceedings of the Workshop on Large-scale Exploits and Emergent Threats (LEET), 2008.
[53] S. Herona, Working the botnet: how dynamic DNS is revitalising the zombie army, Network Security 2007 (2007) 9–11.
[54] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, T. Wobber, How dynamic are ip addresses?, in: Proceedings of the ACM SIGCOMM Conference on Data communication (SIGCOMM), 2007.
[55] M. Feily, A. Shahrestani, S. Ramadass, A survey of botnet and botnet detection, in: Proceedings of the Conference on Emerging Security Information, Systems, and Technologies, 2009.

Hyunsang Choi received the B.S. and M.S. degree in computer science and engineering from Korea University in Seoul, Korea, in 2007 and 2009, respectively. He is currently working toward doctorate degree in computer and communication security at Korea University in Seoul, Korea. He was an intern at Microsoft Resarche Asia (Beijing, China) from 2009 to 2010.

Heejo Lee is an associate professor at the Division of Computer and Communication Engineering, Korea University, Seoul, Korea. Before joining Korea University, he was at AhnLab, Inc. as a CTO from 2001 to 2003. From 2000 to 2001, he was a postdoc at the Department of Computer Sciences and the security center CERIAS, Purdue University. Dr. Lee received his BS, MS, PhD degree in Computer Science andEngineering from POSTECH, Pohang, Korea. Dr. Lee serves as an editor of Journal of Communications and Networks. He has been an advisory member of Korea Information SecurityAgency and Korea Supreme Prosecutor's Office.