Computer Networks 97 (2016) 48–73

Contents lists available at ScienceDirect

Computer Networks

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

PsyBoG: A scalable botnet detection method for large-scale DNS traffic

Jonghoon Kwon[a], Jehyun Lee[a], Heejo Lee[a,*], Adrian Perrig[b]

[a] Korea University, Seoul 136-713, Republic of Korea
[b] ETH, Zurich 8092, Switzerland

ARTICLE INFO

Article history:
Received 15 April 2015
Revised 5 December 2015
Accepted 21 December 2015
Available online 12 January 2016

Keywords:
Network security
Botnet detection
DNS analysis
Power spectral density
Group activity

ABSTRACT

Domain Name System (DNS) traffic has become a rich source of information from a security perspective. However, the volume of DNS traffic has been skyrocketing, such that security analyzers experience difficulties in collecting, retrieving, and analyzing the DNS traffic in response to modern Internet threats. More precisely, much of the research relating to DNS has been negatively affected by the dramatic increase in the number of queries and domains. This phenomenon has necessitated a scalable approach, which is not dependent on the volume of DNS traffic. In this paper, we introduce a fast and scalable approach, called PsyBoG, for detecting malicious behavior within large volumes of DNS traffic. PsyBoG leverages a signal processing technique, power spectral density (PSD) analysis, to discover the major frequencies resulting from the periodic DNS queries of botnets. The PSD analysis allows us to detect sophisticated botnets regardless of their evasive techniques, sporadic behavior, and even normal users' traffic. Furthermore, our method allows us to deal with large-scale DNS data by only utilizing the timing information of query generation regardless of the number of queries and domains. Finally, PsyBoG discovers groups of hosts which show similar patterns of malicious behavior. PsyBoG was evaluated by conducting experiments with two different data sets, namely DNS traces generated by real malware in controlled environments and a large number of real-world DNS traces collected from a recursive DNS server, an authoritative DNS server, and Top-Level Domain (TLD) servers. We utilized the malware traces as the ground truth, and, as a result, PsyBoG performed with a detection accuracy of 95%. By using a large number of DNS traces, we were able to demonstrate the scalability and effectiveness of PsyBoG in terms of practical usage. Finally, PsyBoG detected 23 unknown and 26 known botnet groups with 0.1% false positives.

1. Introduction

Botnets have become one of the most significant Internet threats over the last decade [1]. Since the Internet has become a conflict zone in which the cyber criminals and the defenders of the Internet engage in an endless war, the attack capability is rapidly escalating to attain worldwide influence. In this conflict, botnets are emerging as the top instrument with which cyber attacks are committed [2,3].

A bot is a malware remotely controlled by an attacker known as a botmaster. A botnet is a network of bot-infected computers. The nature of botnet architecture provides the botnet with tremendous attack power by simultaneously mobilizing the capability of the members of a bot army to attack a target. Moreover, the botnet is the Swiss Army knife of attackers because of its usability. A botnet attack can perpetrate to distributed denial-of-service (DDoS), identification theft, ad-ware installation,

* Corresponding author.
E-mail address: heejo@korea.ac.kr (H. Lee).
URL: http://ccs.korea.ac.kr (H. Lee)

http://dx.doi.org/10.1016/j.comnet.2015.12.008
1389-1286/

mass spamming, and so on. Botnets are widely used not only for acquiring economic profits, but also to elicit military or political advantages from individuals, organizations, and even governments, by leveraging their unlimited attack capability and usability [4,5].

Internet threat report [6] foresees a continuous increase in botnet activities in 2015. The report also stress the fact that new and unseen types of botnets are continuously being developed. Botnet countermeasures are evaded in that bot authors continuously research and adopt state-of-the-art techniques to develop the bot codes. Therefore, botnets are evolving to be more complicated, sophisticated, and intelligent. In addition, along with the rise of the Internet of Things (IoT), attackers are expanding their territory to include billions of smart devices, such as smart phones, smart TVs, and smart meters [7,8]. As a consequence of the high infection rates achieved by botnets, together with their capability of conducting wide-range network attacks, and evadability for traditional detection methods, botnets are considered a major threat to Internet security.

In response to botnet threats, many methods and approaches have been suggested to detect botnets and these can be classified into two categories: host-based detection and network-based detection. Host-based detection methods mainly aim to analyze the internal components of a computer system [9,10], by allowing users to easily detect abnormal activities in the system. However, host-based methods have limitations, such as the difficulties associated with their deployment to cover a wide range of hosts, because of the overhead resulting from the attempts to detect the bots.

Network-based detection monitors the network traffic at servers and routers, thereby avoiding the abovementioned coverage problem. Early researchers focused on the contents of a botnet [11,12]. However, as botnets apply the techniques of encryption and obfuscation, detecting botnets with content-based methods became difficult. Traffic pattern-based detection, which analyzes the patterns of network traffic generated by the infected host, was suggested as an alternative [13,14]. However, these detection methods have a low detection rate, as a result of the huge amount of traffic generated by normal users. The ***sizable volume*** of traffic generated in the network also present a significant problem. In addition, botnet behavior could be very sporadic, either because hosts are periodically turned off or because of the nature of botnets, and this complicates their detection even more.

Some botnet detection research has focused on Domain Name System (DNS) traffic. For example, BotGAD [15] detects botnets by searching groups of computers submitting similar domain queries during a certain period of time. Unfortunately, botnet authors counter the detection by using domain fluxing schemes, such as fast flux, dynamic DNS, and DGA, which are advanced DNS techniques that generate a huge number of domain names [16]. In fact, Domain Generation Algorithms (DGAs) [17,18] are widely used in various bot codes for bypassing existing domain-based detection systems (*i.e.*, domain blacklists) [19]. DGAs function by periodically generating a large number of domain names that can be used as rendezvous points between the bot hosts and Command and Control (C&C) servers. Attempts to mitigate the problems caused by DGAs have resulted in numerous research efforts. For example, R. Begleiter et al. [20] proposed a DGA detection scheme using language modeling to discover randomly generated unreadable domains. However, these researchers encountered a scalability issue caused by the large number of domain names that had to be processed, as their approach is purely based on the analysis of domain names. Consequently, their approach is no longer sufficient to respond to the sophisticated botnet threats.

The limitations of current countermeasures necessitate the development of a new approach to defeat botnets that generate a large number of domains using domain fluxing. As botnets characteristically spread across wide areas and hide their behavior among the huge quantities of data from ordinary users, leveraging the prevalence of large amounts of traffic has presented itself as a promising approach for dealing with sophisticated botnet threats. The main challenge presented by this approach would be to design a scalable system capable of monitoring, collecting, and analyzing the big data input to extract the essential information. In this paper, we propose a scalable approach, named PsyBoG, to discover botnet groups by analyzing their **periodic** and **simultaneous behavioral patterns** using **DNS traffic**, which are the essence of intrinsic botnet characteristics appearing in large-scale network traffic. Our main goal is to build a scalable method capable of detecting botnet groups by analyzing the periodic and similar behavioral patterns of DNS traffic without any prior knowledge of the botnet.

Periodic behavioral pattern: At first, we observe the periodic behavior of botnets. More precisely, bot hosts regularly communicate with C&C servers or other bot hosts to maintain availability and capability of the botnet in peacetime. When the botnets launch attacks, such as mass spamming, click fraud, and DDoS attacks, each of these attacks causes a huge and regular amount of network traffic. These activities reflect the periodicity of botnet behavior, which is one of its unalterable characteristics. PsyBoG leverages power spectral density (PSD) analysis, which is a signal processing technique, to discover the major frequencies of periodic botnet behavior. The PSD analysis addresses the problems associated with the sporadic nature of botnet behavior and the deterrent caused by traffic generated by normal users.

Simultaneous behavioral pattern: Second, PsyBoG functions by discovering botnet group activities. A botnet is a group of compromised machines controlled by an attacker, the so-called botmaster. Bot hosts can be expected to exhibit similar behavioral patterns. More sophisticated botmasters are able to divide the entire group of bot hosts into multiple subgroups. However, bot hosts from the same subgroup still show similar behavioral patterns, because the fundamental aim of a botnet is to commit cyber-crimes, which can only be accomplished by using a large amount of resources. We define these similar behavioral patterns as a group activity, which forms an inherent property of the botnet. Thus, PsyBoG groups hosts in accordance with the similarity of traffic patterns.

DNS traffic: Third, PsyBoG leverages DNS traffic to measure the periodicity of query patterns, but it does not rely

on the domain names in the DNS traffic. Unlike previous researches or existing countermeasures such as DNS sinkhole and blacklist filtering, PsyBoG only utilizes the timing information of domain queries irrespective of the amount of DNS traffic, therefore, PsyBoG is designed to process large-scale DNS traffic. The periodicity of DNS query patterns is typically reflected by botnet traffic. Thus, the problems presented by traffic encryption and obfuscation, fast flux, dynamic DNS, or DGAs no longer exist. Furthermore, only three entities extracted from DNS traffic, including time for query, target domain, and queried IP address, are leveraged for the analysis. This approach makes PsyBoG extremely fast and scalable.

The concept of applying signal processing techniques to identify an abnormal behavior in network is not a new idea. Since Mitra et al. [21] suggested the notion that signal processing techniques can contribute to improve the internet security by identifying abnormal network behaviors caused by malware, some researchers have proposed methods aiming the discovery of the presence of botnet using the techniques analyzing repeating patterns of attack packets and flows [22–26]. However, as we mentioned above, most of the researches are still suffering from the scalability problem due to the large quantities of network traffic to be analyzed. Unlike previous botnet countermeasures which also apply signal processing techniques, PsyBoG solely deals with DNS traffic thereby solving the scalability problem. Furthermore, it is more convenient to establish the monitoring coverage both logically and physically by harvesting traffic from DNS servers for desired network areas, whereas other methods have difficulties to determine appropriate monitoring spot where network traffic coming from monitoring network is aggregated.

PsyBoG was evaluated by performing experiments with two different data sets. At first, we utilized DNS traces generated by real malware samples in a controlled environment. As it is hard to estimate the detection accuracy from real-world DNS because of the uncertainty as to what to use as the ground truth, we assessed the detection accuracy by using the first DNS traces as the ground truth. In the first experiment, PsyBoG showed 95% detection accuracy with its robustness. Second, we utilized large-scale DNS data collected from three different types of real-world DNS servers. These data sets were used to evaluate the detection performance, scalability, and overheads of PsyBoG. As a result, PsyBoG detected 23 unknown botnets and 26 known botnets with 0.1% of false positives and a relatively small overheads, which indicates the guaranteed scalability of PsyBoG for large quantities of DNS traffic. The experiments revealed the efficiency of PsyBoG as an effective countermeasure to mitigate current botnet threats.

In summary, this paper makes the following contributions:

- **High scalability**: PsyBoG only uses the timing information of each query. While given DNS traffic to be analyzed is increased linearly, the workload of PsyBoG increases in logarithm scale. This indicates the scalability of PsyBoG.
- **Robustness**: PSD enables the extraction of periodic patterns from among the mixture of noisy signals. This allows PsyBoG to distinguish botnets regardless of normal user traffic.
- **High detection accuracy**: PsyBoG is an effective countermeasure against sophisticated botnets which utilize evasion techniques such as payload encryption, frequent change of C&C communication patterns, and domain fluxing.
- **Applicability**: PsyBoG does not require any prior knowledge of botnets, such as the binary signatures, traffic signatures, and training data.

The rest of the paper is organized as follows. Section 2 describes the background of our mechanism including the basic knowledge of DNS, characteristics of botnet behaviors and PSD analysis. Section 3 describes the problem statement and requirements for botnet detection. Section 4 introduces our mechanism, and Section 5 outlines the experimental settings and results. In Section 6, we discuss potential techniques that may exploit our mechanism and further research topics. Section 7 describes related work, and finally we conclude this paper with the outline of future work in Section 8.

2. Background

We introduce the background knowledge of our work: DNS, botnet characteristics and signal processing techniques.

2.1. Domain Name System (DNS)

DNS is a large-scale distributed database for translating domain names into IP addresses and vice versa. The name servers have a hierarchical structure consisting of ROOT, Top Level Domains (TLDs), Second Level Domains, and Third Level Domains. If a domain query arrives at a local name server, the server firstly performs a lookup in the zone file of the domain storing the domain-IP mapping information. As the local name server is not an authoritative name server, the information temporarily survives in the zone file until its time to live (TTL) is expired. If the local name server cannot find any corresponding data for the query, in the worst case, the server recursively requests the information from the ROOT to the Third Level Domain name server. Finally, the local name server updates its zone file to record the information and replies to a resolver.

DNS provides the domain-IP matching service for any systems connected to the Internet. When a domain-based Internet connection is established, a query for a certain domain name is first generated to obtain the corresponding IP address. Therefore, a DNS server is the gateway for an Internet connection. Because botnets base their operations on domain-based Internet services, DNS servers form a very important location in terms of botnet operations. Now, we briefly provide basic information pertaining to DNS, especially, information relating to botnet operation and mitigation.

- **Recursive DNS server (RDNS)**: RDNS is a local name server responsible for providing the IP address corresponding to an intended domain name to requesting

hosts in the local area covered by the RDNS. Recursive DNS servers are usually managed by ISPs or organizations with the authority to manage local networks. RDNS is a very advantageous point to meet a large number of hosts and comprehend the circumstances of the hosts. Consequently, RDNS is an important asset to mitigate botnet threats.

- **Authoritative DNS server (ADNS)**: An ADNS server stores a list of the IP addresses of domain names, whereas the RDNS only temporarily stores the mapping information as a representative. ADNS servers are usually operated by companies which own certain domains. Every single query for domains belonging to an ADNS server has to be congregated to the server, regardless of where the queries originated, *e.g.*, United States, Switzerland, Ghana, or Australia. This is highly advantageous for security responders, because it allows single-point monitoring and mitigating in terms of worldwide influence.

As DNS forms a very important part of botnet operations, botnets utilize numerous advanced DNS techniques to avoid the *single-point failure* problem caused by C&C domain detection. The following are the representative DNS techniques used by sophisticated botnets.

- **Dynamic DNS (DDNS)**: DDNS is a method that allows real-time updating of the domain-IP mapping data [27]. It is used to resolve domain names to IP addresses, which may change frequently. It allows persistent addressing for domains that need to change their server or location very often. Therefore, the DDNS method is very useful for small businesses that need to provide consistent services, because these businesses use a dynamic IP range assigned by their Internet Service Provider (ISP). Recent botnets are designed to abuse the DDNS method to increase the survivability of their C&C servers. Traditional botnets experienced the *single-point failure* problem, because they usually assigned a single IP address to the C&C server. Once the IP address was revealed, the entire botnet could be collapsed by simply blocking the IP address. From this, DDNS has emerged as an effective evasion technique to botnet users.
- **Fast flux**: Fast flux is a DNS technique that is used by botnets to hide their rendezvous points behind proxy servers (bot hosts) [28]. The key idea is to assign numerous IP addresses to a single domain name, where the IP addresses are changing very frequently because of the use of a combination of round-robin (RR) IP addresses and a relatively short TTL. Although the method seems similar to that of DDNS, fast flux and DDNS are based on slightly different concepts. Most importantly, the authoritative name servers of a DDNS domain belong to the DDNS provider, whereas fast flux is able to make the name servers point to numerous IP addresses of hosts located worldwide.
- **Domain Generation Algorithm (DGA)**: DGA is another advanced DNS technique used by sophisticated botnets [16]. Even though the DDNS and fast flux methods are able to improve the survivability of botnets, they are still suffering from the *single-point failure* problem caused by blocking certain domains (*i.e.*, a domain sinkhole). To overcome the problem, DGA has arisen as a more advanced evasion technique for recent botnets. DGA periodically generates thousands of domain names, which can be used as rendezvous points for C&C communication. Among these domains, only a few are used as actual C&C domains at a certain moment. The real C&C domains only live for short periods before they are replaced with other domains; thus, if the C&C domains were retained by the responders, the botnets would persist. The large number of potential C&C domains complicates taking down the botnets.

2.2. Botnet characteristics

A bot is a malware remotely controlled by an attacker (botmaster). A botnet consists of a group of bot-infected hosts forming a network, and it provides cyber criminals with unlimited resources to enable them to commit serious Internet attacks. Despite a considerable amount of research to develop ways to mitigate botnet threats, botnets are regarded as one of the most significant Internet threats. The mitigation of botnet threats requires us to understand the pure nature of a botnet. Now, we describe three inherent features of botnets: utilization of DNS, periodic communication, and the group activity of bot hosts.

- **Utilization of DNS**: Hosts infected by bots use DNS to access the C&C servers. Because the IP addresses of C&C servers used by bot authors have been disclosed by reverse engineering, they attempt to use domain names instead of static IP addresses. Furthermore, they prevent detecting and blocking of the servers by periodically changing the IP addresses and domain names of the C&C server by using techniques such as fast flux or DDNS.
  Generally, a bot transmits DNS queries to maintain contact with the C&C servers. More intelligent botnets use DDNS to hide their query patterns. A C&C server using DDNS changes its IP address frequently and has smaller TTL values in its DNS record, resulting in more frequent DNS queries from the bots. However, in cases in which data stored in the local cache is used, the DNS query is invisible as it is not transmitted outward.
- **Periodic communication**: A bot program is compelled to periodically communicate with the C&C server [22]. This is because the connection with the C&C channel is necessary for the C&C server to check the host status or to issue an attack. The periodic connection guarantees the availability and capability of the botnet. A bot-infected machine automatically accesses the C&C server of the botnet. The bot host queries the DNS server using predefined domain names to obtain the IP address of the C&C server, to which it periodically reports its status.
  More intelligent botnets change the IP addresses of their C&C servers occasionally by using the DDNS and fast flux techniques. In such cases, the DNS servers maintain the domain-IP mapping data with small TTL values, and consequently, more frequent DNS queries can be observed. A DGA also shows periodic DNS query generation. As the DGA algorithm automatically

generates thousands of domain names and only a few domains among the large number of domains are real C&C servers, bot hosts generate a large number of queries to find the real C&C channel.

- **Group activity**: Botnet communications are observed in the form of group activities. A botnet requires certain predefined rules to manage several hundreds, or even thousands, of bot hosts. A centralized botnet (IRC, HTTP) uses DNS to look up the C&C server. The botnet sends periodic DNS queries and connects to the channel. This is a good example of a group activity.
  A distributed (P2P) botnet performs group activities, which are observed while the P2P botnet performs an upgrade or synchronization. For example, the storm P2P botnet often synchronizes with the network time protocol server through the infected hosts. This synchronization activity also shows a group activity. When a botnet behaves maliciously, including performing a DDoS attack, spamming, and click fraud, each bot host simultaneously generates a massive amount of attack traffic to ensure a more efficient and effective attack. For example, a DDoS attack requires numerous bot hosts to launch concurrent attacks on target systems.

2.3. Signal processing techniques

PsyBoG utilizes a signal processing technique for extracting the periodic communication pattern in a botnet. Signal processing is the operation involving the analysis of analog and digitized signals. One of the typical operations in signal processing is to extract frequencies from a given sequence of signals.

A discrete Fourier transform (DFT) operation is used to convert a discrete-time domain signal, such as a time series, to frequency domain data as a sum of the sinusoidal components (*sine* and *cosine*). The frequency domain data contain the amplitudes of each frequency. A fast Fourier transform (FFT) is an efficient algorithm capable of conducting a DFT and its reverse execution in a short time. The time complexity of DFT is $O(N^2)$, whereas FFT shows a time complexity of $O(Nlog_2N)$.

$$F_N = \sum_{n=1}^{N} f_n \cdot e^{-i2\pi kn/N}. \quad (1)$$

We input the sequence of data points (time series) $f(1)$, $f(1)$, $f(2)$, ..., $f(N)$ into Eq. (1), where N (2^n) is the size of the entire data, n is the index value, and k is the frequency which needs to be known. The transformation to the frequency area result consists of the complex numbers of $F(1)$, $F(1)$, $F(2)$, ..., $F(N)$.

$$P_{xx}(\omega) = \frac{(\Delta t)^2}{T}\left|\sum_{n=1}^{N} f_n e^{-i\omega n}\right|^2. \quad (2)$$

We assume that the periodic pattern displayed by the host traffic can be determined from the high amplitudes of certain frequencies. Second, Eq. (2) is the definition of the PSD. The PSD uses a straightforward manner to generalize the finite time-series f_n with $0 \le n \le N$, such as a signal sampled at discrete times $f_n = f(n\Delta t)$ for a total measurement period $T = N\Delta t$. $P_{xx}(\omega)$ is the average of the Fourier transform magnitude squared and ω is $2\pi k/N$. The PSD describes how the power of a signal or time series is distributed with a unit of energy per frequency. The power can be defined as the squared value of the signal.

3. Problem definition

This section presents a description of the problem to be considered when detecting botnets, following which we suggest the requirements for solving the problem along with the goal of our study.

3.1. Problem statement

We state the following three problems for botnet detection.

- **The huge volume of DNS traffic**: As the volume of DNS traffic skyrockets, previous anomaly detection mechanisms require more resources and time to thoroughly analyze the huge volume of traffic. Currently, the increase in the volume of Internet traffic is usually the consequence of an increase in the number of domain names rather than the number of IP addresses. Unfortunately, many of the countermeasures and much of the research rely on domain-based analysis, and are therefore negatively affected by the increase in the number of domains. Consequently, scalability has become an important issue.
- **User traffic**: A botnet consists of a group of compromised hosts; hence, there is a great possibility that benign users are also involved in DNS query generation. Furthermore, sophisticated botnets mimic user behavior by generating bogus DNS queries with well-known domain names. From the point of view of a network, distinguishing between legitimate and malicious DNS queries is a great challenge because of the lack of internal information.
- **Host detection**: Detecting a botnet, either individually or in part, is insufficient to mitigate the threats it poses. Because a botnet consists of a group of compromised hosts, it is difficult to prevent botnet attacks without denying access to the entire host group. Unfortunately, many botnet mitigation methods insist on targeting particular domains or IPs. As a result, these methods are not able to effectively prevent botnet attacks.

3.2. Requirements

Previous studies were unable to provide effective solutions to the abovementioned problems. In this work, the following requirements are addressed in an attempt to overcome these problems.

- **Scalability for data size increments**: Motivated by situations in which the DNS traffic volume increases rapidly, a new approach capable of managing huge quantities of traffic is required by considering scalability.

- **Robustness against user traffic**: In terms of bot host detection, it is never possible to distinguish between different types of user traffic originating from the same host. Therefore, a new method should be able to distinguish the characteristics of botnet traffic from among the combined traffic.
- **Effective group detection**: Botnet group has to be detected through their group activities. Botnet attacks can be prevented by denying access to the botnet group. This requires the necessary capability to detect entire botnet group.

3.3. Goal

Our goal is to suggest a fast and scalable method capable of detecting botnet groups by scanning the periodic pattern of traffic without any prior knowledge of the botnet, in an efficient manner using only essential information of the large-scale DNS traffic.

4. Proposed mechanism

In this section, we introduce the concept and structure of our mechanism, which we have named PsyBoG (Power Spectrum analYsis for detecting Botnet Groups). Then, we explain the operations of PsyBoG in detail.

4.1. Overview

The periodicity of a bot host can be extracted from the DNS traffic using PSD. Botnet groups can be detected by performing a similarity measurement based on the periodicity of bot hosts. Fig. 1 shows the structure of PsyBoG, which consists of the following four modules.

- **Traffic collector** collects DNS traffic, such as host IPs, domain names, and query time stamps from the DNS servers.
- **Periodicity analyzer** uses PSD to extract the frequency information of host DNS traffic.
- **Significant peak analyzer** analyzes the significance of peak values in a power spectrum. If a peak crosses the significance threshold, PsyBoG determines that the host contains very suspicious periodic query patterns.
- **Group activity analyzer** analyzes the similarities between the power spectrums of the hosts, which contain a significant periodic component. If these power spectrums show a high similarity rate with each other, it indicates that they have similar periodic query patterns and belong to the same botnet group.

Fig. 2 illustrates the flowchart of the main components of PsyBoG including periodicity analyzer, significant peak analyzer and group activity analyzer. Detailed descriptions for each component will be shown in the following sections.

4.2. DNS traffic collector

The sensors collect the DNS traffic from a monitored network by tapping DNS servers and aggregating the DNS traffic to the DNS traffic collector. Bot infected hosts could widely spread across the entire world. An effective response to the botnets necessitates a scalable detection method covering a wide network range. Given these considerations, the following two requirements are addressed:

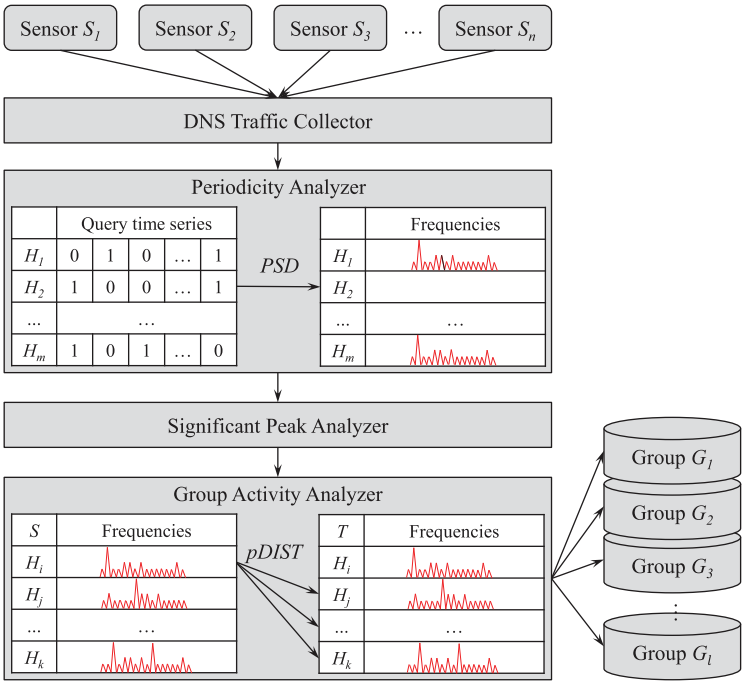


Fig. 1. Design of PsyBoG architecture.

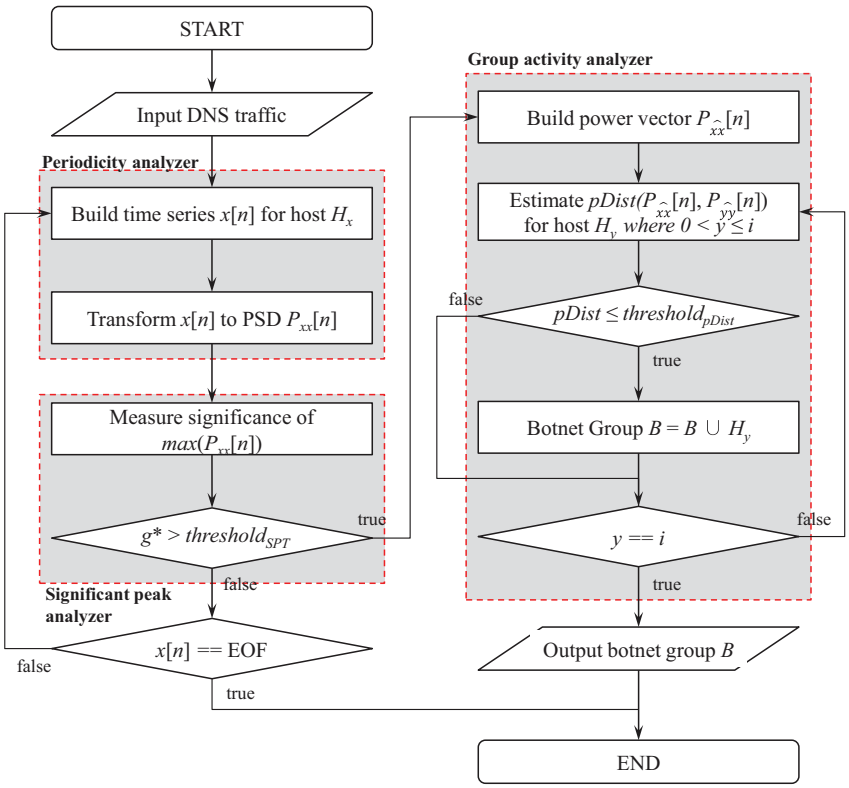


Fig. 2. Flowchart of the main functionality of PsyBoG.

(1) single-point monitoring to enable a wide network range to be observed, and (2) reducing a huge volume of data to an amount that is practically manageable.

Driven by the above requirements, we specify that PsyBoG only uses DNS traffic. The main reasons for this are threefold. First, in accordance with the nature of the DNS system, the DNS server is an incredibly efficient asset with which to monitor a wide network. the RDNS and ADNS servers are responsible for the local network (*i.e.*, an ISP network) and could even be responsible for the entire Internet. Second, DNS closely relates to botnet activities. Numerous botnets apply highly advanced DNS techniques, such as fast flux, DDNS and DGAs, to avoid detection of the C&C and spam servers; they show a higher rate of DNS usage compared to normal users or systems. Third, the amount of DNS traffic on a network is less than the entire network traffic from the same network range.

The malicious attempts of the botnet can be prevented by denying access to the DNS. As the C&C server periodically changes its IP address, blocking the DNS queries of botnet hosts disables these hosts by preventing them from accessing the server as they do not know its address. However, this may result in the problem of DNS caching. The DNS cache can vary with different operating systems or browsers. Therefore, it is necessary to consider the caching procedures of the different kinds of operating systems and browsers. Meanwhile, the advanced DNS techniques that are used by the newer botnets maintain a shorter TTLs; thus, the DNS caching problem will not occur.

Even though leveraging DNS traffic is a very efficient way in terms of botnet detection and mitigation, the amount of DNS traffic that would have to be analyzed remains too large. To ensure more efficient usage of the DNS traffic, PsyBoG is designed to only analyze queries. Furthermore, the sensors only aggregate the host IP, the domain name, and its time stamp, which are extracted from the DNS traffic. This approach renders PsyBoG scalable for processing large volumes of DNS data by rapidly reducing the volume of data to be analyzed.

4.3. Periodicity analyzer

We can analyze the periodicity of the botnet communications by using PSD. When using this analytical technique, we assume that the periodic traffic of a botnet is transformed into certain high-power frequencies, while the aperiodic traffic caused by normal users is transformed into a broadly distributed rage of low-power frequencies. Therefore, botnet traffic can be extracted by scanning for certain high-energy frequencies, and we use PSD to transform signal-time data into frequency data.

First, we specify a number of segments to use as input time series for the PSD analysis. The number of segments affects the frequency domain and its range. For a high-quality PSD, the number of segments is selected from among the powers of two, and we limit the length of the number of segments to $2^{14} = 16{,}384$ to ensure a fast PSD analysis. Note that the size of a single segment is 1 s,

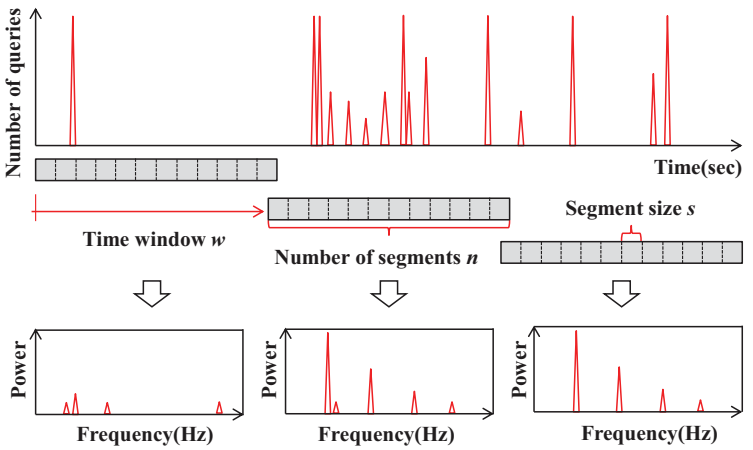


Fig. 3. Concepts of time window and segmentation.

and we apply the sliding time-window strategy to cover a long input trace. Fig. 3 briefly shows the concept of the segments, time window, and sliding window. The second step in the PSD estimation is to remove the mean value of the Fourier mode from the time series. This is a standard technique [29] that allows a more accurate PSD estimation. In the third step, we use a Hanning window that is used on half-overlapped intervals to ensure the best signal-to-noise ratio (SNR). The last step consists of operating the PSD analysis for the specified segments of the input time series.

Fig. 4 represents an example of the periodic and aperiodic signals and their periodograms, which project the respective corresponding PSDs for the signals. The first plot shows a periodic query sample in which the period of the query is 10 s, the duration of each query is 3 s, and the number of segments is 256. The second plot, which is a periodogram of the first plot, consists of the largest peak at 0.1 Hz, small peaks at multiples of the largest peak 0.1 Hz and almost at zero, which indicates that the original signal has a periodic query pattern within every 10 s. The third plot contains an aperiodic signal that follows a Gaussian random distribution, and its associated periodogram, the last plot, contains several peaks, but none of them has sufficiently large power; thus, there is no periodicity. Note that the decision as to whether a peak value is sufficiently large, will be explained in the following section.

4.4. Significant peak analyzer

As we can see in Fig. 4, there is always a peak in the periodogram regardless of whether the original signal contains a periodic pattern. Therefore, the decision as to whether the peak is sufficiently significant to determine whether it is caused by a periodic component with a specific frequency, has to be made. To this end, we use significant peak testing (SPT) by applying the significant test method which is commonly used in signal processing and communications.

The design of a critical test for significant frequencies has long been researched in signal processing area. Since the introduction of the periodogram by Schuster in 1898 [30], the investigation of periodicity in time series has widely been researched. Walker [31] studied the statistical significance of the periodicities for large sample test. Fisher [32] introduced the exact probability distribution indicating that it is not required to use the large sample asymptotic assumptions at the Walker's test. Further researches by Lomb [33], Scargle [34], and Horne and Baliunas [35] have been presented as the key papers. Among them, we consider that Walker's large sample test would be more appropriate for PsyBoG rather than other tests, because Walker's test is known to be more accurate for large sample data [36,37] and the most important contribution of PsyBoG is to guarantee the scalability for large scale network.

The extraction of the significance of the largest peak in periodogram performs as follows. Commonly, the significant test for periodogram ordinates is designed to work on the binary hypothesis [38,39]. For a query sequence $x[n]$,

H_0: x[n] follows a Gaussian distribution, while
H_1: x[n] has a periodic component at the largest ordinate.

For H_0, the ordinates $P_{xx}[\omega]$ for the sequence $x[n]$ has a distribution which is proportional to χ^2 in two degrees of freedom. Therefore,

$$P_{xx}[\omega] = \sigma_x^2 \chi_2^2 \tag{3}$$

The probability distribution of a χ_2^2 is an exponential function [40],

$$f(z) = 2^{-1} exp(-z/2) \tag{4}$$

Hence, for any value of $z \geq 0$, the probability that $P_{xx}[\omega]/\sigma_x^2 \leq z$ is given by,

$$\begin{aligned} Pr[P_{xx}[\omega]/\sigma_x^2 \leq z] &= \int_0^z f(x)dx \\ &= \int_0^z 2^{-1} exp(-x/2)dx \\ &= 1 - exp(-z/2). \end{aligned} \tag{5}$$

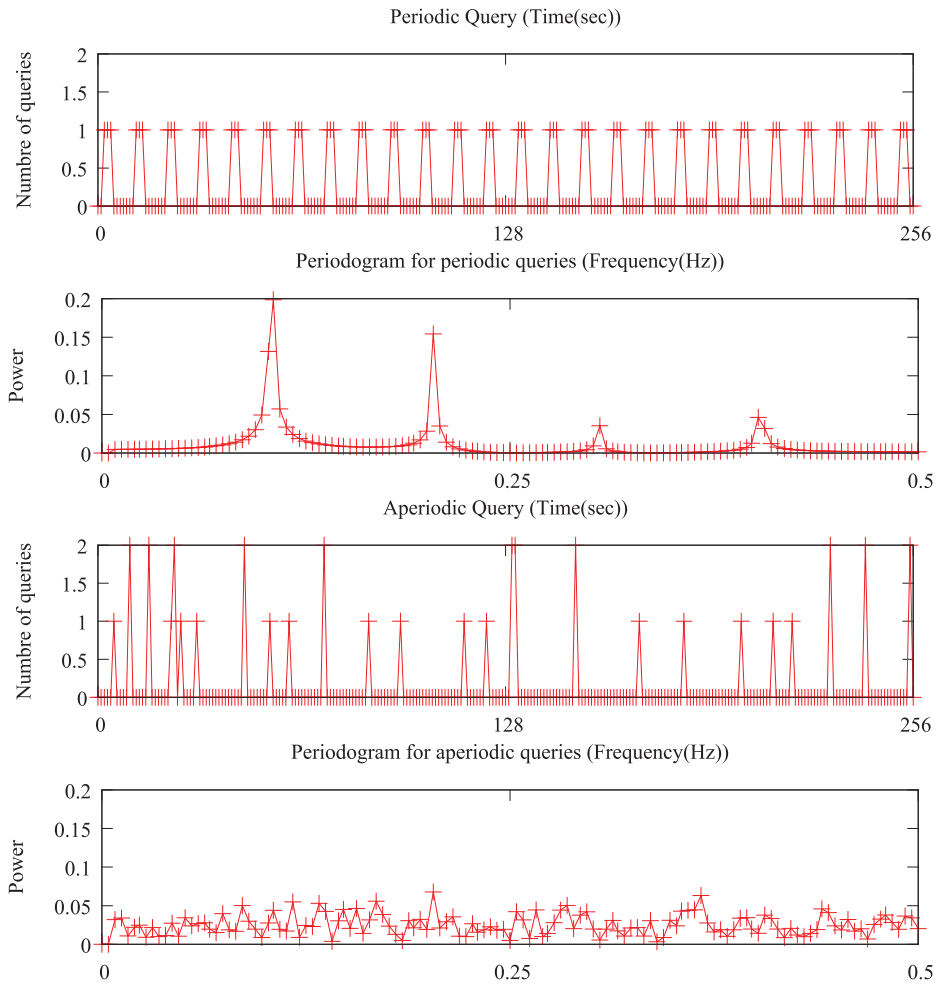


Fig. 4. Periodograms for periodic and aperiodic signals.

Under H_0 that γ_x indicates one of the $N/2$ independently identically distributed variables, then for any value of $z \geq 0$,

$$Pr[\gamma_x > z] = 1 - Pr[P_{xx}[\omega]/\sigma_x^2 \leq z, \text{for } \omega] = 1 - [1 - exp(-z/2)]^{N/2} \quad (6)$$

Eq. (6) can be used for determining whether the largest ordinate in the periodogram is significantly different from a zero mean distribution with variance σ_x^2, which can be evaluated directly by using Eq. (7):

$$\sigma_x^2 = N^{-1} \sum_{k=1}^{N/2} P_{xx}[\omega] \quad (7)$$

According to the above estimation of σ_x^2, we can derive the asymptotic distribution g_x^* from Eq. (3), which is Walker's large sample test for $max(P_{xx}[\omega])$,

$$g_x^* = \frac{max(P_{xx}[k])}{N^{-1} \sum_{k=1}^{N/2} P_{xx}[k]} \quad (8)$$

Under H_0, g_x^* will have the same distribution as γ_x, thus for $z \geq 0$,

$$Pr[g_x^* > z] \sim 1 - [1 - exp(-z/2)]^{N/2} \quad (9)$$

In order to be significant, g_x^* has to be larger than the critical value of:

$$z = -2\ln(1 - Pr_c^{2/N}) = -2\ln(1 - (1 - Pr_{FPR})^{2/N}) \quad (10)$$

Where Pr_c is the confidence level probability, $1 - Pr_{FPR}$, and Pr_{FPR} indicate the expected probability of false alarm. By simply applying this test to the example exhibited in Fig. 4, we obtain the following results. The critical value $z_{0.1\%}$ with the binary hypothesis for the expected false positive rate 0.1% is 23.52, where the number of segments $N = 256$. The periodic and aperiodic signals (first and third plots) return g_x^* as 35.34 and 5.39, respectively. This enables us to determine that the first plot has a periodic component at a frequency of $k = 0.1$Hz, whereas the third plot does not display any periodicity.

The critical value z is regarded as the threshold of SPT, $threshold_{SPT}$. Note that, as shown in Eq. (10), z can be

flexible according to the number of segments N and the probability of the expected false positive rate Pr_{FPR}. PsyBoG has the predefined values of N for input time series, therefore our only concern for appropriate z is the Pr_{FPR}. In order to set an optimized $threshold_SPT$, we performed an experiment and it will be discussed in Section 5.2.

4.5. Botnet group activity detection

PsyBoG investigates the activities associated with entire botnet group, because detecting the group is necessary for the efficient prevention of botnet threats. This is done by applying the similarity measurement algorithm pDist (Power Distance) [41] to detect the group activities of botnet. pDist compares the periodic structure of two input signals. More precisely, pDist utilizes the specific frequencies in the periodogram with sufficient power (Fig. 4). Suppose there are two distinct periodograms $P_{aa}[k_n]$, $P_{bb}[k_n]$ with length n, and it is determined that their largest ordinates exceed the threshold for significant testing. Now, we can obtain the frequencies with sufficient power values $p_a \subset [\{x_1, y_1\} \ldots \{x_i, y_j\}]$. Finally, we can simply compare the power values located at the frequencies f_a with f_b. The distance pDist represents the similarity between two signals a and b:

$$pDist = \| f_a - f_b \| \tag{11}$$

For example, three time series a, b and c exist and their Fourier transforms are $A = \{(1+i), (2+i), (1+i), (3+2i)\}$, $B = \{(1+i), (1+2i), (1+i), (4+i)\}$, and $C = \{(1+i), (1+i), (3+3i), (1+i)\}$ respectively. The periodogram of A can be represented as: $P_{aa} = \|A\|^2 = (2, 5, 2, 13)$, and its significant power vector would be $P_{\hat{aa}} = (0, 0, 0, 13)$. Similarly, we can derive the other significant power vectors for B and C as $P_{\hat{bb}} = (0, 0, 0, 16)$ and $P_{\hat{cc}} = (0, 0, 18, 0)$.

To make the distance measurement more meaningful, pDist was designed to perform the normalization of any sequence $x(n)$ for power vector P_{xx} as follows:

$$\hat{x}(n) = \frac{x(n) - N^{-1} \sum_{i=1}^{N} x(i)}{\sqrt{\sum_{i=1}^{N} (x(n) - N^{-1} \sum_{i=1}^{N} x(i))^2}} \tag{12}$$

In [41], the clustering accuracy of pDist was examined and compared with four other clustering approaches namely, Euclidean, DTW, Cepstrum, and CDM. Among them, pDist not only shows the highest accuracy but is also the most lightweight method, because pDist functions in a very low-dimensional space, precisely only on the i dimension.

Algorithm 1 describes a method for investigating botnet grouping with pDist. Once a certain number of hosts h_n are revealed to exhibit a significant periodicity on their queries, the group activity analyzer performs botnet grouping by comparing the periodograms f_n extracted from the hosts. Performing this comparison for every hosts might result in relatively large computation overheads of the order $O(n^2)$. Nevertheless, these overheads are nonnegotiable if each host faces the possibility of being compromised by a number of distinct bots. The bot hosts used to be compromised by different bot codes simultaneously due to the poor security awareness. Furthermore, the number of bot hosts included in the comparison could be reduced to a manageable size (please refer to Section 5.6). PsyBoG only considers hosts that show a significant periodicity, thus only a small number of hosts in a particular network is regarded as input for this analysis.

Algorithm 1 Botnet grouping.

Input: Frequencies f_n for hosts h_n
Output: Botnet group set $B = \{b\}$

```
k ← 0;
for i = 0 to n − 1 do
    for j = i + 1 to n do
        dist ← pDist(f_i, f_j);
        if dist ≤ threshold_pDist then
            b_k ← b_k ∪ {h_i, h_j};
        end if
    end for
    if |b_k| ≥ 0 then
        k ← k + 1;
        B = B ∪ b_k;
    end if
end for
```

5. Experimental results

In this section, we present the experimental results obtained by evaluating the performance of PsyBoG, including robustness, scalability, speed and detection accuracy. We implemented a prototype of PsyBoG using the *.NET* framework and collected the large number of real-world DNS traces for the experiments. The experiments were conducted on an in-lab machine with a 3.30 GHz Intel i5 CPU, 8 GB main memory and Microsoft Window 7 64bit. The PsyBoG prototype was implemented in a fully automatic manner, but we considered a manual investigation to verify the detection accuracy.

5.1. Data set

5.1.1. Collecting real-world data

The performance of PsyBoG was evaluated by collecting the real-world DNS traces, including malware traces, and real DNS server traffic. These data sets have different purposes. First, we verified the detection accuracy and robustness of PsyBoG by utilizing the DNS traces generated from the real malware samples, and second, we evaluated the scalability and overheads of PsyBoG with the large-scale DNS data collected from the real DNS servers.

Table 1 lists the detail of the DNS traces extracted from real malware samples. In the background of this paper, we explained that recent malware connected to the Internet generates DNS traffic very frequently and periodically. After obtaining our initial experimental results, we were strongly motivated by the need to prove how malware currently functions in the real world. To this end, we utilized the PCAP dumps provided by the Contagio Web site. The files store all the network traffic generated by real malware samples in controlled environments. We extracted the DNS traffic from the dump files, and selected the top 20 malicious DNS traces in the order of their size.

Table 1
Twenty DNS traces obtained from malware dumps.

Malware name	Tracing time (s)	Number of queries	Number of domains	DNS size	Original PCAP
BIN_Ramnitpcap..	66,469	60,116	382	2.87 MB	16.30 MB
Sality..	13,570 K	26,509	6341	1.08 MB	39.20 MB
BIN_Kuluoz-Asprox..	1171	6479	1334	279 K	25 M
purplehaze	2177	2689	786	114 KB	230 MB
BIN_Cutwail-Pushdo(2)..	1990	2260	188	83.10 KB	6.16 MB
BIN_Wordpress_Mutopy_Symmi..	140	1159	289	45.40 KB	1.68 MB
BIN_ZeroAccess_Sirefef..	941	1004	224	42.4 KB	11.7 MB
BIN_dirtjumper..	123	770	4	32.2 KB	1.57 MB
BIN_CitadelPacked..	90,312	662	4	26.0 KB	1.61 MB
cryptolocker..	169	524	130	24.6 KB	77.0 KB
Sweet-Orange-EK..	164	899	404	47.0 KB	557 KB
Angler-EK-uses-java..	71	572	285	31.7 KB	1.67 MB
Angler-EK-uses-silverlight..	52	564	280	31.3 KB	1.79 MB
Magnitude-EK..	350	628	251	26.4 KB	7.16 MB
phishing-malware..	231	437	219	24.7 KB	60 KB
Sweet-Orange-EK..(2)	161	416	206	24.5 KB	310 KB
Angler-EK-uses-flash..	35	412	203	22.7 KB	1.80 MB
Sweet-Orange-EK..(3)	92	414	207	22.6 KB	92 KB
phishing-malware..(2)	329	407	188	22.5 KB	393 KB
phishing-malware..(3)	630	336	163	18.9 KB	484 KB

Table 2
DNS data captured from real-world DNS servers.

Data set	Date	Time	Number of queries	Number of domains	Number of clients	DNS type
Campus1	Feb. 24, 2014	00:00–23:59	24,278 K	100 K	25 K	RDNS
Campus2	Feb. 25, 2014	00:00–23:59	23,768 K	93 K	23 K	RDNS
DDNS1	Jun. 15, 2010	00:00–23:59	15,563 K	1668 K	51 K	ADNS
DDNS2	Jun. 22, 2010	00:00–23:59	17,097 K	1578 K	50 K	ADNS
KrTLD1	Aug. 19, 2014	00:00–23:59	14,643 K	3583 K	64 K	TLD
KrTLD2	Aug. 19, 2014	00:00–23:59	33,937 K	2720 K	387 K	TLD

Evaluating PsyBoG in terms of scalability and applicability required us to utilize large-scale DNS traces captured from real DNS servers. Table 2 provides a brief overview of the distinct DNS data sets captured from different locations. *Campus1* and *Campus2* are DNS traces obtained by tapping a gateway router of a /16 campus network on Feb. 24 and 25, 2014. *DDNS1* and *DDNS2* were given by a company providing a dynamic DNS service in the US. These traces were captured on Jun. 15 and 22, 2010. *KrTLD1* and *KrTLD2* were collected from two different .kr TLD servers on Aug. 19, 2014.

The above data sets are distinguishable by the characteristics of the queries they contain; for example, *Campus1* and *Campus2* were collected from an RDNS server in a local network. These queries are only received from local hosts, but the queried domains may be located worldwide. On the other hand, the DDNS server is an ADNS server; thus, *DDNS1* and *DDNS2* were aggregated from a server containing queries received from hosts located across the world and the target domains could also be located in any country. Even though the last data sets were obtained from TLD servers that also function as an ADNS server, the IP range of target domains was limited to a certain region. We expected that the distinguishable data sets enable us to evaluate the effectiveness of PsyBoG under real-world circumstances.

5.1.2. Filtering

We applied both host and domain filtering for effective DNS traffic analysis. Because the amount of DNS data collected from real DNS servers is relatively large, analyzing every single host, even those hosts that rarely generate queries, would be a tedious task. Under the consideration, we decided to improve the efficiency of the data. As a preprocessing step, we removed the hosts that generated fewer than 13 DNS queries in DNS trace lasting one hour, because this number is too small to allow for the discovery of behavioral periodicity.[1] The preprocessing step enabled us to reduce the number of hosts to approximately 80% of the original number.

Domain queries listed on the whitelist are also excluded from our trace. In recent times, many benign programs periodically connect to specific domains to ensure they remain up-to-date, *e.g.*, Windows update, and AV update. Furthermore, well-known Web sites, such as Google, Facebook, and Twitter are queried very frequently. These well-known and thriving domains would therefore expose some periodicity. To this end, we built the whitelist to include top legitimate domains collected from Alexa.com [42].

[1] Note that, selection of the threshold 13 for host filtering will be discussed with the analysis in Section 6.4.

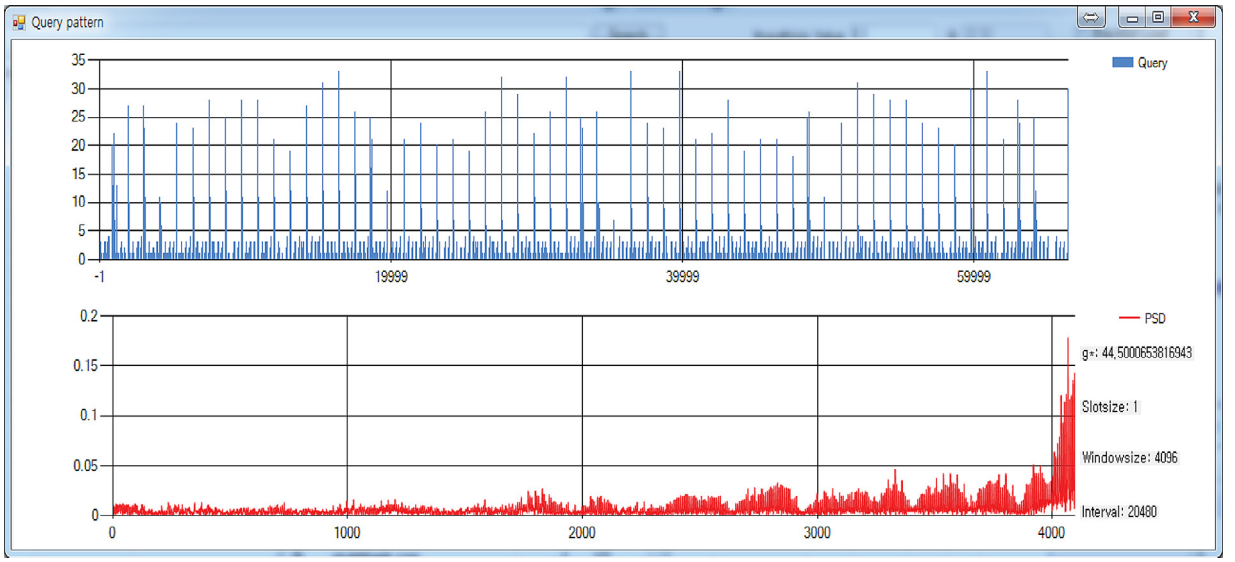


Fig. 5. Example of DNS queries and the corresponding PSD graph for real malware samples *BIN_Ramnit* (images were captured with the PsyBoG prototype).

5.1.3. Time series

In this step, we sample the DNS trace as though it were a binary signal by assigning it the value of 1 at each query request and 0 at intervals between query requests, and the sampling interval is 1 s. According to the recommendation for accurate and fast signal processing operation, a power of two, precisely 256, 512, and 1024, until 16,384, is regarded as the length of a time series. Although we observed the introduction of large gaps between queries, a large gap does not significantly compromise the PSD results [24].

5.2. Detection accuracy

Before evaluating the accuracy of PsyBoG, first we had to prove our assumption that recent botnets generate periodic DNS queries. To this end, we utilized real malware traffic (refer to Table 1) to observe the pattern of DNS queries generated by the malware. Fig. 5 illustrates an example showing the representative periodic pattern of malware *BIN_Ramnit*. The queries are generated within almost every second (upper graph). Even though the queries fluctuate, the fluctuation also represents a certain periodic pattern. The PSD result peaks at near 0.5 MHz with $g^* = 44.50$ (lower graph). *BIN_Ramnit* generated 60,116 queries for 382 domains which consisted of **unreadable** strings. It seems that the abnormal domains are generated via DGA to avoid blocking of its communication channel. Recall that, DGA is a DNS technique used by botnets that dynamically generates many "quasi-random" domain strings for C&C communication purposes. Examples of the abnormal domain strings are listed in Table 3.

The malware traces prove that our assumption is correct. Now, as mentioned in Section 4.4, we need to set an optimal threshold z for g^*. z depends on the expected false positive rate Pr_{FPR} which can be set by users, but the detection accuracy will be affected accordingly in a tradeoff relationship. In order to get appropriate Pr_{FPR} for an optimal threshold z, we performed a simple test of detection accuracy for the given malware traces with various Pr_{FPR}. Fig. 7 indicates that the detection accuracy seems stable as long as $Pr_{FPR} \geq 0.1\%$, but dramatically drops when z is set with $Pr_{FPR} < 0.1\%$. From this observation, we conclude that the proper $threshold_{SPT}$ is $z_{0.1\%}$.

Table 3
Examples of queried domain names by *BIN_Ramnit*.

Domain name	Domain name
ufxsqnjtryrny.com	stleikxkbjwo.com
stleikxkbjwo.com	lrqxvrqsihwtudox.com
eeuprbpohspwje.com	tlxfrilp.com
:	:
bvkdfvxoqxsabk.com	atfkpyicxsrrwqbct.com
wqnefkerofcmrap.com	jjfcilvuchkjvutlho.com
ginbkjuweobmwp.com	mmwhewlrckie.com
:	:
kbadlfpgtec.com	kjuldacvvmdffxi.com
ymcwineqkj.com	jyvfsnsqddbgxq.com

Table 4 exhibits the detection results. Nineteen of the malware traces in the test set were detected based on their abnormal periodicity, whereas only *BIN_dirtjumper* remained undetected in our experiment. To understand the reason for the miss-detection, we examined the detail of the trace. Fig. 6 illustrates the DNS trace and corresponding PSD graph. As we can see, it reveals that *BIN_dirtjumper* did not generate periodic queries during the monitoring time; therefore, the PSD score was not significant. The highest PSD value was 11.36, where $s = 1$, $n = 64$ and $t = 832$. Despite the number of queries (1004 queries within 941 s), concentrating the generation of queries in certain time periods complicates discovery by periodicity estimation. However, a partial periodic pattern of DNS queries emerged from around 650 to 940 s where the interval is approximately 70 s; thus, we strongly believe that there is a rare chance to reveal the periodicity of the malware as

Table 4
Detection results for the 20 real malware samples of which 19 were successfully detected.

Malware name	Detected	g^*	Segment size s	Number of segment N	Detected time t
BIN_Ramnitpcap..	O	44.50	1	4096	20,480
Sality..	O	30.70	4	1024	1024
BIN_Kuluoz-Asprox..	O	39.45	1	1024	1024
purplehaze	O	33.16	1	2048	2048
BIN_Cutwail-Pushdo(2)..	O	25.78	1	256	1046
BIN_Wordpress_Mutopy_Symmi..	O	22.05	1	32	96
BIN_ZeroAccess_Sirefef..	X	–	–	–	–
BIN_dirtjumper..	O	19.45	8	8	128
BIN_CitadelPacked..	O	36.25	6	16,384	540,672
cryptolocker..	O	18.62	1	64	128
Sweet-Orange-EK..	O	25.09	1	128	128
Angler-EK-uses-java..	O	27.41	1	64	64
Angler-EK-uses-silverlight..	O	16.27	1	51	51
Magnitude-EK..	O	42.93	1	128	128
phishing-malware..	O	27.38	1	64	64
Sweet-Orange-EK..(2)	O	41.58	1	64	104
Angler-EK-uses-flash..	O	18.71	1	16	16
Sweet-Orange-EK..(3)	O	25.32	1	64	64
phishing-malware..(2)	O	49.34	1	128	128
phishing-malware..(3)	O	44.87	1	64	474

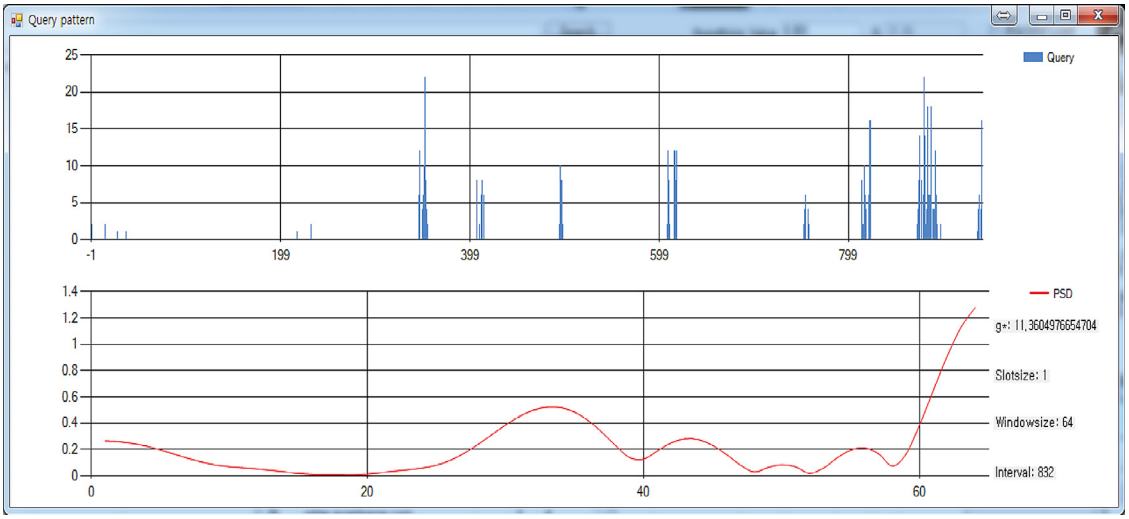


Fig. 6. DNS query pattern and corresponding PSD values of undetected malware sample (*BIN_dirtjumper*).

long as the subsequent trace is available. Nevertheless, we decided to consider the malware trace as a false negative; consequently, the detection accuracy of PsyBoG for the test data is 95% (19/20).

Now, we discuss the parameters that are used in the PSD estimation step. Recall that, in PsyBoG, we apply two different parameters for a single PSD calculation round, namely the size of a segment s and the number of segments n. For example, PSD can utilize 4,096 input signals with $s = 1$ and $n = 4096$, or $s = 2$ and $n = 2048$. The parametric divergence was intended to cover various types of periodic communication patterns emerging from the activities of different types of botnets. Even though we intended performing multiple PSD calculations according to the parameters, our computational resources may be insufficient in some environments. Therefore, suggesting a certain parameter setting which shows the best performance for general botnets was considered helpful. To this end, we evaluated the detection performance with different parameter combinations, where $s = \{1, 2, 3, \ldots, 9, 10\}$ and $n = \{8, 16, 32, \ldots, 8192, 16384\}$. Fig. 8 shows the detection results.

The best detection performance was obtained when s was 1 s and n ranged from 256 to 4096. More precisely, s recorded the best result with 1 s, and the detection performance was continuously lowered as long as s was increasing. It seemed that the shortest segment size provided a better reflection of the traffic pattern, whereas a longer segment size might dispel the characteristics of the pattern in some sense. Similar results were obtained for n ranged from 256 to 16,384. This is because n directly indicates the observation time, which has to be sufficiently long to determine characteristic behavior. Even though the best performance was obtained for $n = 256$, this result only covers 15 of the 20 malware samples. Our in-depth

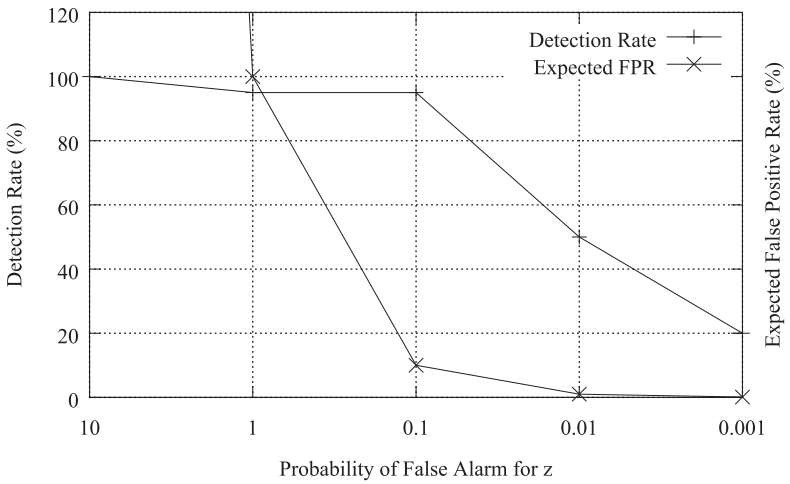


Fig. 7. Detection performance along with the PSD parameters.

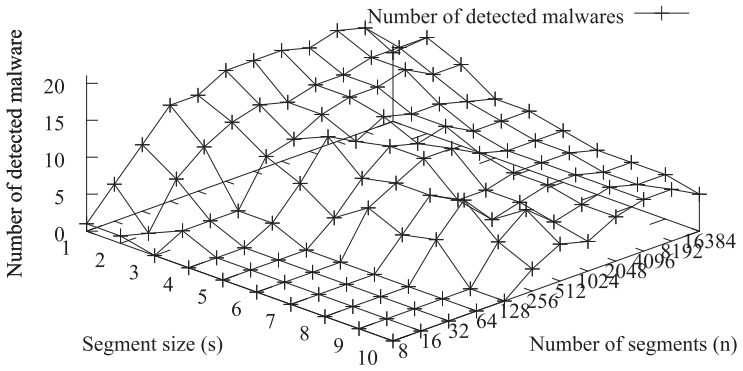


Fig. 8. Detection performance along with the PSD parameters.

analysis revealed that the different values of n could cover different types of malware.

Based on the experimental results, we regarded the size of a segment s as 1 and the number of segments n as ranging from 256 to 16,384 in further experiments.

5.3. Signal-to-noise test

The robustness of PsyBoG against the normal user traffic was verified by conducting signal-to-noise ratio (SNR) tests with the malware traces. The SNR has been researched in previous studies, and it has already been proven that the noise signals cannot influence other spectrum components in the frequency domain. Nevertheless, we conducted an experiment on similarity analysis with SNR to investigate the extent to which PsyBoG is robust to normal user traffic. The reason is that there is no guarantee that the similarity of the significant ordinates of the periodogram between two signals would not be affected by noise, even though the noise has no effect on the ordinates in which the periodicity is located. To this end, we utilized the real malware traces, to which we randomly added noise traffic by increasing the SNR up to 95% of the total traffic volume.

Fig. 9 shows the variation in the similarity results between two signals according to the SNR. We performed the experiment by using both PSD and pDist, and illustrate five representative results in the figure. The Y and X axes represent the similarity ratio and the SNR respectively. When the noise traffic accounted for 75% of the total traffic, pDist between the original traffic and the mixed traffic archived approximately 80% of similarity on average. At SNR values exceeding 75%, pDist continued to show reasonable similarity results with 60% being the worst case. In the best case in this experiment, *BIN_Kuluoz-Asprox* scored over 96% of similarity across all the SNRs. Considering the fact that bot programs typically generate a relatively larger number of queries compared to normal users, the SNR test shows that PsyBoG is very robust against the noise traffic and that it has the ability to discover the periodic behavior and group activities of botnets regardless of user behavior, even if there are many users queried in the DNS traces.

5.4. PsyBoG performance with large-scale data

This section describes the estimation of PsyBoG performance with large-scale DNS traces. The scalability and overheads are also discussed in detail.

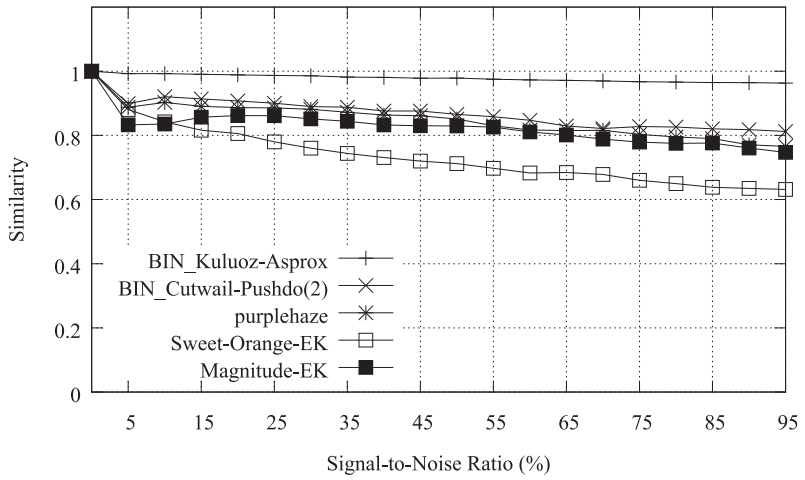
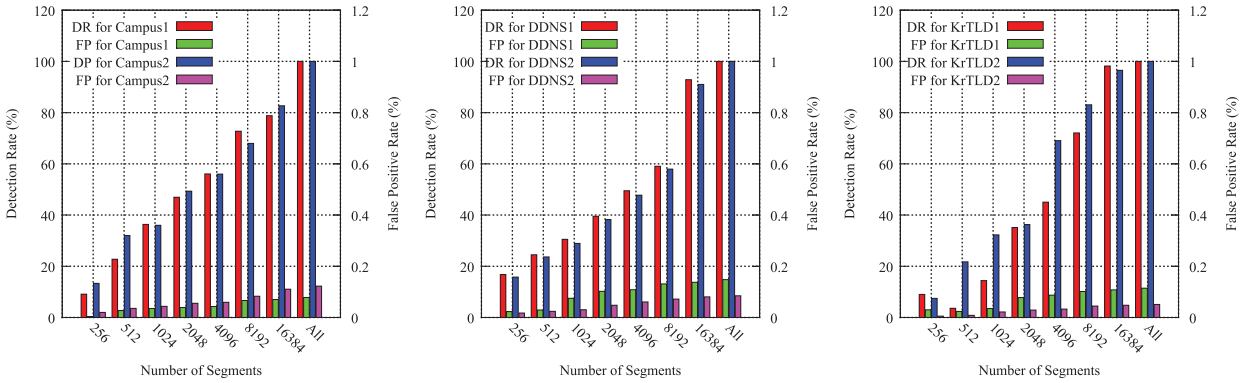


Fig. 9. SNR tests for the malware traces.



(a) RDNS (b) ADNS (c) TLD

Fig. 10. Detection accuracy as the number of segments of time series.

5.4.1. Detection performance in real DNS

In the experiments described in the previous section, we showed that PsyBoG is accurate and robust. Here, we describe the results of allowing PsyBoG to operate on the large-scale DNS traffic captured from real-world DNS servers (refer to Table 2).

As mentioned before, the PSD input data is based on time series of different lengths. This difference impacts the detection accuracy; thus, we investigated the relations between the number of segments and the detection accuracy to determine the efficiency for different time series. Fig. 10 exhibits the detection results for the different time series. The results show that increasing the number of segments improves the performance, while maintaining the false positive rate around 0.1%.

Interestingly, even though the small number of segments (*i.e.*, 256, 512, and 1024) shows a relatively low performance, the detection results are still valuable, because some bot hosts were discovered by only using input settings with a small number of segments. According to our analysis, this result was caused by the sporadic behavior of the bot hosts. For example, a bot host would only operate for a while until the host owner turns off the system. In such a case, even if the bot host were to show periodic behavior, it would only be reflected during a short period of the time series. Therefore, for a large time series, the periodic behavior would not be able to affect the peak value significantly.

The sporadic behavior of a botnet presents an actual problem for many botnet countermeasures in terms of practicality. Advanced botnets automatically show generated domain names with time seeds for the synchronization of each bot host irrespective of the condition of hosts, such as being turned off. The existence of a long term consisting of two algorithmically generated domain queries resulting from the machines being powered off complicates discovery of the botnets. This is the reason some research leveraging the characteristics of continuously generated domain names are unable to determine the relationship between the two domain names under these circumstances. Another type of botnet mitigation, which monitors synchronized botnet behavior, also has difficulties with such sporadic behavior.

By contrast, PsyBoG operation is based on the *divide and conquer* strategy. We separate the input data into several time series in accordance with n, and perform the PSD

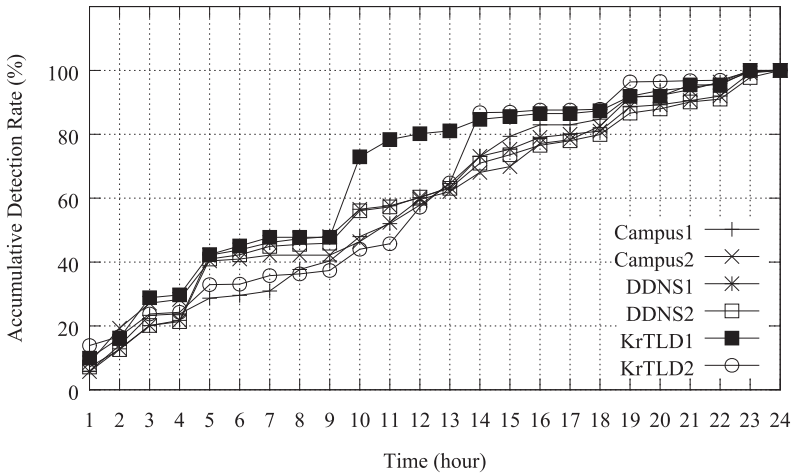


Fig. 11. Accumulative detection rate.

analysis for each input signal by using the sliding window approach. Finally, we extract a representative periodicity for each host from the entire monitoring data. Therefore, PsyBoG is available to accurately analyze the behavior of each host even if some of the bot hosts only operate during a short period of time. This strategy improves the robustness of PsyBoG against sporadic botnet patterns.

Fig. 11 exhibits the accumulative detection rates of PsyBoG on the 24-h traces. As we can see, the detection rate continuously increases as a function of time. Before we accumulated the detected hosts, we expected an increasing trend to occur on a log-scale, as shown in the accumulative number of total hosts (please refer to Fig. 14(b)). Interestingly, the increasing trend observed for the detection rate is linear. According to our analysis, the increments are caused by the sporadic behavior of the bot hosts. Approximately 50% of bot hosts perform periodic query generation at a certain time after the first DNS query generation. Furthermore, some bot hosts only show periodic query generation during a short time period, when the host machines continuously generate DNS queries. From the analysis, we can say that, in the real world, the sporadic behavior of botnets is an actual problem, although PsyBoG is practically robust against this problem.

5.4.2. Botnet group detection

The group detection results for the real-world DNS traces are presented in the appendices. Note that we utilized a DNS blacklist [6,43–46] to classify the detection results into the following categories: known malicious, unknown malicious, and false positives. If a domain name is involved in periodic DNS queries and the domain appeared on the blacklist, we determined the host group querying the domain to be known malicious. If a domain name was queried periodically, but was not listed on the blacklist, then we performed a manual investigation through Google search to determine whether the host group belonged to the unknown malicious or false positive categories.

As a result, PsyBoG discovered six known botnets, 19 unknown botnets, four adwares and three suspicious groups from *Campus1* and *Campus2* (Appendix A). The suspicious groups showed relatively high periodicity and abnormal domain queries. Especially, one group generated queries against more than 3000 domain names which are variants of 15 original domains. The domain variations showed very similar patterns with the DGA domains. The only reason why we classified the groups as suspicious was that there was insufficient evidence to prove that these domains were malicious.

With *DDNS1* and *DDNS2*, 20 known botnets and three unknown botnets were successfully detected (Appendix B). The detection results include well-known botnets such as Maliposa, Palevo, and Geinimi. Our in-depth inspection revealed the individual botnet groups to show distinct periodic query patterns even though they belong to the same botnet families. This is because of the prevalence of the bot codes at that time; thus, many attackers utilized these bot codes with different operational settings. Among the three unknown botnets, Nitrol, which utilized the domain names 3322.org, showed an identical infection rate. The domain 3322.org, which is served by a Chinese DDNS provider, was notorious for malware hosting until 2012, but in Sep. 2012, Microsoft took down the domains using DNS sinkholing. Even though these domains are disappeared, we show that PsyBoG has the ability to discover malicious DDNS usage.

We discovered another seven suspicious botnets and one unknown botnet from *KrTLD1* and *KrTLD2*. Unlike the other results, all the botnets had ***unreadable*** strings in the third domain name. It seems that the detected botnets utilized DGA approaches to evade current countermeasures. Although the domains are obviously malicious, we were unable to obtain any information from the Internet, security report, or any blacklists about these domains. The lack of information about these DGA domains appears to be caused by the cost efficiency problem. As we mentioned, DGA generates a huge amount of random domains for temporary use. From the point of view of security responders, counteracting every single DGA domain results in a huge cost requirement. Considering the problem of practicality,

Table 5
False positives.

Type	IP#
Torrent trackers *i.e.,* tracker.gaytorrent.ru	17
A scientific research crawler	3
Mail servers	1
Network Time Protocol (NTP) queries	8
Web hard service	161
Web streaming service	49
Etc.	4

again, PsyBoG was shown to be an effective and efficient countermeasure against DGA botnets.

Table 5 shows the false positives detected in our experiments. These false positives were caused by legitimate services such as Torrent trackers, mail delivering services, and NTP. One false positive group was identified as a scientific research crawler, which crawled information from thousands of Web servers. Despite the low false positive rate obtained by PsyBoG (0.1%), we expect it to be possible to continue reducing the false positive rate by listing the legitimate domains in the whitelist.

5.5. Scalability analysis

In this section, we analyze the scalability of PsyBoG. The definition of scalability is a characteristic of a system that describes its capability to perform under an increase or expanding workload. In a point of view of PsyBoG that performs an analysis using network information, *i.e.* DNS traffic, the next three features including the number of queries (feature 1), domains (feature 2), and IP addresses (feature 3), which affect the PsyBoG's workload, can be considered. In order to estimate the scalability of PsyBoG, we therefore investigate the performance of each component in PsyBog along with the increase of workload based on the features.

Early researches analyzing network packets or flows have suffered from the expansion of network usage, because the number of packets and flows to be analyzed is also dramatically increased. To solve the problem, many methods have focused on the probability model of sampling data [47–49]. The sampling has drawbacks in which the sampling may bring incorrect results or delay the aggregation of meaningful data. Adopting signal processing techniques is able to provide a great advantage at this point. Since signal processing transforms a data from the time domain to the frequency domain, the increase of the number of packets is not an issue any more. Similarly, it is applicable to solve the problem of the number of queries, and therefore the workload of PsyBoG is not affected by the feature 1.

Since PsyBoG analyzes the traffic generation pattern for each host irrespective of domains, the feature 2 cannot affect PsyBoG. Therefore, we only need to see how the increase of the number of IP addresses affects the PsyBoG workload. Figs. 12 and 13 illustrate the relationship between the number of queries m and the number of IP addresses k. As we can see, k shows a relatively small incremental trend while m increases, which makes perfect sense. The maximum number of k in a certain network is obviously limited, *i.e.*, the IP range for the /16 campus network is only 65,536. Therefore, the incremental ratio of the number of IP addresses decreases while m increases, and finally, k will no longer increase when it reaches its maximum value $max(k)$. Accordingly, a relation function f between m and k can be evaluated directly as follows.

$$k = f(m) = \log m$$

The workload of PsyBoG increases in a logarithm scale while the size of given DNS traffic is linearly increased. Consequently, we can conclude that PsyBoG is scalable.

5.6. Overhead estimation

The practicality of PsyBoG was evaluated by estimating the computation and memory overhead for each of its core methods, PSD, SPT, and pDist. We first consider the computation overhead as follows:

- **PSD computation overhead**: The computation time for PSD is affected by the length of the input signal (segments) n and the number of execution rounds $r = \frac{L}{n}$,

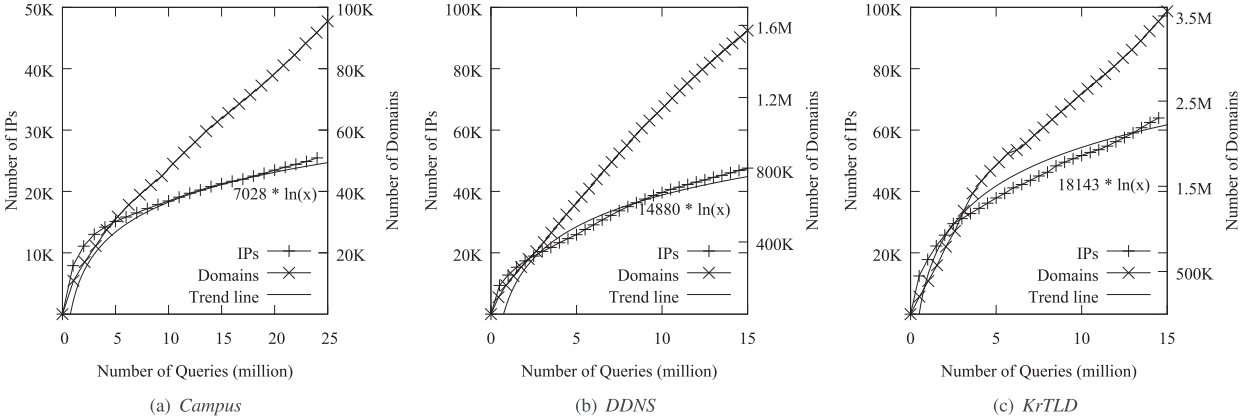


Fig. 12. Comparison between the number of IP addresses and the number of domains as a function of the number of queries for scalability estimation.

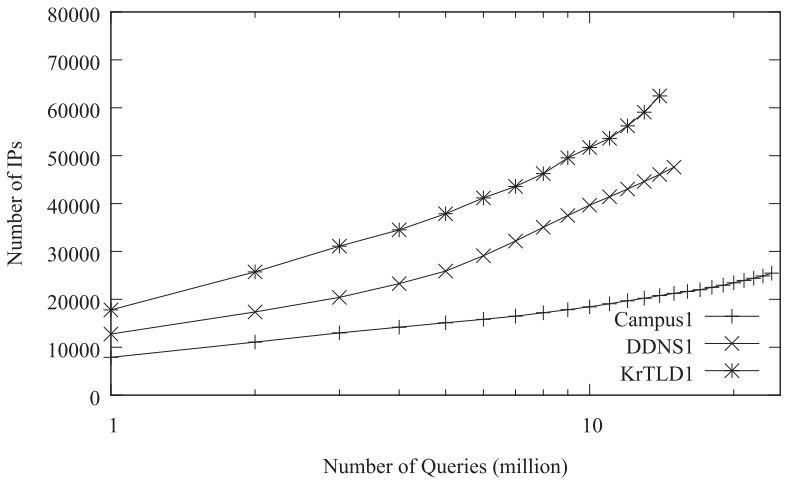


Fig. 13. Number of IP addresses accounted in a linear increment with a log-scale *x* axis. This provides proof of the log-scale increment of the number of IP addresses.

Table 6
PSD computation time for a 24 h trace (per IP address) and the pDist calculation time for a single IP address.

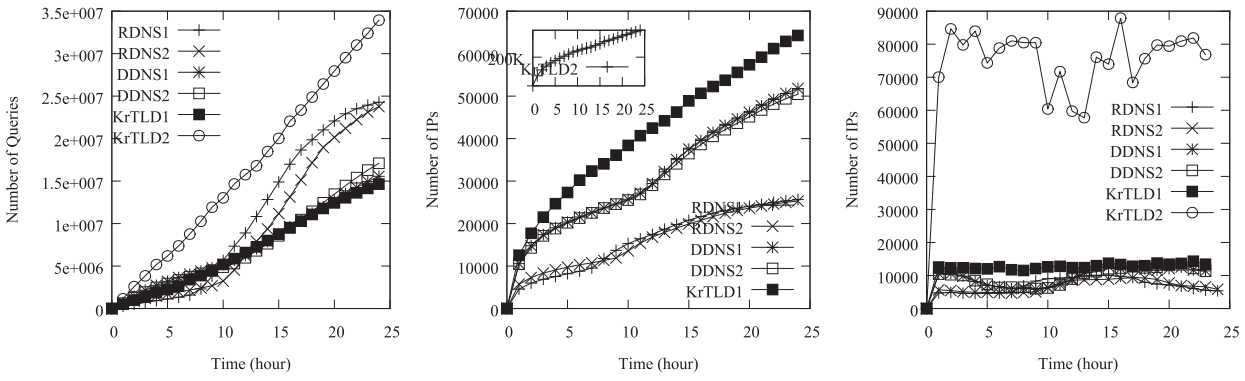
Number of segment n	Time for PSD (ms)	Time for pDist (μ s)
256	9.62	3.6
512	10.30	7.0
1024	10.87	16.0
2048	11.64	31.1
4096	12.73	63.4
8192	12.58	117.8
16,384	13.75	243.0

where L is the total time length of input data. In our experiments, the maximum value of n is limited to 16,384; therefore, the PSD computation time for a single round is reduced to the expected time. Moreover, the number of PSD execution rounds r is decreasing as long as n is increasing. Consequently, the computation overhead for PSD depends only on the total length of the input data L and the number of hosts k.

Table 6 shows the average PSD execution time per IP address for a 24-h trace. As we can see, the PSD computations required only 9.62–13.75 ms, which is a small variation in spite of the different lengths of the input signal n. Considering the different length of n, the PSD computations required only 81.52 ms for a single host. Based on the execution time, the PSD calculation for the B-class network (65,536 IP range) will require 5342 s (less than 2 h). In fact, it required only approximately 2 h to analyze a 24-h trace for approximately 387 K hosts (*KrTLD2*) using a multi-threading scheme with a quad-core processor. We believe that this PSD computation overhead is relatively small.

- **SPT computation overhead**: SPT only requires a small portion of computation overhead. After the PSD analysis, SPT performs $\frac{n}{2} * r$ time comparisons for a single IP address. In our experiments, SPT only required an average of 8.58 μ s for a 24-h DNS trace. Considering the different lengths of n, SPT only requires 0.6 ms. For the B-class network, SPT will require less than 40 s. As mentioned above, we used multi-threading, and as a result, SPT required less than 1 min for *KrTLD2*. This is an almost negligible computation overhead.
- **pDist computation overhead**: pDist is performed for each pair of hosts, which requires $k * (k - 1)$ times computation. Even though the pDist operation for a single round only requires several microseconds, the total computation overhead might be huge as a result of the k^2 computation rounds. Nevertheless, the number of IP addresses k could be limited in accordance with the IP range of a certain monitoring network as we mentioned before. Fig. 14 exhibits the statistics of our test data. As we can see, the accumulative number of IP addresses represents the log-scale increment (Fig. 14(b)), while the total number of queries is constantly increasing (Fig. 14(a)). Furthermore, the number of IP addresses detected during every hour does not show significant change (Fig. 14(c)). From this observation, we can say that the number of IP addresses k for a certain moment is grounded in a reasonable and expectable number.

  Table 6 presents the average computation time for pDist. pDist records different computation times depending on the length of the input signal n, but the maximum value of the computation time is only 243 μ s. Recall that pDist leverages only the hosts discovered by the SPT analysis as they display significant periodic communication attempts. Accordingly, we expect the number of hosts k for the pDist operation to be considerably reduced. In our experiments, only 223, 116, 711, 713, 111, and 750 IP addresses are actually fine-grained from a total of 25K, 23K, 48K, 46K, 63K, and 378K IP addresses, respectively. Under the assumption that there are 1,000 revealed IP addresses and the hosts are all discovered by $n = 16{,}384$, pDist will require 243 s ($1000 \times 1000 \times 0.000243$ s), which is also a relatively short computation time.



(a) Total number of generated queries (b) Accumulative number of IP addresses (c) Number of IP addresses per hour

Fig. 14. The statistics of the DNS data. The number of IP addresses is converged while the total number of queries is continuously increased.

Now, we estimate the memory overhead for PsyBoG.

- **PSD memory overhead**: PsyBoG applies the sliding window strategy for an input data length L. According to the strategy, PSD generates the corresponding periodograms for every round r, where $r = \frac{L}{n}$. The periodogram ordinates have a length $\frac{n}{2}$ and each ordinate can be represented with 4 bytes (type double) to store the power of each frequency $\frac{n}{2}$.

$$\left(r * \frac{n}{2}\right) * 4 = \left(\frac{L}{n} * \frac{n}{2}\right) * 4 = \frac{L}{2} * 4 \text{ bytes}$$

In addition, PsyBoG utilizes different lengths of n, $e = |n|$. Because the periodograms are temporarily stored in memory before SPT is performed, we can measure the PSD memory overhead for a single IP address as follows.

$$e * \frac{L}{2} * 4 \text{ bytes}$$

In our experiments, $e = 7$ and $L = 86{,}400$ s; thus, the total PSD memory overhead for each IP address was 1.2 MB. Note that, the PSD computations for individual IP addresses are not influenced by each other, which means that only four PSD computation were operating simultaneously in our experiments due to the multi-threading scheme running on a quad-core processor. As a result, 4.8 MB of memory overhead was incurred.

- **SPT memory overhead**: In terms of SPT memory overhead, we only have to consider the memory spaces for a single periodogram with the highest significant peak value. From the periodograms $P_{xx}[K_{n/2}]$ which are the results of the PSD computation for a single host, SPT analyzes g_x^* for $P_{xx}[K_{n/2}]$ and selects a representative periodogram $P_{ii}[K_{n/2}]$ with the highest significant peak value g_i^*. Therefore, the SPT memory overhead for a single IP address is,

$$\frac{n}{2} * 4 \text{ bytes}$$

A comparison of each of the SPT results in the pDist operation requires us to maintain the representative periodograms for k hosts in memory.

$$k * \frac{n}{2} * 4 \text{ bytes}$$

In the worst case achieved with our experiments, where $n = 16{,}384$ and $k = 387$K (*KrTLD2*), SPT requires approximately 12 GB of memory space, which is considerable. Nevertheless, considering the fact that there was only a relatively small portion of IP addresses ($k_{SPT} = 750$), where $k_{SPT} = |IP|$ and $IP = \{ip | SPT(ip_i) \geq z, ip_i \in IP_k\}$, the SPT memory overhead was reduced to 24 MB, which is almost negligible.

- **pDist memory overhead**: The result of a single pDist computation occurs in the range between 0 and 1; thus, pDist requires 4 bytes of memory space (double) to store the result for a single computation. Because there are k_{SPT} IP addresses that have to be compared, the memory overhead for pDist would be,

$$(k_{SPT})^2 * 4 \text{ bytes}$$

As we mentioned above, the number of IP addresses k_{SPT} is a small number. The pDist memory overhead is also accounted in a very practical manner.

Through the overhead estimation, we show that the core methods of PsyBoG, namely, PSD, SPT, and pDist, are very efficient and scalable even for large-scale DNS traces. We believe that the overheads for PsyBoG operations are tolerable in terms of practical usage.

6. Discussion

In this section, we discuss potential techniques that may exploit PsyBoG, and discuss the extent to which PsyBoG is resilient against those techniques. Ways to improve the efficiency of PsyBoG and further research topics are also addressed.

6.1. Random query pattern

PsyBoG leverages the periodic query patterns of bot programs in accordance with the assumption that the bot program simply follows the source code that is written to ensure automatic and constant bot operations. We already used experiments to show that this assumption holds for real-world botnets. Nevertheless, bot writers might attempt to evade PsyBoG by modifying their query patterns.

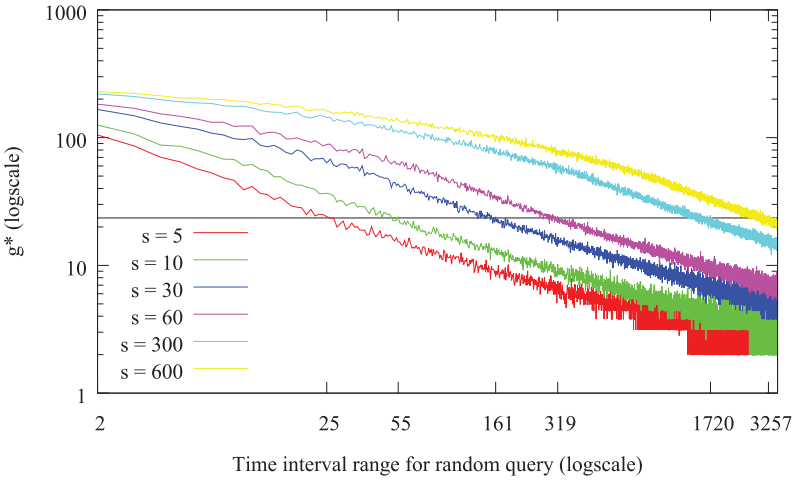


Fig. 15. Periodicity measurements against randomly generated query patterns.

A reasonable potential alternative would be to apply a random function to generate non-periodic queries. More precisely, bot authors could randomize the time interval between DNS queries by applying a random function *rand*(y), where the time interval ranges from 0 to y. A randomized time interval i located within this range, could exploit the periodicity of query patterns.

To examine this, we analyzed the periodicity measurements against artificially randomized query patterns. First, we generated random query traffic with different maximum random ranges from 2 to 3600, which means that, *e.g.*, a random query traffic $x[n]^{rand(600)}$ has a query sequence such as x_{k-1}, x_k, and x_{k+1}, and the time intervals between the queries can be random numbers between 0 and 600. Second, the input time series was built by varying the size of the segment s, *e.g.,* 5, 10, 30, 60, 300, and 600 s. According to s, every query generated on time from t to $t+s$, is accumulated into a single segment of time series T_j. Finally, the PSD of time series T was analyzed. Fig. 15 depicts the significant peak testing with the randomized query patterns.

According to the simple tests, we confirmed that the randomization of query pattern is able to disturb the periodic pattern extraction. Even though some random query traffic like $x[n]^{rand(25)}$ could be detected by using $s \geq 5$, growing s would bring serious false positives. In order to detect the randomized query patterns, it is necessary to leverage more information such as queried domain names, which are not affected from the randomization. This problem will be investigated as part of our future work.

6.2. Slow query pattern

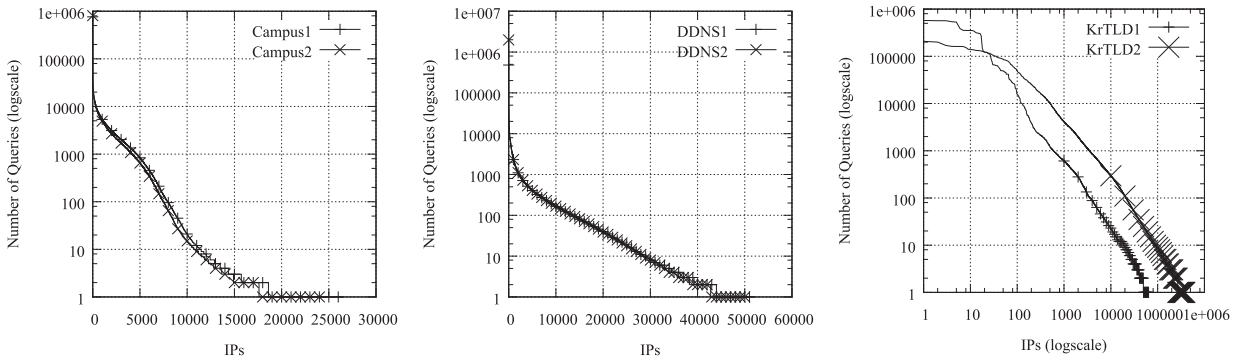
Bot authors might apply slow query patterns to hide their communications. For example, some bots could generate a DNS query once every hour, day, or week to communicate with the botmaster. However, the application of slow query patterns causes the function of botnet to deteriorate considerably. The only reason botnets continue to prevail is their ability to concentrate attack resources for massive attacks; therefore, the power of a botnet completely depends on its availability and capability. Slow query patterns ruin the availability and capability, because slow queries limit the response to the master's commands. Generating slow queries also limits the agility of a botnet, which may be the cause of a *single-point failure*. Despite the limitations, a botmaster could apply slow query generation. However, theoretically, PSD analysis is not affected by large gaps of two queries, as long as the given time series is long enough to carry the periodic signal.

6.3. Bot hosts behind NAT boxes

Bot hosts can be behind a NAT box. In such a case, PsyBoG is only able to recognize the IP address of the NAT, because it is limited to harvest an enough information of the hosts from DNS traffic. However, it does not mean that PsyBoG fails to extract the periodic queries coming from behind of the NAT. No matter how many hosts are behind a NAT, if a periodic query signal q_i from one of the hosts is aggregated at the NAT, $Q \subset \{q_1, \ldots, q_k\}$, the periodicity of q_i is reflected in a certain frequency of aggregated signal Q. Therefore, detecting periodic behavior of bot hosts behind a NAT is not our concern.

6.4. More efficient PsyBoG

The efficiency of PsyBoG could be further improved by considering another approach, which involves the coarse-graining of the IP addresses in accordance with the number of queries per host. In the real world, the network usage per host follows a power law, and the DNS usage also follows a similar pattern. Furthermore, considering the fact that recent botnets generate a large number of domain queries to evade botnet countermeasures, there is a great possibility that bot hosts could be ranged in IP addresses located in the upper region of the power law. Driven by this phenomenon, we could improve the



(a) RDNS (b) ADNS (c) TLD

Fig. 16. Statistics of the number of queried domains per IP address.

efficiency of PsyBoG by narrowing down the number of hosts to be investigated.

Fig. 16 depicts the number of queries generated by each host for 24 h. Among the hosts appearing in our data set, only approximately 20% of the hosts generate over 300 queries, and none of the other 80% of the hosts was identified as an abnormal host. The number 300 should obviously not be regarded as a precise number for the threshold for every DNS server, but we strongly believe that there is an appropriate threshold for DNS servers in accordance with their circumstances *i.e.,* considering the number of clients, regions, and monitoring times. Because the PsyBoG operations rely on the number of hosts to be analyzed, reducing this number should enhance the speed and efficiency of the PsyBoG operations. If we require a more efficient way to apply PsyBoG in the view of the limited system resources in practical usage, reducing the number of hosts before the PSD analysis according to their query amounts could be a reasonable choice.

6.5. Malicious domain extraction

In the modern Internet, DNS is very important from a security perspective. Considering the way in which modern botnets utilize DNS in their life cycle and the fact that DNS is the gateway for current Internet usage, DNS forms the topic of many security research efforts and attracts the attention of many network operators. More precisely, a domain name is the key element of botnet mitigation. Once a domain name is revealed as a malicious domain used by a botnet, the responders are able to seize the botnet by simply updating the zone files to deny access to the corresponding domain information. Driven by this reason, most of the botnet countermeasures have focused on the discovery of malicious domain names.

Unfortunately, the number of domain names is growing extremely fast; thus, the scalability problem has a negative impact on previous countermeasures. The rapid increase in Internet usage as a consequence of the growth in the number of Internet users and devices has resulted in an increase in the total volume of DNS traffic along with the Internet traffic. Especially, the increasing number of domain names simply follows the increase in DNS traffic. Hence, extracting malicious domains from among the millions of domains is a great challenge in terms of practicality.

PsyBoG operations do not rely on the domain names. Rather than extracting malicious domain names, we have focused on IP address discovery, because the number of IP addresses is limited in the modern Internet. Actually, discovering the malicious host constitutes another way to mitigate botnet threats. With IP address discovery, PsyBoG effectively mitigates the damage caused by botnets. Nevertheless, there is no doubt that denying access to the malicious domains is the most efficient approach. Therefore, the question remains as to which other approaches could be considered to achieve the malicious domain extraction with PsyBoG.

The most reasonable approach would be to extract those domains with the most significant effect on the periodicity. For example, if thousands of distinct domain names have been queried and only a small number of malicious domains are queried periodically, by performing signal processing, only the malicious domains would be affected by significant peak values. Therefore, the malicious domains can be extracted if we can calculate the rate at which each domain influences the significant peak value. This work will form part of further research.

7. Related work

Since botnet becomes one of the most significant Internet threats, numerous research efforts have been devoted to mitigate botnet threats. In this section, we review various botnet detection mechanisms that can be classified according to their approaches, detection object, and source data.

In the early period of the age of botnet, many botnet detection approaches have been suggested and host-based detection is the one of them. Host-based detection approaches mainly aim to investigate abnormal activities in a computer by analyzing the internal components of the computer system [5,50]. Main advantage in using host-based detection approaches is that they allow normal users to easily and actively detect abnormal activities in their system.

BotSwat [9] is a tool to monitor the execution of Win32 binary by intercepting the system calls. BotSwat traces all input data using a taint propagation trace technique to discover botnet commands, regardless of botnet's communication protocol and architecture. Unfortunately, BotSwat showed unignorable false alarms and high system overheads caused by the taint propagation scheme.

Liu et al. [51] models botnet behavior into three phases; automatic startup, C&C channel establishment, and launching attack. In their work, a tool called BotTracer was implemented to analyze these phases with the assistance of virtual machine. BotTracer was built on an assumption in which a bot code is unable to recognize the virtual machine, but, in real-world, there are many techniques which can detect the virtual machine. Furthermore, the three phases of behavior can be monitored in normal program; thus, it brings many false detection results in practice.

Jacob et al. present JACKSTRAWS [52], which leverages system call information extracted from bot code execution. By monitoring system call graphs observed from the code execution, they distinguish unknown C&C communication traffic and unrelated traffic with machine learning techniques. However, if traffic that is not related to C&C communication is included in the learning process, the detection results would be compromised. Moreover, recent botnets may not strictly follow the system call graphs.

Despite the devoted research efforts, host-based detection approaches have a fundamental weakness. Performing individual computer system analysis is a cost inefficient approach. Because of security awareness of normal users, deployment of detection mechanism is also an unsolved problem [53].

There have been efforts on network-based detection such as Bothunter, BotSniffer, BotMiner, and BotGrep. BotHunter [54] models a botnet infection model and deals with IDS-driven dialog correlation. BotSniffer [55] focuses on a highly synchronized communication of botnets, and BotMiner [13] applies clustering algorithms to perform cross-plane correlation. BotGrep [56] analyzes C&C communication on the overlay topologies to defeat P2P botnets. BotCop is a botnet traffic detection system which classifies the network traffic into different application communities using packet payload signatures and a decision tree model. In spite of these outstanding research results, network-based botnet detection is still suffering from high false alarms and significant overhead due to the massive traffic volume.

Tegeler et al. proposed BotFinder [24], a system to detect infected hosts in a network using only high-level properties of bot traffic. BotFinder applies a clustering approach to model botnet behaviors, especially bot traffic patterns including time interval, duration, and FFT of communication. This work is the one which is partially similar to our work, but BotFinder still suffers from the user generated traffic and huge volume of network traffic.

As the network-based botnet detection mechanisms have experienced difficulties when processing huge amounts of network traffic, the DNS has been considered a source of malware detection [57,58]. Compared to other research, DNS-based approach gives researchers advantages to face with payload encryption and huge amounts of traffic. From the advantages, many research has focused on DNS traffic analysis [59].

BotGAD [15] distinguished group activities in DNS traffic from legitimate users activities by using the concept of the client set of a domain name. BotGAD defined a group activity as an inherent property of botnets. They measured the similarity of the DNS clients according to domain names using quantitative likelihood. Unfortunately BotGAD can be countered by sophisticated botnets which separate their bot hosts into several sub-groups using multiple domain names.

Choi et al. [60] and Yadav et al. [61,62] respond to the multi-domain botnet problem by grouping domain names based on the lexical similarity of domain names and network features, such as corresponding IP addresses. However, these approaches still have limitations when faced with the multi-domain malwares that do not use DGA and have little lexical similarity in their domain names.

Sharifnya et al. [63] introduced reputation-based DGA botnet detection approach. They identify DGA domain names by estimating the number of failed DNS queries. Their idea is that only small number of DGA domain names are resolved to be used as C&C servers; thus, there is a high possibility that the failed DNS queries indicate an infection of DGA botnet. However, their approach is not capable of mitigating other domain-fluxing techniques used by botnet, such as DDNS and fast flux.

Antonakakis et al. [64] utilized NX domain names for detecting DGA domain names. Their approach is efficient to detect newly generated DGA domain names. However, it could not respond to most of malicious DNS queries except DGA-based C&C queries, because the clustering method of Pleiades was dependent on lexical and structural features of domain names. Therefore, their detection coverage would be limited, because DNS activities of botnet are not limited on C&C communication.

Salomon et al. [65] proposed a Bayesian approach for botnet detection based on the similarity of their DNS traffic. The hypothesis of their approach is that bot hosts belonging to the same botnet have similar DNS traffic query pattern that can be distinguished from normal DNS traffic. However, their study results can be affected by background traffic, because there is a high possibility of a similarity of DNS traffic in well-known domain names.

As compared to the previous studies, PsyBoG is the most capable approach to detect sophisticated botnets leveraging highly advanced evasion techniques, by considering inherent properties of botnet *e.g.,* fully automated DNS query generation. Furthermore, PsyBoG has resilience against large-scale DNS traffic, therefore the increase in the number of network traffic is no longer a problem.

8. Conclusion

This paper presented an introduction to PsyBoG, a novel botnet detection approach that leverages large-scale DNS traffic. PsyBoG addresses practical problems associated with real-world DNS traffic analysis, including scalability, normal user traffic, and botnet grouping issues. Because the number of domains has been increasing rapidly, recent botnet countermeasures that rely on domain-based

analysis have experienced difficulties to deal with huge amounts of DNS traffic. By contrast, PsyBoG is based on IP-based analysis. Despite the rapid increase in the volume of DNS traffic, the total number of IP addresses is limited in accordance with the size of the monitoring network. Therefore, PsyBoG guarantees scalability for large-scale DNS traffic. Through the PSD analysis, we can successfully identify malicious behavior regardless of the existence of normal user traffic. The PSD analysis leverages the periodic patterns of query generation, such that advanced botnets, which utilize domain-fluxing techniques, such as DDNS, fast flux, and DGA, can be successfully discovered. Furthermore, PsyBoG performs botnet detection by grouping the periodic behavioral patterns of bot infected hosts. In our experiments, we evaluated the performance of PsyBoG with large-scale DNS traces collected from real-world malware samples and DNS servers. With the various real-world DNS traces, we also analyzed the effectiveness, efficiency and robustness of PsyBoG. The experimental results showed that PsyBoG is capable of delivering superior performance in terms of accuracy and practicality. In the future, we plan to extend the abilities of PsyBoG. To improve the resilience of PsyBoG against highly advanced botnet threats, we are considering fine-grained clustering methods as well as the extraction of malicious domains. We believe that PsyBoG will provide efficient and effective prevention of botnet operations in the current and future Internet.

Acknowledgments

This research was supported by the Public Welfare & Safety Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (2012M3A2A1051118).

Appendix A. Detection results for *Campus1* and *Campus2*

Known malicious group

Name	Domains	IP#
Palavo	ilo.brendz.pl ant.trenz.pl	2
Palavo(2)	peer.pickeklosarske.ru juice.losmibracala.org jebena.ananikolic.su teske.pornicarke.com	3
Palavo(3)	pica.banjalucke-ljepotice.ru sandra.prichaonica.com l33t.brand-clothes.net	2
W32/Tupym-D	h1.ripway.com	
Worm.Win32.AutoRun.fnc	www.balu0{xx}.0catch.com (14)[1] www.gearext.com	3
Trojan-pws.Win32.QQPass	@[a-z]{3,6}.@[a-z]{2,3}.ijinshan.com (4) @[a-z]{2,3}.@[a-z]{2,5}.duda.net (14) up.liebao.cn	6

Unknown malicious group

Name	Domains	IP#
N/A	@[a-z]{2}.cdn.qhimg.com (4) @[a-z]{3}.shouji.360tpcdn.com (3)	24
	loveting.no-ip.org	1
	sexman69.mlbfan.org	3
	sh.rstrainer.net sh.rstrainer.net.local	1
	biz5.sandai.net miserupdate.aliyun.com	9
	www.xiaopijia.com	2
	data2.168sm.com	1
	r.usyncapp.com t.usyncapp.com	12
	l33t.brand-clothes.net	3
	d2uzsrnmmf6tds.cloudfront.net	92
	@[a-z]{3,6}orbitdownloader.com (3)	2
	sc{xx}.rules.mailshell.net (6)	10
	js.moatads.com	45
	optimizedby.brealtime.com	35
	mydati.com	5
	c.ztstatic.com	5
	cm1.jssearch.net	109
	s@[a-z]{1}.kgridhub.net (6)	37
	hcimg.realclick.co.kr	51
Adware	servedby.bigfineads.com	2
	b.scorecardresearch.com	61
	servedby.bigfineads.com	2
	servedby.myinfotopia.com	4

[1]The domains have (n) variations with changing characters at the braces.

Suspicious group

Name	Domains	IP#
N/A	@[a-z][a-z0-9]{0,7}.www.0538hj.com (2644) @[a-z][a-z0-9]{0,7}.lieb.76yxw.com (40) @[a-z][a-z0-9]{0,7}.www.8885ok.com (82) @[a-z][a-z0-9]{0,7}.baidu.915hao.com (178) @[a-z][a-z0-9]{0,7}.www.jrj001.net (38) @[a-z][a-z0-9]{0,7}.www.kr5b.net (40) @[a-z][a-z0-9]{0,7}.wwee.iinfobook.net (154) @[a-z][a-z0-9]{0,7}.www.jeeweb.net (40) @[a-z]{8}.www.chuansf-1.com (19) @[a-z]{1}.wwww.boeeo.com (10) @[a-z]{1}.www.booooook.com (24) @[a-z]{1}.www.jiaduolu.net (15) @[a-z]{1}.jiaduolu.net (15) @[a-z]{1}.qianliri.com (13) @[a-z]{1}.hj19.com (24) @[a-z]{4,8}}.com (134) @[a-z]{8}.info (128) @[a-z]{8}.net (132)	1 3
	xayazkesh.in dubstepdrop.net halo4beta.co	6

Appendix B. Detection results for *DDNS1* and *DDNS2*

Known malicious group

Name	Domains	IP#
P2P -Worm.Win32	a0.twimg.com	263
	a2.twimg.com	
Kolabl worm	mails3.pes2009.biz	34
	mails.pes2009.biz	
TR/Dropper.Gen	mefound.com	13
Mal/FBScam-E	medialand.net	1
	rpcthai.com	
	e-junky.net	
Geinimi	clicksor.com	12
	trafbuy.ru	
bhek	webs.ono.com	20
	youvoid.info	
	www.ffil.uam.es	
Travnet	gamil.com	10
Mariposa	legion.sinip.es	4
Mariposa(2)	booster.estr.es	15
	shv4b.getmyip.com	
	shv4.no-ip.biz	
Mariposa(3)	bfisback.no-ip.org	23
	butterfly.sinip.es	
	qwertasdfg.sinip.es	
	youare.sexidude.com	
	mierda.notengodominio.com	
Mariposa(4)	legionarios.servecounterstrike.com	6
	legion.sinip.es	
Palevo	irc.zief.pl	13
Palevo(2)	tlaloc666.com	4
Palevo(3)	alotibi.panadool400.com	3
Palevo(4)	shv4b.getmyip.com	14
	shv4.no-ip.biz	
Palevo(5)	panchitox.laweb.es	1
	penchatox.sin-ip.es	
Palevo(6)	f5v9w.com	3
Palevo(7)	ns3.mclovin.org	4
Palevo(8)	bff4.7oorq8.com	1
	bff.7oorq8.com	
Palevo(9)	masterkey.com.ua	2
	mst.com.ua	
	bunker.org.ua	
	ssl.aukro.ua	

Unknown malicious group

Name	Domains	IP#
Nitol	@[a-z]{5,9}.3322.org (52)	195
N/A	google.2waky.com	1
	isasecret.com	1

Appendix C. Detection results for *KrTLD1* and *KrTLD2*

Unknown malicious group

Name	Domains	IP#
N/A	@[a-z]{3,17}.aghouse.kr (18)	247
	@[a-z]{3,17}.bsassy.kr (12)	
	@[a-z]{3,17}.dygl.kr (32)	
	@[a-z]{3,17}.f4u.kr (24)	
	@[a-z]{3,17}.fnplus.kr (16)	
	@[a-z]{3,17}.iticker.kr (13)	
	@[a-z]{3,17}.kyland.kr (13)	
	@[a-z]{2,17}.lemonsky.kr (17)	
	@[a-z]{2,17}.sc2424.kr (17)	
	@[a-z]{2,17}.shoepreme.kr (17)	
	@[a-z]{5,17}.sj4989.kr (11)	
	@[a-z]{3,17}.sweetaroma.kr (17)	
	@[a-z]{2,15}.ycosori.kr (16)	

Suspicious group

Name	Domains	IP#
N/A	@[a-z]{10}.interiorfine.kr (5,484)	114
	@[a-z]{10}.interni.kr (748)	
	@[a-z]{10}.project365.kr (1,695)	
	@[a-z]{3}.apzr.kr (24)	6
	@[a-z]{3}.bwep.kr (52)	
	@[a-z]{3}.jtvx.kr (96)	
	@[0–9]{4}[a-z0-9]{4,13}.b8z.kr (10)	9
	@[0–9]{4}[a-z0-9]{4,13}.q0z.kr (7)	
	@[a-z0-9]{5}.biztalkguy.kr (19)	223
	@[a-z0-9]{5}.gallery-m.kr (32)	
	@[a-z0-9]{7,13}.ayj.kr (7)	283
	@[a-z0-9]{5}.cyberna.kr (52)	
	@[a-z0-9]{3,5}.missingchild.kr (25)	
	@[a-z0-9]{2,12}.mirrorball.kr (22)	
	@[a-z0-9]{10,14}.star777.kr (6)	
	@[a-z0-9]{4,18}.villet.kr (28)	
	@[a-z]{5,10}.bo0.kr (19)	157
	@[a-z]{4,6}.brkfpeyz.kr (5)	6
	@[a-z]{4,8}.cakdlsjt.kr (52)	
	@[a-z]{4,7}.dltkfkadltksms.kr (17)	
	@[a-z]{3,5}.hoojo.kr (21)	
	@[a-z]{4,8}.pvkskfduw.kr (7)	
	@[a-z]{4,9}.xalefhoeg.kr (9)	
	@[a-z]{3}[0–9]{2,3}.yxqpo.kr (33)	
	@[a-z]{4,7}.zekflakh.kr (12)	
	@[a-z]{2,13}.tooz.kr (19)	110

References

[1] E. Cooke, F. Jahanian, D. McPherson, The zombie roundup: understanding, detecting, and disrupting botnets, in: Proceedings of the USENIX SRUTI Workshop, vol. 39, 2005, p. 44.

[2] N. Ianelli, A. Hackworth, Botnets as a Vehicle for Online Crime-coordination Center, CERT cMellon University, Carnegie CERT, 2005.

[3] N. Ianelli, A. Hackworth, Botnets as a vehicle for online crime, Int. J. Forensic Comput. Sci. IJoFCS 2 (1) (2007) 19–39.

[4] P. Bacher, T. Holz, M. Kotter, G. Wicherski, Know Your Enemy: Tracking Botnets, Technical Report, The Honeynet Project, 2005.

[5] T. Micro, Taxonomy of Botnet Threats, Whitepaper, November, 2006.

[6] McAfee, McAfee Thread Reports. http://www.mcafee.com/apps/viewall/publications.aspx, 2014.

[7] G. Bottazzi, G. Me, The botnet revenue model, Proceedings of the Seventh International Conference on Security of Information and Networks, ACM, 2014, p. 459.

[8] L. Trappeniers, M.A. Feki, F. Kawsar, M. Boussard, The internet of things: the next technological revolution, Computer 46 (2) (2013) 0024–25.

[9] E. Stinson, J.C. Mitchell, Characterizing bots remote control behavior, in: Proceedings of the Fourth International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2007, pp. 89–108.

[10] J. Kwon, J. Lee, H. Lee, Hidden bot detection by tracing non-human generated traffic at the zombie host, in: Proceedings of International Conference on Information Security Practice and Experience, Springer, 2011, pp. 343–361.

[11] K. Rieck, G. Schwenk, T. Limmer, T. Holz, P. Laskov, Botzilla: detecting the phoning home of malicious software, Proceeding of the ACM Symposium on Applied Computing, ACM, 2010, pp. 1978–1984.

[12] J. Goebel, T. Holz, Rishi: identify bot contaminated hosts by IRC nickname evaluation, Proceeding of the First Conference on Hot Topics in Understanding Botnets, Cambridge, MA, 2007, p. 8.

[13] G. Gu, R. Perdisci, J. Zhang, W. Lee, et al., Botminer: clustering analysis of network traffic for protocol-and structure-independent botnet detection., in: USENIX Security Symposium, 2008, pp. 139–154.

[14] C.J. Dietrich, C. Rossow, N. Pohlmann, Cocospot: clustering and recognizing botnet command and control channels using traffic analysis, Comput. Networks 57 (2) (2013) 475–486.

[15] H. Choi, H. Lee, H. Kim, Botgad: detecting botnets by capturing group activities in network traffic, Proceeding of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE, COMSWARE, ACM, 2009, p. 2.

[16] 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, Proceedings of the 16th ACM Conference on Computer and Communications Security, ACM, 2009, pp. 635–647.
[17] P. Porras, H. Saidi, V. Yegneswaran, An Analysis of Confickers Logic and Rendezvous Points, Tech. Rep., Computer Science Laboratory, SRI International, 2009.
[18] M. Antonakakis, J. Demar, C. Elisan, J. Jerrim, Dgas and Cybercriminals: A Case Study, 2012.
[19] L. Bilge, E. Kirda, C. Kruegel, M. Balduzzi, Exposure: finding malicious domains using passive DNS analysis, in: Proceedings of the Annual Network and Distributed System Security Symposium (NDSS), 2011.
[20] R. Begleiter, Y. Elovici, Y. Hollander, O. Mendelson, L. Rokach, R. Saltzman, A fast and scalable method for threat detection in large-scale DNS logs, 2013 IEEE International Conference on Big Data, IEEE, 2013, pp. 738–741.
[21] U. Mitra, A. Ortega, J. Heidemann, C. Papadopoulos, Detecting and identifying malware: a new signal processing goal, IEEE Signal Process. Mag. 23 (5) (2006) 107–111.
[22] B. AsSadhan, J.M. Moura, D. Lapsley, Periodic behavior in botnet command and control channels traffic, IEEE Global Telecommunications Conference (GLOBECOM), IEEE, 2009, pp. 1–6.
[23] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, C. Kruegel, Disclosure: detecting botnet command and control servers through large-scale netflow analysis, Proceedings of the 28th Annual Computer Security Applications Conference, ACM, 2012, pp. 129–138.
[24] F. Tegeler, X. Fu, G. Vigna, C. Kruegel, Botfinder: finding bots in network traffic without deep packet inspection, Proceeding of the Eighth International Conference on Emerging Networking Experiments and Technologies, ACM, 2012, pp. 349–360.
[25] Y. Qiao, Y.-x. Yang, J. He, C. Tang, Y.-z. Zeng, Detecting p2p bots by mining the regional periodicity, J. Zhejiang Univ. Sci. C 14 (9) (2013) 682–700.
[26] T. Paul, R. Tyagi, B. Manoj, B. Thanudas, Fast-flux botnet detection from network traffic, in: 2014 Annual IEEE India Conference (INDICON), IEEE, 2014, pp. 1–6.
[27] J. Bound, Y. Rekhter, Dynamic Updates in the Domain Name System (DNS Update), 1997.
[28] J. Nazario, T. Holz, As the net churns: fast-flux botnet observations, Third International Conference on Malicious and Unwanted Software (MALWARE), IEEE, 2008, pp. 24–31.
[29] L.A. Poyneer, J.-P. Veran, Toward feasible and effective predictive wavefront control for adaptive optics, SPIE Astronomical Telescopes+ Instrumentation, International Society for Optics and Photonics, 2008, p. 70151E.
[30] A. Schuster, On the investigation of hidden periodicities with application to a supposed 26 day period of meteorological phenomena, Terr. Magn. 3 (1) (1898) 13–41.
[31] G.T. Walker, Correlation in seasonal variations of weather VIII, Mem. India Meteor. Dept. 24 (1923) 75–131.
[32] R.A. Fisher, Tests of significance in harmonic analysis, in: Proceedings of the Royal Society of London, in: Series A, Containing Papers of a Mathematical and Physical Character, 1929, pp. 54–59.
[33] N.R. Lomb, Least-squares frequency analysis of unequally spaced data, Astrophys. Space Sci. 39 (2) (1976) 447–462.
[34] J.D. Scargle, Studies in astronomical time series analysis. II—statistical aspects of spectral analysis of unevenly spaced data, Astrophys. J. 263 (1982) 835–853.
[35] J.H. Horne, S.L. Baliunas, A prescription for period analysis of unevenly sampled time series, Astrophys. J. 302 (1986) 757–763.
[36] G. Hernandez, Time series, periodograms, and significance, J. Geophys. Res.: Space Phys. (1978–2012) 104 (A5) (1999) 10355–10368.
[37] A.V. Oppenheim, G.C. Verghese, Signals, Systems, and Inference,Class Notes for 6, 2010.
[38] C. Koen, Significance testing of periodogram ordinates, Astrophys. J. 348 (1990) 700–702.
[39] B. AsSadhan, J.M. Moura, An efficient method to detect periodic behavior in botnet traffic by analyzing control plane traffic, J. Adv. Res. 5 (4) (2014) 435–448.
[40] P.G. Hoel, et al., Introduction to Mathematical Statistics, second ed..
[41] M. Vlachos, S.Y. Philip, V. Castelli, On periodicity detection and structural periodic similarity, SDM, vol. 5, SIAM, 2005, pp. 449–460.
[42] Alexa, Alexa Top 500 Sites on the Web. http://www.alexa.com, 2014.
[43] Malware Domains, DNS-BH Malware Domain Block List. http://www.malwaredomains.com/, 2013.
[44] Palevo Tracker, Palevo Botnet Domains. https://palevotracker.abuse.ch/, 2013.
[45] ICS-SERT, Advisory (ICSA-10-090-01) Mariposa Botnet Domains. http://ics-cert.us-cert.gov/advisories/ICSA-10-090-01, 2013.
[46] Malware Domains List.. http://www.malwaredomainlist.com/, 2013.
[47] C. Estan, K. Keys, D. Moore, G. Varghese, Building a better netflow, ACM SIGCOMM Comput. Commun. Rev. 34 (4) (2004) 245–256.
[48] L. Braun, G. Munz, G. Carle, Packet sampling for worm and botnet detection in TCP connections, 2010 IEEE Network Operations and Management Symposium (NOMS), IEEE, 2010, pp. 264–271.
[49] J. Zhang, X. Luo, R. Perdisci, G. Gu, W. Lee, N. Feamster, Boosting the scalability of botnet detection using adaptive traffic sampling, Proceedings of the Sixth ACM Symposium on Information, Computer and Communications Security, ACM, 2011, pp. 124–134.
[50] K. Xu, D. Yao, Q. Ma, A. Crowell, Detecting infection onset with behavior-based policies, Fifth International Conference on Network and System Security (NSS), IEEE, 2011, pp. 57–64.
[51] L. Liu, S. Chen, G. Yan, Z. Zhang, Bottracer: Execution-based bot-like malware detection, in: Proceedings of the 11th Information Security Conference (ISC), Springer, 2008, pp. 97–113.
[52] G. Jacob, R. Hund, C. Kruegel, T. Holz, Jackstraws: Picking command and control connections from bot traffic., in: USENIX Security Symposium, 2011.
[53] S.S. Silva, R.M. Silva, R.C. Pinto, R.M. Salles, Botnets: a survey, Comput. Networks 57 (2) (2013) 378–403.
[54] G. Gu, P.A. Porras, V. Yegneswaran, M.W. Fong, W. Lee, Bothunter: detecting malware infection through IDS-driven dialog correlation, in: USENIX Security Symposium, 2007, pp. 1–16.
[55] 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.
[56] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, N. Borisov, Botgrep: finding p2p bots with structured graph analysis., in: USENIX Security Symposium, 2010, pp. 95–110.
[57] A. Ramachandran, N. Feamster, D. Dagon, et al., Revealing botnet membership using DNSBL counter-intelligence, in: Proceedings of Second USENIX Steps to Reducing Unwanted Traffic on the Internet, 2006, pp. 49–54.
[58] H. Choi, H. Lee, H. Lee, H. Kim, Botnet detection by monitoring group activities in DNS traffic, Seventh IEEE International Conference on Computer and Information Technology (CIT), IEEE, 2007, pp. 715–720.
[59] M. Feily, A. Shahrestani, S. Ramadass, A survey of botnet and botnet detection, Third International Conference on Emerging Security Information, Systems and Technologies (SECURWARE'09), IEEE, 2009, pp. 268–273.
[60] H. Choi, H. Lee, Identifying botnets by capturing group activities in DNS traffic, Comput. Networks 56 (1) (2012) 20–33.
[61] S. Yadav, A.K.K. Reddy, A. Reddy, S. Ranjan, Detecting algorithmically generated malicious domain names, Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, ACM, 2010, pp. 48–61.
[62] S. Yadav, A.K.K. Reddy, A.N. Reddy, S. Ranjan, Detecting algorithmically generated domain-flux attacks with DNS traffic analysis, IEEE/ACM Trans. Networking 20 (5) (2012) 1663–1677.
[63] R. Sharifnya, M. Abadi, A novel reputation system to detect DGA-based botnets, Third International eConference on Computer and Knowledge Engineering (ICCKE), IEEE, 2013, pp. 417–423.
[64] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou II, S. Abu-Nimeh, W. Lee, D. Dagon, From throw-away traffic to bots: detecting the rise of DGA-based malware., in: USENIX Security Symposium, 2012, pp. 491–506.
[65] R. Villamarín-Salomón, J.C. Brustoloni, Bayesian bot detection based on DNS traffic similarity, Proceedings of the 2009 ACM Symposium on Applied Computing, ACM, 2009, pp. 2035–2041.

Jonghoon Kwon received the B.S. degree in Computer Science and M.S. degree in Computer Science and Engineering from Korea University, Korea, in 2008 and 2010, respectively. He is currently working toward doctorate degree in Computer and Communication Security at Korea University, Korea. He was an intern at Microsoft Research Asia (Beijing, China) from 2012 to 2013. His research interests focus on network security, malware detection, and vulnerability detection.

Jehyun Lee received the B.S. and M.S. degrees in Computer Science and Engineering from Korea University, Korea, in 2007 and 2009. Currently, he is a Ph.D. candidate in Department of Computer Science and Radio Communication Engineering, Korea University. He was an intern at Microsoft Research Asia (Beijing, China) from 2013 to 2014. His research interests include network security and malware.

Heejo Lee is a professor at the Department of Computer Science and 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 postdoctorate at the Department of Computer Sciences and the security center CERIAS, Purdue University. He received his B.S., M.S., Ph.D. degrees in Computer Science and Engineering from POSTECH, Pohang, Korea. He serves as an editor of the Journal of Communications and Networks. He has been an advisory member of Korea Internet Security Agency and Korea Supreme Prosecutor's Office.

Adrian Perrig is a professor of Computer Science at the Department of Computer Science at the Swiss Federal Institute of Technology (ETH) in Zurich, where he leads the network security group. From 2002 to 2012, he was a professor of Electrical and Computer Engineering, Engineering and Public Policy, and Computer Science (courtesy) at Carnegie Mellon University. He served as the technical director for Carnegie Mellon's Cybersecurity Laboratory (CyLab). He earned his Ph.D. degree in Computer Science from Carnegie Mellon University under the guidance of J. D. Tygar, and spent 3 years during his Ph.D. degree at the University of California at Berkeley. He received his B.Sc. degree in Computer Engineering from the Swiss Federal Institute of Technology in Lausanne (EPFL). He is a recipient of the NSF CAREER award in 2004, IBM faculty fellowships in 2004 and 2005, the Sloan research fellowship in 2006, the Security 7 award in the category of education by the Information Security Magazine in 2009, the Benjamin Richard Teare teaching award in 2011, and the ACM SIGSAC Outstanding Innovation Award in 2013. His research revolves around building secure systems—in particular secure future Internet architectures.