Hindawi
Security and Communication Networks
Volume 2017, Article ID 5960307, 11 pages
https://doi.org/10.1155/2017/5960307

WILEY | Hindawi

Research Article

Detecting Web-Based Botnets Using Bot Communication Traffic Features

Fu-Hau Hsu,[1] Chih-Wen Ou,[1] Yan-Ling Hwang,[2] Ya-Ching Chang,[1] and Po-Ching Lin[3]

[1]*Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan*
[2]*School of Applied Foreign Languages, Chung Shan Medical University, Taichung, Taiwan*
[3]*Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan*

Correspondence should be addressed to Chih-Wen Ou; chihwen.frankou@gmail.com

Received 28 March 2017; Revised 18 June 2017; Accepted 25 September 2017; Published 3 December 2017

Academic Editor: Steffen Wendzel

Web-based botnets are popular nowadays. A Web-based botnet is a botnet whose C&C server and bots use HTTP protocol, the most universal and supported network protocol, to communicate with each other. Because the botnet communication can be hidden easily by attackers behind the relatively massive HTTP traffic, administrators of network equipment, such as routers and switches, cannot block such suspicious traffic directly regardless of costs. Based on the clients constituent of a Web server and characteristics of HTTP responses sent to clients from the server, this paper proposes a traffic inspection solution, called Web-based Botnet Detector (WBD). WBD is able to detect suspicious C&C (Command-and-Control) servers of HTTP botnets regardless of whether the botnet commands are encrypted or hidden in normal Web pages. More than 500 GB real network traces collected from 11 backbone routers are used to evaluate our method. Experimental results show that the false positive rate of WBD is 0.42%.

1. Introduction

A botnet is a group of compromised computers, namely, bots, controlled by one or multiple controllers [1–3]. These botnet controllers, also named bot masters, provide commands to their bots through C&C (Command-and-Control) servers so that the bots can perform actions for their bot masters. There are several criteria to categorize botnets, including the attacking behavior, C&C model, communication channel, rallying mechanism, and the evasion technique. We firstly focus on the centralized C&C model and discuss details about it in this study.

For a botnet with a centralized C&C model, each bot connects to its C&C server to retrieve commands or to deliver data. There are many advantages to use such an architecture to organize C&C servers and their bots compared to the decentralized and randomized models. The first advantage is the low cost to construct such a botnet, because bot masters can easily create this kind of botnets using many off-the-shelf open resources and applications. Meanwhile, the centralized model allows a bot master to quickly rally a large number of its bots by commanding few C&C servers. Such efficiency obviously facilitates cybercriminals to use botnets to conduct malicious activities, such as DDoS attacks and spamming [4].

According to the communication protocols used by botnets, botnets can be classified into several categories. These categories include the IRC- (Internet Relay Chat-) based botnet, the IM- (Instant Message-) based botnet, and the Web-based botnet. This paper focuses on the Web-based botnet, also named *HTTP botnet*, whose communication channel between the C&C server and its bot clients is via the HTTP. A C&C server of a Web-based botnet works like a normal Web server, and bot clients of a Web-based botnet work as normal Web clients. We call the C&C server of a Web-based botnet a *botnet Web server* hereafter. Two botnets, Spyeye [5] and Zeus [6], are well-known HTTP-based botnets. According to previous studies on these two botnets, there are several reasons why the HTTP is attractive to botnet owners. First, HTTP traffic is the most popular Internet traffic nowadays so that Web-based botnet traffic can be easily disguised as normal HTTP traffic, making the botnets more difficult to be discovered than those that use less popular protocols. Second, most network firewalls/proxies allow hosts behind them to access Internet via the HTTP. As a result,

Web-based botnets can easily provide stable and qualified client-to-server connectivity. Third, many promising solutions [1, 7–9] have been developed to precisely detect traditional IRC-based botnets instead of Web-based botnets. Therefore, the HTTP gradually becomes an ideal alternative protocol for botnet owners to use as the communication channel in recent years, and our study focuses on this kind of botnets.

1.1. Web-Based Botnet Detection. Bot clients are trojans, executable programs, or scripts running on compromised hosts. Hence, their behavior is different from human user behavior. Besides, their activity pattern, sizes, and transferred content are also different from human users. As programs generate the communication traffic automatically, the Web-based botnet communication has some prominent characteristics. According to our preliminary survey on a botnet taxonomy study [2], a typical bot client of a centralized C&C botnet often needs to synchronize with its botnet Web server to retrieve commands or deliver execution results. Such synchronization is often scheduled when bot clients are effectively controlled by botnet Web servers. Hence, we think that this phenomenon of synchronization can be utilized as a hint to indicate whether Web clients are controlled by human users or by bot clients. Besides, we also found that if a group of Web clients associated with a Web server consists of human users, each of them often has a different access pattern to the Web server. For example, these human clients may visit the Web server at different times of a day, or these clients may visit the Web server different numbers of times each day. On the contrary, if these Web clients are bot clients, which run programs or scripts, they may act together and behave similarly. Therefore, they may contact their botnet Web server repeatedly according to a predefined time interval to access commands from their botnet Web server. Such repeated contact to certain botnet Web servers may continue for several days, which is apparently different from normal human behavior. In addition, the same group of bot clients usually tends to communicate with the same botnet Web server. Based on the long-term repeated contact phenomenon and similar access pattern of the clients of a Web server, we use a metric named *Total Host Repetition Rate*, or *THR* in short, as one of our criteria to examine whether a Web server is a suspicious botnet C&C server.

Instead of THR, we also found that the payload inside the traffic between bot clients and their botnet Web server usually contains short and simple commands. Furthermore, all bot clients commanded by a certain C&C server tend to receive commands at the same time. This similarity of payloads among the bot clients controlled by the same botnet Web server is also described as the command-response pattern by BotProbe [7]. A normal Web server usually contains many Web pages and different users accessing different Web pages. Hence, unless the Web server contains only one Web page, the probability that its users retrieve the same Web page from the Web server simultaneously is low, and different Web pages usually have different sizes. As a result, during a period of time, the sizes of responding payloads of different Web clients accessing the same Web server are supposed to be different, while a botnet Web server often dispatches similar commands to its bot clients at the same time. Thus, we utilize the payload size difference as a metric called the *payload size similarity*, or *PSS* in short, to judge whether a Web server is a suspicious botnet C&C server or not. The formalization for these two metrics will be discussed later in Section 2.2.

In our prototype implementation, we designed an automatic mechanism based on the above two metrics and integrated this mechanism into our prototype system, named *Web-based Botnet Detector*, or *WBD* in short, to perform the inspection. WBD is attached to a network traffic monitoring system which is able to generate traffic logs from the online network stream and analyzes these logs simultaneously. Only few arithmetic calculations are required by WBD while performing runtime inspection on those monitored traffic logs. Such calculation brings significant overhead to other similar approaches during traffic inspection.

1.2. Contributions. The solution of this paper contains the following characteristics: (1) Compared to mainstream machine learning approaches which often rely heavily on tens or even hundreds of features, an approach with only few features can reduce notable overhead. WBD requires only several deterministic calculations which are easily extracted and calculated from monitored network traffic. (2) WBD inspects traffic of backbone networks. It does not require any program installed on network end-hosts and servers. (3) WBD does not use features based on traffic content mining. It does not rely on particular protocol-parsing as well. In summary, the contributions of WBD include the following.

(1) WBD requires only several deterministic calculations, which means that it is ideal to cooperate with heavy-loading backbone equipment.

(2) We conducted large-scale backbone data inspection for this study. It reveals those IP addresses and timestamps of Web servers that generate suspicious Web-based botnet communications across the global Internet.

(3) Due to the low correlation between the content itself, our solution can target the HTTPS protocol theoretically. Also, botnet owners may deliberately embed their botnet commands into some normal traffic, such as universal Web contents, to bypass the potential inspection along the traffic path. Our solution can work for this situation, because the calculation on THR and PSS requires the source and destination IP addresses of the packets in the traffic, which are not encrypted by most secure protocols.

This paper includes six sections. Section 2 will explain how we use features calculated from these criteria for the datagram-like network traffic logs. Sections 3 and 4 evaluate our approach and discuss issues including comparison with other similar approaches and the accuracy. Section 5 describes previous studies aiming at botnet related issues. Section 6 summaries this study.

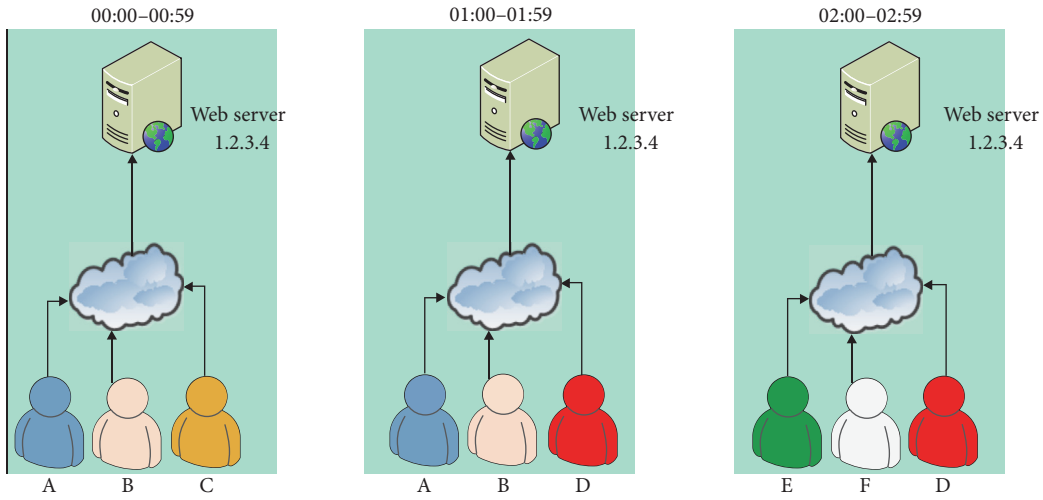

Figure 1: A communication pattern between 6 Web clients and a Web server during a three-hour monitoring period.

2. Methodology

In order to develop a traffic inspection approach, several issues have to be considered. These issues include the involved scale of monitoring, the volume of the traffic, and the feasibility to obtain such traffic. If an approach has to monitor the user-side traffic, an appropriate inspecting location may be at a gateway, a router, or a proxy (if it is mandatory for each network user) of the target user-side network. If an approach requires to monitor the server-side network, a possible location to do this work may be located at the intrusion detection equipment or firewall equipment of the target server-side network. These two deployments are commonly selected by many traffic inspection approaches because of their deployment feasibility and the affordable traffic volume. Different from these two categories, our approach aims at monitoring the global Internet as much as possible. The possible inspecting locations for such kind of approach should include backbone equipment that routes and processes large amount of IP packets. In our study, we are allowed to obtain the traffic from several online backbone routers in Taiwan so that we can develop a solution that is not specifically restricted by user-side or server-side networks.

Even though we are able to obtain logs from actual backbone routers, these routers are so important for our Internet service provider and they are always full-loaded. Hence, directly running inspecting procedures on them is certainly impractical so we adopted offline analyzing-after-recording method to make our experiments. We collected log samples from these backbone routers for several times and analyzed them. The detailed information about this will be described in Section 3. Due to many security and privacy concerns, all these actions were conducted and completed in an office of an Internet service provider. We cannot see the content of IP packet payloads and we cannot take any log-out from the office. Information about the recorded data from the traffic will be described later in Section 3.1.

2.1. A Case Study. To provide a clear understanding of our approach, considering an input case extracted from our logs depicted in Figure 1, there are six Web clients named from A to F requesting, respectively, a Web server with an IP address denoted as 1.2.3.4 hereafter. The three-hour long monitoring period is separated into three consecutive time intervals, as shown in Figure 1, and the length of each time interval is one hour. If clients A, B, and C make requests to the Web server in the first time interval, there will be arrows connecting them to the Web server, as shown in the left time interval marked with 00:00–00:59. Similarly, clients A and B repeat requesting and D makes the request in the second time interval. Client D repeats requesting, and clients E and F make requests in the third time interval in this case. A graphic representation of this case is shown in Figure 2. A Web server is denoted as an S-vertex, and a Web client is denoted as a C-vertex. The communication between a Web server and a client is denoted as an undirected edge connecting these two vertexes, as an example shown in Figure 2. There is no edge between two different C-vertexes because we do not need to consider the case when a Web client also runs a Web service. After all, we only focus on centralized botnets. We can also ignore edges between two S-vertexes because we only focus on communication made by Web clients. A graph is used to describe the communication patterns between Web clients and a Web server in a time interval. These graphs will not be used directly for graph computation. How these graphs are used will be described in Section 2.2.

2.2. Features Formulation. We use graphic representation only for conceptive discussion. For actual calculation conducted by WBD, equivalent formulas calculation is adopted after obtaining traffic logs. Such a design ensures that related calculation is theoretically light-weight, and such an approach is suitable for working with existing Internet backbone equipment. As we have mentioned in Section 1.1, two metrics are used to determine whether there exists botnet

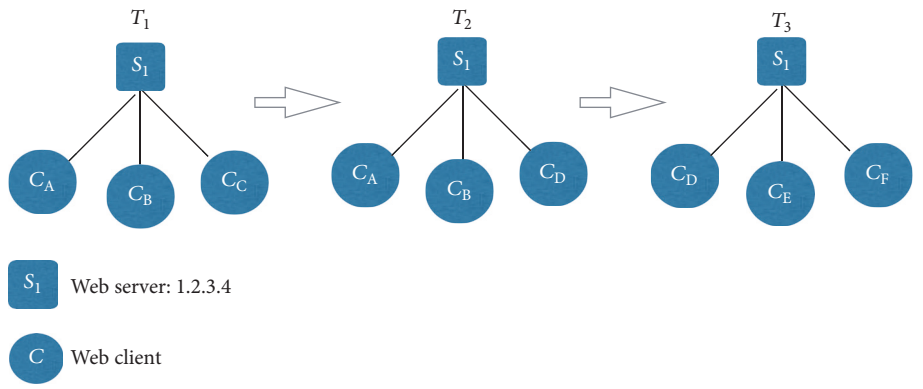

FIGURE 2: Graph representations of communication patterns between Web clients and their Web server at different monitoring time intervals.

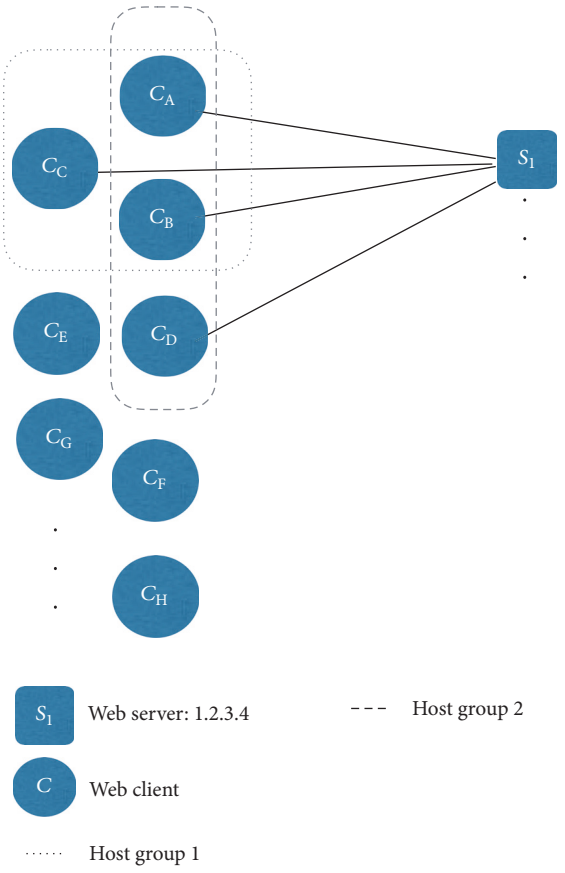

FIGURE 3: Two host groups related to Web server S_1 at two consecutive time intervals T_1 and T_2.

communication in the monitored HTTP traffic. In this paper, we call the group of Web clients that communicate with a Web server in a time interval a *host group*, as two groups of this case shown in Figure 3 show two host groups appearing at two different time intervals. If a Web server is a botnet Web server, the associated host groups are called *bot groups*. According to the previous studies [2, 10] and observation, the hosts of a bot group tend to communicate with the same botnet Web server all the time. Even though the constituent members of a bot group may change due to some technical issues or management reasons, such a change does not occur dramatically in a short period of time.

Two host groups, which are associated with the same Web server appearing in successive time intervals, are called *adjacent host groups*. We use two scores, *Access (AC)* score and *Total Host Repeat (THR)* score, to evaluate the THR feature of a Web server. Equation (1) defines these two scores, respectively. Score AC_s represents the number of total hosts communicating with Web server s in these n time intervals. For a certain time interval, the *HR* score is defined as the

proportion of the number of hosts appearing in both the current host group and its previously adjacent host group to the number of hosts in the current host group. The intersection of $\text{hostgroup}_{s_{t-1}}$ and hostgroup_{s_t} denotes the set of hosts appearing in both the adjacent host groups. Score THR_s is an average of n *Host Repeat (HR)* scores, denoting the similarity of the host groups of Web server s in n time intervals.

$$\begin{aligned} \text{AC}_s &= \sum_{t=1}^{n} \left|\text{hostgroup}_{s_t}\right| \\ \text{THR}_s &= \frac{1}{n}\sum_{t=1}^{n} \frac{\left|\text{hostgroup}_{s_{t-1}} \bigcap \text{hostgroup}_{s_t}\right|}{\left|\text{hostgroup}_{s_t}\right|}. \end{aligned} \tag{1}$$

Assume the host group of Web server s is hostgroup_{s_t} in time interval t and there are k_t different total responding payload sizes, $\text{PS}_{t_{k_1}}, \text{PS}_{t_{k_2}}, \ldots, \text{PS}_{t_{k_t}}$, for the $|\text{hostgroup}_{s_t}|$ hosts. $|\text{PS}_{t_{k_i}}|$ represents the number of hosts whose payload size is $\text{PS}_{t_{k_i}}$ in time interval t. Score *payload size similarity (PSS)* defined in (2) is used to evaluate the payload similarity feature of a Web server. Equation (2) gives its definition.

$$\text{PSS}_s = \frac{\sum_{t=1}^{n} \max_{i=1}^{k_t} \left(\left|\text{PS}_{t_i}\right|\right)}{\sum_{t=1}^{n} \left|\text{hostgroup}_{s_t}\right|} = \frac{\sum_{t=1}^{n} \max_{i=1}^{k_t} \left(\left|\text{PS}_{t_i}\right|\right)}{\text{AC}_s}. \tag{2}$$

2.3. WBD Classifier. After quantification of features for each input Web server, we can distinguish between normal Web servers and suspicious botnet Web server by comparing their THR, AC, and PSS scores to thresholds. In order to find the proper thresholds, we extracted the HTTP traffic traces of Alexa top 20 Websites in Taiwan from 11 backbone routers. The traffic related to these popular Websites is supposed to be nonbotnet traffic. However, when botnet traffic is hidden in the traffic of a social network Website, the social network traffic may contain botnet traffic. Hence, when collecting nonbotnet traffic, we can filter out social network related traffic first to avoid the above problem. However, in our traces, except Facebook which continues detecting and removing fake or malicious accounts, almost all of the top 20 Websites are nonsocial network Websites. Therefore, we consider the traffic associated with these popular Websites as benign. We calculated the THR score and AC score of each Web server and then selected the maximum THR score and AC score as the thresholds. Equation (3) shows the equations. THR_i represents the THR score of Web server i, and AC_i represents the AC score of Web server i.

$$\begin{aligned} \text{THR}_{\text{thershold}} &= \max_{i \in \text{top\_20\_benigh\_servers}} \left(\text{THR}_i\right) \\ \text{AC}_{\text{thershold}} &= \max_{i \in \text{top\_20\_benigh\_servers}} \left(\text{AC}_i\right). \end{aligned} \tag{3}$$

WBD uses the thresholds to examine Web servers appearing in the HTTP traffic traces and uses (4) to check whether Web server i exhibits the THR feature denoted by $\text{HR}_{\text{Susp\_Server}}(i)$.

$$\text{HR}_{\text{Susp\_Server}}(i) = \begin{cases} \text{True}, & \text{THR}_i > \text{THR}_{\text{thershold}} \wedge \text{AC}_i < \text{AC}_{\text{thershold}} \\ \text{False}, & \text{otherwise.} \end{cases} \tag{4}$$

WBD uses (5) to check whether Web server i exhibits the similar payload size feature denoted by notation $\text{PSS}_{\text{Susp\_Server}}(i)$. $\text{PSS}_{\text{threshold}}$ in (5) is defined as 0.5 because if in average the payload sizes of 50% hosts of each host group of a Web server during a time interval are similar to each other, the Web server is unlikely to be a normal Web server.

$$\text{PSS}_{\text{Susp\_Server}}(i) = \begin{cases} \text{True}, & \text{PSS}_i > \text{PSS}_{\text{threshold}} \\ \text{False}, & \text{otherwise.} \end{cases} \tag{5}$$

Based on the values determined by (4) and (5) for Web server i, WBD uses (6) to determine whether Web server i is a suspicious botnet Web server. All hosts which connect to it more than once are supposed to be its bot clients.

$$\text{Susp\_C\&C}(i) = \begin{cases} \text{True}, & \text{HR}_{\text{Susp\_Server}}(i) == \text{True} \vee \text{PSS}_{\text{Susp\_Server}}(i) == \text{True} \\ \text{False}, & \text{otherwise.} \end{cases} \tag{6}$$

WBD is built based on the above equations. There are four components in our prototype system. The first is to collect the raw data. The second is a module able to calculate THR and AC. The third is a module able to calculate PSS. The last is a combination of a report generator and a classifier operating according to the output from the second and the third modules.

3. Evaluation

The evaluation of WBD has two purposes. The first purpose is to discover appropriate thresholds of our solutions. The second purpose is to estimate the effectiveness of WBD. In order to evaluate the effectiveness of WBD, we need to know the number of botnet Web servers whose network traffic is recorded in our datasets. However, according to the phishing domain survey reports made by McGrath et al. [9] and Aaron et al. [11], attackers usually do not use a compromised host for more than a couple of days. Hence, we are not able to check all the Web servers in our collected datasets before attackers stop using some botnet Web servers that are hidden inside these large numbers of Web servers. Instead of checking all Web servers for entire datasets, we can only perform manual check on malicious hosts identified by WBD to determine whether they are truly malicious, so that we can at least calculate the false positive rate of WBD in our evaluation.

We collected network traces three times from 11 backbone routers in Taiwan. These routers belong to one of the three largest Internet service providers in Taiwan. Each collection generates a dataset. Each collection lasts for 48 hours to generate a dataset. Hence, we obtained 3 datasets. These routers

Table 1: Fields of a NetFlow V5 record.

Content	Bytes offset	Description
srcaddr	0–3	Source IP address
dstaddr	4–7	Destination IP address
dPkts	16–19	Packets in the flow
srcport	32-33	Source port number
dstport	34-35	Destination port number
prot	38	Protocol (6 = TCP, 17 = UDP)

Table 2: Information of our training phase dataset.

Time period	Size of the raw file	Number of Web servers
2013/03/16 00:00–2013/03/17 23:59	About 200 GB	9933

Table 3: Information of our testing phase datasets.

Index	Time period	Size of the raw file	Number of Web servers
1	2013/06/01 00:00–2013/06/02 23:59	About 140 GB	156294
2	2013/01/18 00:00–2013/01/19 23:59	About 160 GB	170920

Table 4: Results of the testing phase.

Index	False positive	Number of suspicious Web servers
1	3 (0.28%)	1047
2	9 (0.83%)	1085
Total	12 (0.42%)	2132

are Cisco routers equipped with NetFlow [12]. Therefore, these datasets were recorded in the NetFlow V5 compatible format. Our experiments include two phases. The first is the training phase which is used to determine the thresholds using the first dataset. The second is the testing phase which uses the other two datasets.

3.1. NetFlow. NetFlow is able to record all traffic passing through a Cisco router. It fetches data from IP packets and generates flow records. Those flow records can be transferred to other devices for further analysis. The source address field *srcaddr*, destination address field *dstaddr*, source port field *srcport*, destination port field *dstport*, and protocol field *prot* of a NetFlow V5 record specify a session between a certain source host and a destination host via the HTTP, as shown in Table 1. The *dPkts* field contains the raw packet data, so that we can calculate the payload size of a packet and the total payload size of an HTTP session.

3.2. Threshold and Training Phase. The first part of our experiments is to calculate the thresholds and perform training. The training data were collected from March 16 to 17 in 2013. The number of backbone routers involved in this phase is less than the number of routers in the testing phase because we chose the routers which forward packets to popular Web servers in this phase. As shown in Table 2, the total raw data size in this phase is about 200 GB which consists of the IP addresses of 9,933 Web servers. The THR scores of Alexa top 20 popular Websites in Taiwan are all less than 0.521. We also used a browser to manually connect to the 9,933 Web servers to check which of them are normal Web servers and which of them are abnormal. The THR scores of the above normal Web servers are almost all less than 0.521. In contrast, the THR scores of the above abnormal Web servers are almost all greater than 0.521. Therefore, we set $\mathrm{THR}_{\mathrm{threshold}}$ as 0.521 and set $\mathrm{AC}_{\mathrm{threshold}}$ as 12,000. Besides, $\mathrm{PSS}_{\mathrm{threshold}}$ is set to 0.5 as described in previous subsection. We also calculated the average Web page size for these top 20 Websites, and the size is 47,087 bytes.

3.3. Testing Phase. In the testing phase, two datasets were used. Table 3 shows the information of these samples. More than 300 GB data were used in our analysis. These two datasets contain network traces of 156,294 and 170,920 Web servers, respectively. Among these Web servers, WBD found 1,047 suspicious botnet Web servers from testing dataset 1 and 1,085 suspicious servers from testing dataset 2. For each of these 2,132 suspicious botnet Web servers, we use a browser to manually check their content. If a suspicious botnet Web server replies to a normal Web page, we treat this case as a false positive case. Besides, bot clients usually retrieve commands from their botnet Web server, and the sizes of the commands are supposed to be smaller than the size of normal Web pages. Therefore, if the size of data returning from a Web server is greater than 47,087 bytes, we will deem the Web server as a normal one and also treat this case as a false positive case. The result of the testing phase is shown in Table 4. To calculate the false negative rate, we need to

Table 5: Features and classifiers used by four similar approaches.

Approaches	Features	Classifier
Venkatesh and Nadarajan	ORT, RIO, PT, SYN, FIN, PSH	Neural network
Zhao et al.	PX, PPS, NR, APL, FPS, PV, FPH, TBP	Decision tree
Cai and Zou	SHH, CC, SCL, BIC, PR, DWS	Multilayer filter
WBD	AC, THR, PSS	Decision tree

Table 6: Comparisons of features used by four approaches.

Calculations	Venkatesh and Nadarajan	Zhao et al.	Cai and Zou	WBD
Counting specific packets	ORT, RIO, PT, SYN, FIN, PSH	PX, PPS, NR	—	—
Arithmetic based on packet size	—	APL, FPS, PV	SHH, CC, SCL	PSS
Arithmetic based on numbers of hosts	—	FPH	BIC	AC, THS
Arithmetic based on interpacket timing	—	TBP	PR	—
Host fingerprinting	—	—	DWS	—

manually check 327,214 Web servers to confirm the botnet Web servers within them. However, according to the phishing domain survey reports made by McGrath et al. [13] and Aaron et al. [14], attackers usually do not use a compromised host for more than a couple of days. Because we are not able to check all the 327,214 Web servers before attackers stop using some botnet Web servers that are hidden inside these 327,214 Web servers, currently we are not able to calculate the false negative rate of WBD. However, when comparing with a malicious IP list provided by ICST [15], we found that the majority of the botnet Web servers we found are not in the list, which shows that WBD provides a list of originally unknown botnet Web servers to system administrators.

4. Discussion

Some approaches aiming at detecting HTTP botnets were also proposed in recent years. These approaches use various features to inspect network traffic to detect HTTP botnets. Three of such approaches are selected and compared with WBD. Table 5 lists these approaches with their features and classifiers. Venkatesh and Nadarajan [16] proposed a multilayer feedforward neural network solution with six features, including one-way ratio of TCP packets (ORT), ratio of incoming to outgoing TCP packets (RIO), the proportion of TCP packets in the flow (PT), and TCP flags counting on SYN, FIN, and PSH flags. These features require only counting specific packets, so that they increase relatively slight performance overhead compared to other complex features used by the rest of the approaches. Such counting-based features are simple and can be manipulated by communicators, so that botnet owners who are aware of such features can bypass the detection by specifically changing their forms of communication packets. Zhao et al. [17] proposed a solution with eight features. Three of them are related to counting-based features including the number of packets exchanged (PX), the number of packets exchanged per second in short time interval (PPS), and the number of reconnections (NR). Three of them are related to arithmetic operations based on the packet payload size, including the average payload packet length (APL), the variance of the payload packet length (PV), and the size of the first packet (FPS). One of the two remaining features involves arithmetic calculations for the number of flows from this address over the total number of flows generated per hour (FPH), and the other feature calculates the average time interval between two consecutive packets (TBS). This approach has higher accuracy than the previous study of Venkatesh and Nadarajan, and its performance overhead is certainly increased due to involving more complicated features compared to the previous study. Cai and Zou [10] proposed a solution with six features. Three of them require arithmetic operations based on the packet payload size, including short HTTP header (SHH), constant content (CC), and short content length (SCL). One feature is related to the bot IP clustering (BIC), one focuses on the periodical request (PR), and the last one requires the host fingerprinting among Web servers to estimate the extent of diversified Web services (DWS). Although this approach has the comprehensive discussion about the features of HTTP botnet and comes up with a set of complicated features that is suitable for determining the existence of botnet communication precisely, the performance overhead is still a significant issue. Many complex features, especially the DWS feature, are involved in this approach for traffic inspection.

Based on the above discussion, we discovered that some kinds of calculations are commonly required by some of these four approaches. Table 6 describes the summarization. Both the study of Venkatesh and Nadarajan and the study of Zhao et al. count specific packets. Both the study of Zhao et al. and the study of Cai and Zou have features which require performing arithmetic operations based on the packet payload size or based on interpacket timing. Three approaches, including WBD, have features requiring execution of arithmetic operations based on the numbers of hosts and the packet size. However, WBD uses only three features requiring execution of arithmetic operations based on numbers of hosts and the packet payload size. Compared to the study of Venkatesh and Nadarajan and the study of Zhao et al., WBD does not need to count specific packets, so that botnet owners have fewer opportunities to bypass WBD. Besides,

Table 7: Comparison among false positive rates of four similar approaches.

Approaches	False positive rates of various test datasets
Venkatesh and Nadarajan	Spyeye-1 (0.97%), Spyeye-2 (0.98%), Zeus-1 (0.99%), Zeus-2 (0.96%)
Zhao et al.	BlackEnergy (0%), Weasel (82%)
Cai and Zou	SJTU1 (17.6%), SJTU2 (26.3%), QingPu (13.6%)
WBD	0.42%

unlike the study of Cai and Zou, WBD does not apply time consuming features such as features of interpacket timing and host fingerprinting so that WBD has limited performance overhead and is able to complete classification in time.

To evaluate the effectiveness of their approaches, each of these four similar approaches used their own datasets to obtain the false positive rates of the chosen datasets. Table 7 lists test datasets and respective false positive rates of these four similar approaches. Four datasets were used by Venkatesh and Nadarajan, and all false positive rates of these datasets are under 1%. For the false positive rates of Zhao et al., the false positive rate of test dataset BlackEnergy is 0%. Dataset BlackEnergy is a pure botnet traffic dataset. The false positive rate of test dataset Weasel is 82%. Dataset Weasel contains normal traffic. The authors analyzed these 82% false positives (2902 false alerts) and discovered that all of these false alerts belong to six applications. They claimed that once a whitelist is adopted for their approach, these false positives would be reduced. The study of Cai and Zou used three datasets to test their approach. The false positive rates range from 13.6% to 26.3%. WBD used logs directly captured from backbone routers and the false positive rate is 0.42%. Compared to other three approaches, WBD is better than the study of Venkatesh and Nadarajan and the study of Cai and Zou. WBD does not need a prebuilt whitelist to remove normal applications before detection.

4.1. False Positives. The total false positive rate of our study is 0.42%. This excellent accuracy results from the adoption of THS and PSS. In fact, many existing front-end Web applications may repeat contacting a Web server. The Web-based instant messenger is one of the typical examples where Web clients contact their Web servers repeatedly. However, as mentioned in previous paragraph, other related approaches may not distinguish the differences between a botnet and a Web server functioning as a Web-based instant messenger.

4.2. False Negatives. Due to the reasons described in this subsection, currently we are not able to discuss the false negative of our work. According to the phishing domain survey reports made by McGrath et al. [13] and Aaron et al. [14], attackers usually do not use a comprised host for more than a couple of days. Apparently, we are not able to check all the Web servers classified as benign in our datasets in time before most attackers stop using botnet Web servers that are hidden inside these large amounts of Web servers. Compared to several previous similar studies [10, 16, 17], most of them evaluate their solution by the datasets containing specific botnets of Spyeye [5] and Zeus [6] instead of real live traffic from Internet. This means that these similar approaches may be accurate when they are applied for detecting those botnets which have similar characteristics to Spyeye and Zeus, but the accuracy is not evaluated for other Web-based botnets. However, most botnet owners keep changing attributes and characteristics of their botnets to avoid being detected. Another reason why false negatives sometimes are impractical is that the Web-based botnets may provide legal online Web services simultaneously. Mostly they may act like normal Web services, and it is very difficult, if not impossible, to enumerate all Internet Web servers having such a characteristic. All issues listed here lead to the uncertainty of the discussion about the false negatives. We will keep discussing this issue in our future work.

4.3. Detection Evasion. Experimental results show that WBD is an ideal solution for Web-based botnet detection. However, current Web-based botnets may change their designs to bypass the detection of WBD. For example, bot clients may connect to their C&C server at nonadjacent time intervals, or various lengths of gibberish bytes may be added to the response payloads of different bot clients to diversify the response lengths. However, such evasion methods may create several drawbacks in the modified botnets. First, this makes the design and operation of a botnet much more complicated because a botnet needs to coordinate the action of each bot client. Second, gibberish bytes increase network traffic. Besides, if different bot clients use the same URL but get Web pages with different lengths, this may be a sign that the related server is not a normal Web server. Besides, C&C servers may apply fast-flux domain technique to change their IP addresses frequently in a very short period of time. Botnets with such ability theoretically possibly bypass WBD deliberately with the price that all bots need to connect and disconnect different hosts frequently, which makes them much more detectable by system/network administrators. In the literature of fast-flux research, an approach proposed by Hsu et al. [18] has been developed to detect fast-flux domains from a single host without using router traces. Hence, by integrating both kinds of approaches, we can create an effective method to detect various Web server-based botnets.

The goal of this paper is to find the botnet Web servers. However, during the detection, we can also obtain the hosts, that is, bot clients, that connect to the botnet Web servers. Hence, in our future work, we will make more detailed survey to find the properties of bot clients. Moreover, this study also works for detecting botnet Web servers communicating with their bot clients via the HTTPS channel because the detection relies only on unencrypted parts of IP packets instead of inspecting the payload content. The unencrypted parts include the information of the source host and

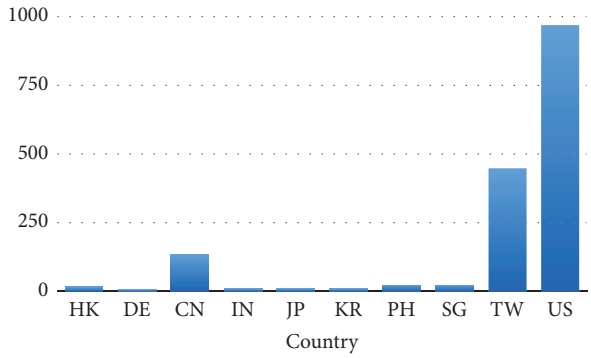

Figure 4: C&C server locations.

destination host and the payload size. Furthermore, the THR feature and the PSS feature will not be changed by modifying the content that a botnet Web server sends to its bot clients. Hence, even if a bot master hides its command inside a normal-looking Web page, WBD is still able to detect it. After detecting a list of C&C servers, we also survey the distribution of the locations of these C&C servers. Figure 4 shows the locations of the 1,085 C&C servers that WBD detects from testing dataset 2. The majority of them are located in the USA and Taiwan. Some of them are located in China, Singapore, Philippines, and so on.

5. Related Work

Most previous studies aim at generic botnet detection. Gu et al. proposed several correlation-based detection solutions: BotMiner [19], BotSniffer [3], BotProbe [7], and BotHunter [20]. BotMiner is a well-known network level correlation-based and protocol-structure independent solution. It performs the connection behavior (C-Plane) and attack behavior (A-Plane) clustering and then performs cross-plane correlation to build a model for botnet C&C servers. BotMiner requires some real-world C&C server network traces in the training phase. However, such reliable C&C traces are not always available in practice. BotSniffer is also a correlation-based solution able to detect C&C servers in a port-independent manner. It is composed of a protocol matcher, an activity/message response detector, and a correlation engine. The correlation engine runs group activity/message response analysis based on the outputs from this protocol matcher and response detectors without requiring other prior knowledge of these botnet C&C servers. Only few packets are needed for training BotSniffer, and it also works well at detecting small botnets. BotProbe is a behavior-based solution, specifically focusing on the command-response pattern of the botnet and its deterministic behavior (for the stateless bot client). BotMiner, BotSniffer, and BotProbe have some problems when the botnet attempts to avoid such detection. The possible evasions include using strong encryption, using atypical response, and injecting random noise packets. Especially for BotMiner, the botnet can create a specific evasion for bypassing the C-plane and A-plane clustering. BotProbe has assumptions that the input has to be perspective and the chat protocol between bots has to be available for the detection engine. BotHunter detects botnets based on the bot-specific heuristics and the IDS dialog-based correlation. The IDS dialogs represent different stages of a botnet life cycle. Such correlation can produce signatures for IDS systems. This IDS-driven strategy has a problem when detecting encrypted botnet communications. Furthermore, this solution also has weaknesses similar to IDS, and the signature generation and update problems must be overcome to reach ideal detection performance.

Yu et al. proposed the SBotMiner [21], an approach based on large-scale network traffic filtering aiming at detecting search bots, which often perform suspicious search activities on the Internet, and SBotMiner uses PCA (Principle Component Analysis) to separate the bot traffic from the benign user traffic. This approach suffers from noise-queries because the search bots can generate lots of meaningless search activities to decrease the detection performance considerably. Karasaridis et al. proposed a wide network traffic correlation solution [9]. However, it only focuses on IRC-based botnet and needs many kinds of prior knowledge before performing the correlation. Zand et al. proposed an approach [22] to automatically extract Command-and-Control signatures for detecting botnets. Since the signature generation is based on the extraction of frequent communication patterns, it is also not applicable to encrypted communication. Wang et al. proposed a fuzzy pattern-based filtering algorithm [23]. This algorithm depends on the DNS query patterns, so that the botnet, especially for the Web-based botnet, can easily avoid the filtering by directly using IP address to communicate.

Some recent research aims at detecting the decentralized peer-to-peer (P2P) botnets. Zhang et al. proposed an approach [24] aiming at detecting P2P botnets. Using decentralized architecture greatly increases the survivability because most botnet takedown actions target C&C servers. However, the decentralized architecture also has some critical disadvantages. P2P botnets often have a complex architecture. Hence, maintaining a P2P network always demands significant technical efforts. In addition, its non-client-server architecture makes it inappropriate to be integrated into existing Web services. Other early studies discussed and evaluated the scale and the takedown techniques of a botnet. Both Dagon et al. [2] and Khattak et al. [25] discussed how different kinds of botnets are organized and what activities they may have. Abu Rajab et al. focused on botnet scale evaluation [1], and Stone-Gross et al. addressed detailed issues of taking down a botnet [26]. Honeypots are often used to collect or observe malicious network traffic in early botnet research. However, honeypots usually do not provide outgoing communication. Therefore, they are not suitable for collecting botnet traffic. Nadji et al. proposed a system for the botnet takedowns [27]. Such botnet takedown solution aims at stopping those DNS servers from functioning in the botnet communication. However those C&C servers are able to reorganize using other DNS servers rapidly, since this approach targets deactivating the botnet communication, not removing botnet C&C servers.

Most approaches mentioned so far may be able but not specifically designed to detect Web-based botnet. Since the

majority of botnet owners seldom use their own hosts as the C&C servers, they often use compromised hosts instead. In other words, there must be the HTTP-supported malware on those compromised hosts performing C&C server-like operations. According to the study [28] proposed by Perdisci et al., they addressed the concrete relationship between HTTP-supported malware and Web-based botnets. They also proposed an approach to detect the HTTP-supported malware by using malicious network traces. This approach uses behavioral clustering of these HTTP-supported malware samples by finding their structural similarities among the sequences of HTTP requests. The results of behavioral clustering are used for generating signatures for an IDS system. Since they look into the sequences of HTTP requests, it means that this approach cannot be used for HTTPS-based malware. In addition, this approach still suffers from similar evasions mentioned so far, including injecting noise sequences of HTTP requests and implementing HTTP requests in a time triggering oriented approach.

Many recent botnet studies focus on problems brought by new types of botnets which utilize currently popular Internet applications. For example, some of these papers aim at detecting botnets running on social networks. Kartaltepe et al. proposed a study focusing on the social network-based botnet [11]. Wang et al. proposed an approach [29] to detect the DGA botnet by utilizing social network analysis. Venkatesh and Nadarajan proposed a survey of Stegobot [30], which is a kind of botnets using steganography to mask crucial information in digital images and then transmitting the images over social networks. Ferrara et al. addressed the rise of botnets running on social networks in a recent article [31]. Botnets utilizing IoT and mobile devices were also addressed by several prestigious conferences and projects recently. Bertino and Islam addressed the issues related to botnets and Internet of Things (IoT) security [32]. Project [33], conducted in 2017, is also motivated by Bertino and Islam to analyze the DDoS attack via IoT botnets. Mobile devices suffer from vulnerabilities as well as untrusted firmware and are also vulnerable to botnet owners. Eslahi et al. unveiled MoBots [34], which represent those botnets on mobile devices and networks. MoBots may use some existing services, such as SMS, to communicate with their bot masters. Such issue is critical to the telecommunication industry. Therefore, there is a related patent [35], which has been filed in 2016, disclosing a method for SMS-based botnet detection. Social network-based botnets, IoT botnets, or even mobile device-based botnets are not typical Web-based botnets. To be able to communicate in multiple mechanisms, they are more complicated than traditional Web-based botnets.

6. Conclusion

This study proposes a solution called WBD to detect suspicious Web-based botnets, no matter whether the botnet communication is encrypted or hidden in normal Web pages. We propose three features, two of them related to robot-like repeated contact clustering and one of them related to similar payload size, to detect the existence of botnet Web servers within the network communication. Applying our solutions to 500 GB practical network traces, we found the false positive rate of WBD is only 0.42%.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was funded by the projects of Ministry of Science and Technology of Taiwan under no. 105-2221-E-008-074-MY3 and no. 106-3114-E-002-005.

References

[1] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM on Internet Measurement Conference, IMC 2006*, pp. 41–52, Brazil, October 2006.

[2] D. Dagon, O. Gu, C. P. Lee, and W. Lee, "A taxonomy of botnet structures," in *Proceedings of the 23rd Annual Computer Security Applications Conference, ACSAC 2007*, pp. 325–338, Miami Beach, Fla, USA, December 2007.

[3] G. Gu, J. Zhang, and W. Lee, Botsniffer: Detecting botnet command and control channels in network traffic. In NDSS. The Internet Society, 2008.

[4] FBI. Botnets 101 What They Are and How to Avoid Them, 2013.

[5] A. K. Sood, R. J. Enbody, and R. Bansal, "Dissecting spyeye-understanding the design of third generation botnets," *Computer Networks*, vol. 57, no. 2, pp. 436–450, 2013.

[6] H. Binsalleeh, T. Ormerod, A. Boukhtouta et al., "On the analysis of the Zeus botnet crimeware toolkit," in *Proceedings of the 2010 8th International Conference on Privacy, Security and Trust, PST 2010*, pp. 31–38, Canada, August 2010.

[7] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee, "Active botnet probing to identify obscure command and control channels," in *Proceedings of the 25th Annual Computer Conference Security Applications, ACSAC 2009*, pp. 241–253, Honolulu, Hawaii, USA, December 2009.

[8] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Using machine learning techniques to identify botnet traffic," in *Proceedings of the 31st Annual IEEE Conference on Local Computer Networks (LCN '06)*, pp. 967–974, Tampa, Fla, USA, November 2006.

[9] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets, HotBots'07*, 7 pages, USENIX Association, Berkeley, Calif, USA, 2007.

[10] T. Cai and F. Zou, "Detecting HTTP botnet with clustering network traffic," in *Proceedings of the 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '12)*, pp. 1–7, September 2012.

[11] E. J. Kartaltepe, J. A. Morales, S. Xu, and R. Sandhu, "Social network-based botnet command-and-control: emerging threats and countermeasures," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 6123, pp. 511–528, 2010.

[12] Cisco. NetFlow Services Solutions Guide, 2007.

[13] Behind Phishing: An Examination of Phisher Modi Operandi.

[14] Global Phishing Survey: Trends and Domain Name Use in 2H2009.

[15] Information and Communication Security Technology Center.

[16] G. K. Venkatesh and R. A. Nadarajan, "HTTP botnet detection using adaptive learning rate multilayer feed-forward neural network," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7322, pp. 38–48, 2012.

[17] D. Zhao, I. Traore, B. Sayed et al., "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[18] F.-H. Hsu, C.-S. Wang, C.-H. Hsu, C.-K. Tso, L.-H. Chen, and S.-H. Lin, "Detect fast-flux domains through response time differences," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 10, pp. 1947–1956, 2014.

[19] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th Conference on Security Symposium, SS'08*, pp. 139–154, USENIX Association, Berkeley, Calif, USA, 2008.

[20] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, SS'07*, pp. 12:1–12:16, Berkeley, Calif, USA, 2007.

[21] F. Yu, Y. Xie, and Q. Ke, "SBotMiner: Large scale search bot detection," in *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining, WSDM 2010*, pp. 421–430, USA, February 2010.

[22] A. Zand, G. Vigna, X. Yan, and C. Kruegel, "Extracting probable command and control signatures for detecting botnets," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC 2014*, pp. 1657–1662, Republic of Korea, March 2014.

[23] K. Wang, C. Huang, S. Lin, and Y. Lin, "A fuzzy pattern-based filtering algorithm for botnet detection," *Computer Networks*, vol. 55, no. 15, pp. 3275–3286, 2011.

[24] J. Zhang, R. Perdisci, W. Lee, X. Luo, and U. Sarfraz, "Building a scalable system for stealthy P2P-botnet detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 1, pp. 27–38, 2014.

[25] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, "A Taxonomy of botnet behavior, detection, and defense," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 898–924, 2014.

[26] B. Stone-Gross, M. Cova, L. Cavallaro et al., "Your botnet is my botnet: Analysis of a botnet takeover," in *Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09*, pp. 635–647, New York, NY, USA, November 2009.

[27] Y. Nadji, M. Antonakakis, R. Perdisci, D. Dagon, and W. Lee, "Beheading hydras: Performing effective botnet takedowns," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013*, pp. 121–132, Germany, November 2013.

[28] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of http-based malware and signature generation using malicious network traces," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI'10*, 26 pages, Berkeley, Calif, USA, 2010.

[29] T.-S. Wang, C.-S. Lin, and H.-T. Lin, "DGA botnet detection utilizing social network analysis," in *Proceedings of the 2016 IEEE International Symposium on Computer, Consumer and Control, IS3C 2016*, pp. 333–336, China, July 2016.

[30] N. Venkatachalam and R. Anitha, "A multi-feature approach to detect Stegobot: a covert multimedia social network botnet," *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 6079–6096, 2017.

[31] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[32] E. Bertino and N. Islam, "Botnets and internet of things security," *The Computer Journal*, vol. 50, no. 2, Article ID 7842850, pp. 76–79, 2017.

[33] R. Hallman, J. Bryan, G. Palavicini, J. Divita, and J. Romero-Mariona, Ioddos the internet of distributed denial of sevice attacks, 2017.

[34] M. Eslahi, R. Salleh, and N. B. Anuar, "MoBots: A new generation of botnets on mobile devices and networks," in *Proceedings of the 2012 IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE 2012*, pp. 262–266, Malaysia, December 2012.

[35] C. Adams, Sms botnet detection on mobile devices, May 24 2016. US Patent 9, 351, 167.