Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2009, Article ID 692654, 11 pages
doi:10.1155/2009/692654

Review Article

Botnet: Classification, Attacks, Detection, Tracing, and Preventive Measures

Jing Liu,[1] Yang Xiao,[1] Kaveh Ghaboosi,[2] Hongmei Deng,[3] and Jingyuan Zhang[1]

[1] *Department of Computer Science, The University of Alabama, Tuscaloosa, AL 35487-0290, USA*
[2] *The Centre for Wireless Communications, University of Oulu, P.O. Box 4500, FI-90014, Finland*
[3] *Intelligent Automation, Inc., Rockville, MD 20855, USA*

Correspondence should be addressed to Yang Xiao, yangxiao@ieee.org

Received 25 December 2008; Revised 17 June 2009; Accepted 19 July 2009

Recommended by Yi-Bing Lin

Botnets become widespread in wired and wireless networks, whereas the relevant research is still in the initial stage. In this paper, a survey of botnets is provided. We first discuss fundamental concepts of botnets, including formation and exploitation, lifecycle, and two major kinds of topologies. Several related attacks, detection, tracing, and countermeasures, are then introduced, followed by recent research work and possible future challenges.

1. Introduction

The untraceable feature of coordinated attacks is just what hackers/attackers demand to compromise a computer or a network for their illegal activities. Once a group of hosts at different locations controlled by a malicious individual or organization to initiate an attack, one can hardly trace back to the origin due to the complexity of the Internet. For this reason, the increase of events and threats against legitimate Internet activities such as information leakage, click fraud, denial of service (DoS) and attack, E-mail spam, etc., has become a very serious problem nowadays [1]. Those victims controlled by coordinated attackers are called zombies or bots which derives from the word "robot." The term of bots is commonly referred to software applications running as an automated task over the Internet [2]. Under a command and control (C2, or C&C) infrastructure, a group of bots are able to form a self-propagating, self-organizing, and autonomous framework, named botnet [3]. Generally, to compromise a series of systems, the botnet's master (also called as herder or perpetrator) will remotely control bots to install worms, Trojan horses, or backdoors on them [3]. The majority of those victims are running Microsoft Windows operating system [3]. The process of stealing host resources to form a botnet is so called "scrumping" [3].

Fortunately, botnet attacks and the corresponding preventive measures or tracking approaches have been studied by industry and academia in last decades. It is known that botnets have thousands of different implementations, which can be classified into two major categories based on their topologies [4]. One typical and the most common type is Internet Relay Chat-(IRC-) based botnets. Because of its centralized architecture, researchers have designed some feasible countermeasures to detect and destroy such botnets [5, 6]. Hence, newer and more sophisticated hackers/attackers start to use Peer to Peer (P2P) technologies in botnets [4, 7]. P2P botnets are distributed and do not have a central point of failure. Compared to IRC-based botnets, they are more difficult to detect and take down [4]. Besides, most of its existing studies are still in the analysis phase [4, 7].

Scholars firstly discovered botnets due to the study on Distributed DoS (DDoS) attacks [8]. After that, botnet features have been disclosed using probing and Honeypots [9–11]. Levy [12] mentioned that spammers increasingly relied on bots to generate spam messages, since bots can hide their identities [13]. To identify and block spam, blacklists are widely used in practice. Jung and Sit [14] found that 80% of spammers could be detected by blacklists of MIT in 2004. Besides, blacklists also impact on other hostile actions. Through examining blacklist abuse by botnet's

masters, Ramachandran et al. [15] noted that those masters with higher premiums on addresses would not present on blacklists. Thus, only deploying blacklists may be not enough to address the botnet problem.

So far, industry and much of academia are still engaged in damage control via patch-management rather than fundamental problem solving. In fact, without innovative approaches to removing the botnet threat, the full utility of the Internet for human beings will still be a dream. The major objective of this paper is to exploit open issues in botnet detection and preventive measures through exhaustive analysis of botnets features and existing researches.

The rest of this paper is organized as follows. In Section 2, we provide a background introduction as well as the botnet classification. Section 3 describes the relevant attacks. Section 4 elaborates on the detection and tracing mechanisms. We introduce preventive measures in Section 5. The conclusion and future challenges are discussed in Section 6.

2. Classification

Botnets are emerging threats with billions of hosts worldwide infected. Bots can spread over thousands of computers at a very high speed as worms do. Unlike worms, bots in a botnet are able to cooperate towards a common malicious purpose. For that reason, botnets nowadays play a very important role in the Internet malware epidemic [16]. Many works try to summarize their taxonomy [17, 18], using properties such as the propagation mechanism, the topology of C2 infrastructure used, the exploitation strategy, or the set of commands available to the perpetrator. So far, botnet's master often uses IRC protocol to control and manage the bots. For the sake of reducing botnet's threat efficiently, scholars and researchers emphasize their studies on detecting IRC-based botnets. Generally speaking, the academic literature on botnet detection is sparse. In [19], Strayer et al. presented some metrics by flow analysis on detecting botnets. After filtering IRC session out of the traffic, flow-based methods were applied to discriminate malicious from benign IRC channels. The methods proposed by [20, 21] combined both application and network layer analysis. Cooke et al. [22] dealt with IRC activities at the application layer, using information coming from the monitoring of network activities. Some authors had introduced machine learning techniques into botnet detection [23], since they led a better way to characterize botnets. Currently, honeynets and Intrusion Detection System (IDS) are two major techniques to prevent their attacks. Honeynets can be deployed in both distributed and local context [9]. They are capable of providing botnet attacking information but cannot tell the details such as whether the victim has a certain worm [9]. The IDS uses the signatures or behavior of existing botnets for reference to detect potential attacks. Thus, to summarize the characteristics of botnets is significant for secure networks. To the best of our knowledge, we have not found any other work about anomaly-based detection for botnets. Before going to the discussion of botnet attacks and preventive measures, we will introduce some relevant terms and classification of bots in the rest of this section.

2.1. Formation and Exploitation. To illustrate the formation and exploitation, we take a spamming botnet as an example. A typical formation of botnet can be described by the following steps [3], as shown in Figure 1.

(1) The perpetrator of botnet sends out worms or viruses to infect victims' machines, whose payloads are bots.

(2) The bots on the infected hosts log into an IRC server or other communications medium, forming a botnet.

(3) Spammer makes payment to the owner of this botnet to gain the access right.

(4) Spammer sends commands to this botnet to order the bots to send out spam.

(5) The infected hosts send the spam messages to various mail servers in the Internet.

Botnets can be exploited for criminally purposes or just for fun, depending on the individuals. The next section will go into the details of various exploitations.

2.2. Botnet Lifecycle. Figure 2 shows the lifecycle of a botnet and a single bot [16].

2.3. IRC-Based Bot. IRC is a protocol for text-based instant messaging among people connected with the Internet. It is based on Client/Server (C/S) model but suited for distributed environment as well [18]. Typical IRC severs are interconnected and pass messages from one to another [18]. One can connect with hundreds of clients via multiple servers. It is so-called multiple IRC (mIRC), in which communications among clients and a server are pushed to those who are connected to the channel. The functions of IRC-based bots include managing access lists, moving files, sharing clients, sharing channel information, and so on [18]. Major parts of a typical IRC bot attack are showed in Figure 3 [18].

(i) *Bot* is typically an executable file triggered by a specific command from the IRC sever. Once a bot is installed on a victim host, it will make a copy into a configurable directory and let the malicious program to start with the operating system. Consider Windows as an instance, the bots sized no more than 15 kb are able to add into the system registry (HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVerssion\Run\) [18]. Generally, bots are just the payload of worms or the way to open a backdoor [18].

(ii) *Control channel* is a secured IRC channel set up by the attacker to manage all the bots.

(iii) *IRC Server* may be a compromised machine or even a legitimate provider for public service.

(iv) *Attacker* is the one who control the IRC bot attack.

The attacker's operations have four stages [16].

(1) *The first one is the Creation Stage*, where the attacker may add malicious code or just modify an existing one out of numerous highly configurable bots over the Internet [16].

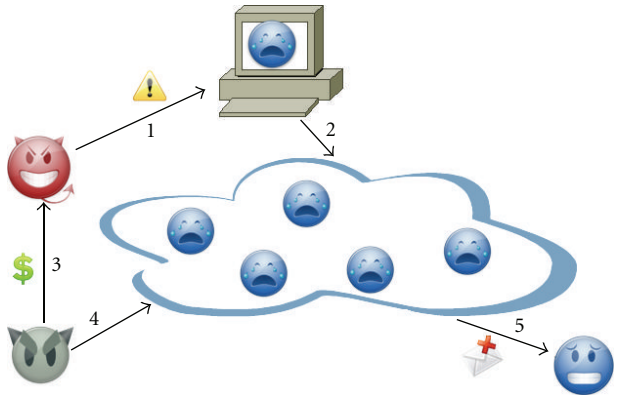

Figure 1: Using a botnet to send spam [3].

(2) *The second one is the Configuration Stage*, where the IRC server and channel information can be collected [16]. As long as the bot is installed on the victim, it will automatically connect to the selected host [16]. Then, the attacker may restrict the access and secure the channel to the bots for business or some other purpose [16]. For example, the attacker is able to provide a list of bots for authorized users who want to further customize and use them for their own purpose.

(3) *The third one is the Infection Stage*, where bots are propagated by various direct and indirect means [16]. As the name implies, direct techniques exploit vulnerabilities of the services or operating systems and are usually associated with the use of viruses [16]. While the vulnerable systems are compromised, they continue the infection process such that saving the time of attacker to add other victims [16]. The most vulnerable systems are Windows 2000 and XP SP1, where the attacker can easily find unpatched or unsecured (e.g., without firewall) hosts [16]. By contrary, indirect approaches use other programs as a proxy to spread bots, that is, using distributed malware through DCC (Direct Client-to-Client) file exchange on IRC or P2P networks to exploit the vulnerabilities of target machines [16].

(4) *The forth one is the Control Stage*, where the attacker can send the instructions to a group of bots via IRC channel to do some malicious tasks.

2.4. P2P-Based Bot. Few papers focus on P2P-based bots so far [4, 24–30]. It is still a challenging issue. In fact, using P2P ad hoc network to control victim hosts is not a novel technique [26]. A worm with a P2P fashion, named Slapper [27], infected Linux system by DoS attack in 2002. It used hypothetical clients to send commands to compromised hosts and receive responses from them [27]. Thereby, its network location could be anonymous and hardly be monitored [27]. One year after, another P2P-based bot appeared, called Dubbed Sinit [28]. It used public key cryptography for update authentication. Later, in 2004, Phatbot [29] was created to send commands to other compromised hosts using a P2P system. Currently, Storm Worm [24] may be the most wide-spread P2P bot over the Internet. Holz et al. have analyzed it using binary and network tracing [24]. Besides, they also proposed some techniques to disrupt the communication of a P2P-based botnet, such as eclipsing content and polluting the file.

Nevertheless, the above P2P-based bots are not mature and have many weaknesses. Many P2P networks have a central server or a seed list of peers who can be contacted for adding a new peer. This process named bootstrap has a single point of failure for a P2P-based botnet [25]. For this reason, authors in [25] presented a specific hybrid P2P botnet to overcome this problem.

Figure 4 presents the C2 architecture of the hybrid P2P-based botnet proposed by [25]. It has three client bots and five servant bots, who behave both as clients and servers in a traditional P2P file sharing system. The arrow represents a directed connection between bots. A group of servant bots interconnect with each other and form the backbone of the botnet. An attacker can inject his/her commands into any hosts of this botnet. Each host periodically connects to its neighbors for retrieving orders issued by their commander. As soon as a new command shows up, the host will forward this command to all nearby servant bots immediately. Such architecture combines the following features [25]: (1) it requires no bootstrap procedure; (2) only a limited number of bots nearby the captured one can be exposed; (3) an attacker can easily manage the entire botnet by issuing a single command. Albeit the authors in [25] proposed several countermeasures against this botnet attack, more researches on both architecture and prevention means are still needed in the future. The relevant future work will be discussed in Section 6.

2.5. Types of Bots. Many types of bots in the network have already been discovered and studied [9, 16, 17]. Table 1 will present several widespread and well-known bots, together with their basic features. Then, some typical types will be studied in details.

2.5.1. Agobot. This well-known bot is written in C/C++ with cross-platform capabilities [9]. It is the only bot so far that utilizes a control protocol in IRC channel [9]. Due to its standard data structures, modularity, and code documentation, Agobot is very easy for attacker to extend commands for their own purposes by simply adding new function into the CCommandHandler or CScanner class [9]. Besides, it has both standard and special IRC commands for harvesting sensitive information [17]. For example, it can request the bot to do some basic operations (accessing a file on the compromised machine by "bot.open" directive) [17]. Also, Agobot is capable of securing the system via closing NetBIOS shares, RPC-DCOM, for instance [17]. It has various commands to control the victim host, for example, using "pctrl" to manage all the processes and using "inst" to manage autostart programs [17]. In addition, it has the following features [17]: (1) it is IRC-based C2 framework,

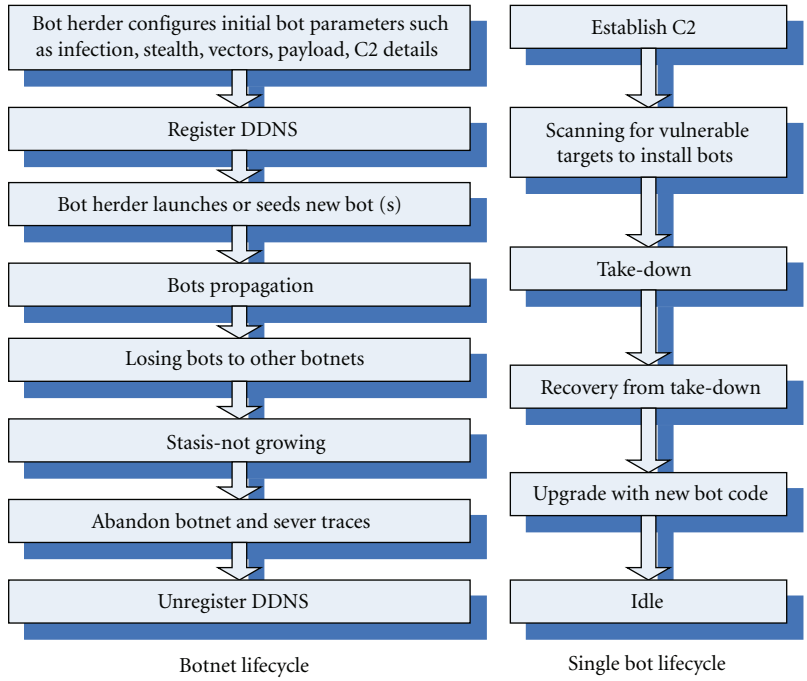

Figure 2: Lifecycle of a Botnet and of a single Bot [16].

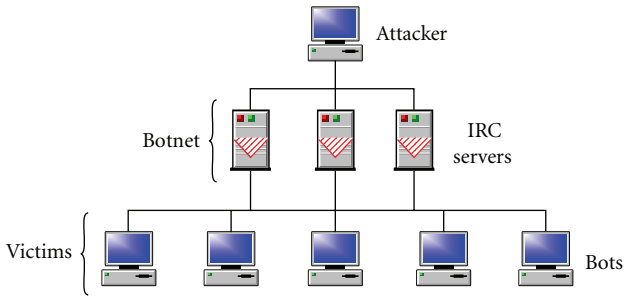

Figure 3: Major parts of a typical IRC Bot attack [18].

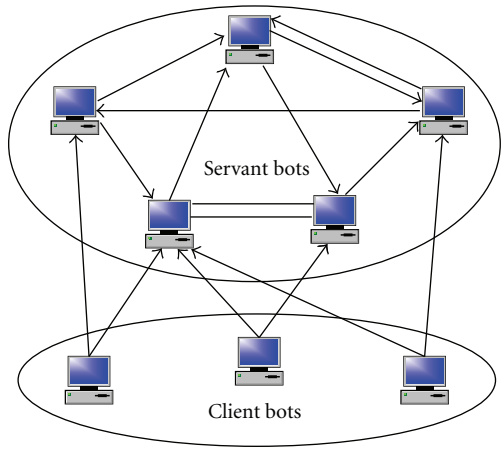

Figure 4: The C2 architecture of a hybrid P2P botnet proposed by [25].

(2) it can launch various DoS attacks, (3) it can attack a large number of targets, (4) it offers shell encoding function and limits polymorphic obfuscations, (5) it can harvest the sensitive information via traffic sniffing (using libpcap, a packet sniffing library [9]), key logging or searching registry entries, (6) it can evade detection of antivirus software either through patching vulnerabilities, closing back doors or disabling access to anti-virus sites (using NTFS Alternate Data Stream to hide its presence on victim host [9]), and (7) it can detect debuggers (e.g., SoftIce and Ollydbg) and virtual machines (e.g., VMware and Virtual PC) and thus avoid disassembly [9, 17].

To find a new victim, Agobot just simply scans across a predefined network range [17]. Nevertheless, it is unable to effectively distribute targets among a group of bots as a whole based on current command set [17].

2.5.2. SDBot. SDBot's source code is not well written in C and has no more than 2500 lines, but its command set and features are similar to Agobot [9, 17]. It is published under GPL [9, 17]. Albeit SDBot has no propagation capability and only provides some basic functions of host control, attackers still like this bot since its commands are easy to extend [17]. In addition, SDBot has its own IRC functions such as spying and cloning [17]. Spying is just recording the activities of a specified channel on a log file [17]. Cloning means that the bot repeats to connect one channel [17]. At present, SDBot may be the most active bot used in the wild [9]. There are plenty of auxiliary patches available on the Internet, including non-malicious ones [17].

Table 1: Types of bots.

Types	Features
Agobot Phatbot Forbot Xtrembot	They are so prevalent that over 500 variants exist in the Internet today. Agobot is the only bot that can use other control protocols besides IRC [9]. It offers various approaches to hide bots on the compromised hosts, including NTFS Alternate Data Stream, Polymorphic Encryptor Engine and Antivirus Killer [16].
SDBot RBot UrBot UrXBot	SDBot is the basis of the other three bots and probably many more [9]. Different from Agobot, its code is unclear and only has limited functions. Even so, this group of bots is still widely used in the Internet [16].
SpyBot NetBIOS Kuang Netdevil KaZaa	There are hundreds of variants of SpyBot nowadays [17]. Most of their C2 frameworks appear to be shared with or evolved from SDBot [17]. But it does not provide accountability or conceal their malicious purpose in codebase [17].
mIRC-based GT-Bots	GT (Global Threat) bot is mIRC-based bot. It enables a mIRC chat-client based on a set of binaries (mainly DLLs) and scripts [16]. It often hides the application window in compromised hosts to make mIRC invisible to the user [9].
DSNX Bots	The DSNX (Data Spy Network X) bot has a convenient plug-in interface for adding a new function [16]. Albeit the default version does not meet the requirement of spreaders, plugins can help to address this problem [9].
Q8 Bots	It is designed for Unix/Linux OS with the common features of a bot, such as dynamic HTTP updating, various DDoS-attacks, execution of arbitrary commands and so forth. [9].
Kaiten	It is quite similar to Q8 Bots due to the same runtime environment and lacking of spreader as well. Kaiten has an easy remote shell, thus it is convenient to check further vulnerabilities via IRC [9].
Perl-based bots	Many variants written in Perl nowadays [9]. They are so small that only have a few hundred lines of the bots code [9]. Thus, limited fundamental commands are available for attacks, especially for DDoS-attacks in Unix-based systems [9].

SDBot's is essentially a compact IRC implementation [17]. To contact the IRC server, it first sends identity information, for example, USER and NICK [17]. As long as it gets an admission message (PING) from the server, the bot will acknowledge this connection with a PONG response [17]. While the bot receives the success code (001 or 005) for connection, it can request a hostname by USERHOST and join the channel by JOIN message [17]. Once it receives a response code 302, this bot has successfully participated in the IRC channel and the master can control it via some IRC commands (e.g., NOTICE, PRIVMSG, or TOPIC) [17].

With the help of many powerful scanning tools, SDBot can easily find the next victim [17]. For instance, using NetBIOS scanner, it can randomly choose a target located in any predefined IP range [17]. Since the SDBot is able to send ICMP and UDP packets, it is always used for simple flooding attacks [17]. Moreover, a large number of variants capable of DDoS attack are available in the wild [17].

2.5.3. SpyBot. SpyBot is written in C with no more than 3,000 lines, and has pretty much variants nowadays as well [17]. As a matter of fact, SpyBot is enhanced version of SDBot [17]. Besides the essential command language implementation, it also involves the scanning capability, host control function, and the modules of DDoS attack and flooding attack (e.g., TCP SYN, ICMP, and UDP) [17]. SpyBot's host control capabilities are quite similar to Agobot's in remote command execution, process/system manipulation, key logging, and local file manipulation [17]. Nevertheless, SpyBot still does not have the capability breadth and modularity of Agobot [17].

2.5.4. GT Bot. GT (Global Threat) Bot, as known as Aristotles, is supposed to stand for all mIRC-based bots which have numerous variants and are widely used for Windows [9, 17]. Besides some general capabilities such as IRC host control, DoS attacks, port scanning, and NetBIOS/RPC exploiting, GT Bot also provides a limited set of binaries and scripts of mIRC [9, 17]. One important binary is *HideWindow* program used to keep the mIRC instance invisible from the user [9, 17]. Another function is recording the response to each command received by remote hosts [17]. Some other binaries mainly extend the functions of mIRC via DDL (Dynamic Link Library) [9]. These scripts often store in files with "*.mrc*" extension or in "*mirc.ini*" [9, 17]. Although the binaries are almost all named as "*mIRC.exe*", they may have different capabilities due to distinct configuration files [17]. Compared to the above instances, GT Bot only provides limited commands for host control, just capable of getting local system information and running or deleting local files [17].

3. Botnet Attacks

Botnets can serve both legitimate and illegitimate purposes [6]. One legitimate purpose is to support the operations of IRC channels using administrative privileges on specific individuals. Nevertheless, such goals do not meet the vast number of bots that we have seen. Based on the wealth of data logged in Honeypots [9], the possibilities to use botnets for criminally motivated or for destructive goals can be categorized as follows.

3.1. DDoS Attacks. Botnets are often used for DDoS attacks [9], which can disable the network services of victim system by consuming its bandwidth. For instance, a perpetrator may order the botnet to connect a victim's IRC channel at first, and then this target can be flooded by thousands of service requests from the botnet. In this kind of DDoS attack, the victim IRC network is taken down. Evidence reveals that most commonly implemented by botnets are TCP SYN and UDP flooding attacks [31].

General countermeasure against DDoS attacks requires: (1) controlling a large number of compromised machines; (2) disabling the remote control mechanism [31]. However, more efficient ways are still needed to avoid this kind of attack. Freiling et al. [31] have presented an approach to prevent DDoS attack via exploring the hiding bots in Honeypots.

3.2. Spamming and Spreading Malware. About 70% to 90% of the world's spam is caused by botnets nowadays, which has most experienced in the Internet security industry concerned [32, 33]. Study report indicates that, once the SOCKS v4/v5 proxy (TCP/IP RFC 1928) on compromised hosts is opened by some bots, those machines may be used for nefarious tasks, for example, spamming. Besides, some bots are able to gather email addresses by some particular functions [9]. Therefore, attackers can use such a botnet to send massive amounts of spam [34].

Researchers in [35] have proposed a distributed content independent spam classification system, called Trinity, against spamming from botnets. The designer assumes that the spamming bots will send a mass of e-mails within a short time. Hence, any letter from such address can be a spam. It is a little bit unexpected that we do not know the effectiveness of Trinity since it is still under experiment.

In order to discover the aggregate behaviors of spamming botnet and benefit its detection in the future, Xie et al. [36] have designed a spam signature generation framework named AutoRE. They also found several characteristics of spamming botnet: (1) spammer often appends some random and legitimate URLs into the letter to evade detection [36]; (2) botnet IP addresses are usually distributed over many ASes (Autonomous Systems), with only a few participating machines in each AS on average [36]; (3) despite that the contents of spam are different, their recipients' addresses may be similar [36]. How to use these features to capture the botnets and avoid spamming is worth to research in the future.

Similarly, botnets can be used to spread malware too [9]. For instance, a botnet can launch Witty worm to attack ICQ protocol since the victims' system may have not activated Internet Security Systems (ISS) services [9].

3.3. Information Leakage. Because some bots may sniff not only the traffic passing by the compromised machines but also the command data within the victims, perpetrators can retrieve sensitive information like usernames and passwords from botnets easily [9]. Evidences indicate that, botnets are becoming more sophisticated at quickly scanning in the host for significant corporate and financial data [32]. Since the bots rarely affect the performance of the running infected systems, they are often out of the surveillance area and hard to be caught. Keylogging is the very solution to the inner attack [9, 16]. Such kind of bots listens for keyboard activities and then reports to its master the useful information after filtering the meaningless inputs. This enables the attacker to steal thousands of private information and credential data [16].

3.4. Click Fraud. With the help of botnet, perpetrators are able to install advertisement add-ons and browser helper objects (BHOs) for business purpose [9]. Just like Google's AdSense program, for the sake of obtaining higher click-through rate (CTR), perpetrators may use botnets to periodically click on specific hyperlinks and thus promote the CTR artificially [9]. This is also effective to online polls or games [9]. Because each victim's host owns a unique IP address scattered across the globe, every single click will be regarded as a valid action from a legitimate person.

3.5. Identity Fraud. Identity Fraud, also called as Identity Theft, is a fast growing crime on the Internet [9]. Phishing mail is a typical case. It usually includes legitimate-like URLs and asks the receiver to submit personal or confidential information. Such mails can be generated and sent by botnets through spamming mechanisms [9]. In a further step, botnets also can set up several fake websites pretending to be an official business sites to harvest victims' information. Once a fake site is closed by its owner, another one can pop up, until you shut down the computer.

4. Detection and Tracing

By now, several different approaches of identifying and tracing back botnets have been proposed or attempted. First and the most generally, the use of Honeypots, where a subnet pretends to be compromised by a Trojan, but actually observing the behavior of attackers, enables the controlling hosts to be identified [22]. In a relevant case, Freiling et al. [31] have introduced a feasible way to detect certain types of DDoS attacks lunched by the botnet. To begin with, use honeypot and active responders to collect bot binaries. Then, pretend to join the botnet as a compromised machine by running bots on the honeypot and allowing them to access the IRC server. At the end, the botnet is infiltrated by a "silent drone" for information collecting, which may be useful

in botnet dismantling. Another and also commonly used method is using the information form insiders to track an IRC-based botnet [11]. The third but not the least prevalent approach to detect botnets is probing DNS caches on the network to resolve the IP addresses of the destination servers [11].

4.1. Honeypot and Honeynet. Honeypots are well-known by their strong ability to detect security threats, collect malwares, and to understand the behaviors and motivations of perpetrators. Honeynet, for monitoring a large-scale diverse network, consists of more than one honeypot on a network. Most of researchers focus on Linux-based honeynet, due to the obvious reason that, compared to any other platform, more freely honeynet tools are available on Linux [6]. As a result, only few tools support the honeypots deployment on Windows and intruders start to proactively dismantle the honeypot.

Some scholars aim at the design of a reactive firewall or related means to prevent multiple compromises of honeypots [6]. While a compromised port is detected by such a firewall, the inbound attacks on it can be blocked [6]. This operation should be carried on covertly to avoid raising suspicions of the attacker. Evidence shows that operating less covertly is needed on protection of honeypots against multiple compromises by worms, since worms are used to detect its presence [6]. Because many intruders download toolkits in a victim immediate aftermath, corresponding traffic should be blocked only selectively. Such toolkits are significant evidences for future analysis. Hence, to some extent, attackers' access to honeypots could not be prevented very well [6].

As honeypots have become more and more popular in monitoring and defense systems, intruders begin to seek a way to avoid honeypot traps [37]. There are some feasible techniques to detect honeypots. For instance, to detect VMware or other emulated virtual machines [38, 39], or, to detect the responses of program's faulty in honeypot [40]. In [41], Bethencourt et al. have successfully identified honeypots using intelligent probing according to public report statistics. In addition, Krawetz [42] have presented a commercial spamming tool capable of anti-honeypot function, called "Send-Safe's Honeypot Hunter." By checking the reply form remote proxy, spammer is able to detect honeypot open proxies [42]. However, this tool cannot effectively detect others except open proxy honeypot. Recently, Zou and Cunninqham [37] have proposed another methodology for honeypot detection based on independent software and hardware. In their paper, they also have introduced an approach to effectively locate and remove infected honeypots using a P2P structured botnet [37]. All of the above evidences indicate that, future research is needed in case that a botnet becomes invisible to honeypot.

4.2. IRC-based Detection. IRC-based botnet is wildly studied and therefore several characteristics have been discovered for detection so far. One of the easy ways to detect this kind of botnets is to sniff traffic on common IRC ports (TCP port 6667), and then check whether the payloads march the strings in the knowledge database [22]. Nevertheless, botnets can use random ports to communicate. Therefore, another approach looking for behavioral characteristics of bots comes up. Racine [43] found IRC-based bots were often idle and only responded upon receiving a specific instruction. Thus, the connections with such features can be marked as potential enemies. Nevertheless, it still has a high false positive rate in the result.

There are also other methodologies existing for IRC-based botnet detection. Barford and Yegneswaran [17] proposed some approaches based on the source code analysis. Rajab et al. [11] introduced a modified IRC client called IRC tracker, which was able to connect the IRC sever and reply the queries automatically. Given a template and relevant fingerprint, the IRC tracker could instantiate a new IRC session to the IRC server [11]. In case the bot master could find the real identity of the tracker, it appeared as a powerful and responsive bot on the Internet and run every malicious command, including the responses to the attacker [11]. We will introduce some detection methods against IRC-based botnets below.

4.2.1. Detection Based on Traffic Analysis. Signature technology is often used in anomaly detection. The basic idea is to extract feature information on the packets from the traffic and march the patterns registered in the knowledge base of existing bots. Apparently, it is easy to carry on by simply comparing every byte in the packet, but it also goes with several drawbacks [44]. Firstly, it is unable to identify the undefined bots [44]. Second, it should always update the knowledge base with new signatures, which enhances the management cost and reduces the performance [44]. Third, new bots may launch attacks before the knowledge base are patched [44].

Based on the features of IRC, some other techniques to detect botnets come up. Basically, two kinds of actions are involved in a normal IRC communication. One is interactive commands and another is messages exchanging [44]. If we can identify the IRC operation with a specified program, it is possible to detect a botnet attack [44]. For instance, if the private information is copied to other places by some IRC commands, we claim that the system is under an attack since a normal chatting behavior will never do that [44]. However, the shortcomings also exist. On the one hand, IRC port number may be changed by attackers. On the other hand, the traffic may be encrypted or be concealed by network noises [21]. Any situation will make the bots invisible.

In [44], authors observed the real traffic on IRC communication ports ranging from 6666 to 6669. They found some IRC clients repeated sending login information while the server refused their connections [44]. Based on the experiment result, they claimed that bots would repeat these actions at certain intervals after refused by the IRC server, and those time intervals are different [44]. However, they did not consider a real IRC-based botnet attack into their experiment. It is a possible future work to extend their achievements.

In [33], Sroufe et al. proposed a different method for botnet detection. Their approach can efficiently and automatically identify spam or bots. The main idea is to extract the shape of the Email (lines and the character count of each line) by applying a Gaussian kernel density estimator [33]. Emails with similar shape are suspected. However, authors did not show the way to detect botnet by using this method. It may be another future work worth to study.

4.2.2. Detection Based on Anomaly Activities. In [21], authors proposed an algorithm for anomaly-based botnet detection. It combined IRC mesh features with TCP-based anomaly detection module. It first observed and recorded a large number of TCP packets with respect to IRC hosts. Based on the ratio computed by the total amount of TCP control packets (e.g., SYN, SYNACK, FIN, and RESETS) over total number of TCP packets, it is able to detect some anomaly activities [21]. They called this ratio as the TCP work weight and claimed that high value implied a potential attack by a scanner or worm [21]. However, this mechanism may not work if the IRC commands have been encoded, as discussed in [21].

4.3. DNS Tracking. Since bots usually send DNS queries in order to access the C2 servers, if we can intercept their domain names, the botnet traffic is able to be captured by blacklisting the domain names [45, 46]. Actually, it also provides an important secondary avenue to take down botnets by disabling their propagation capability [11].

Choi et al. [45] have discussed the features of botnet DNS. According to their analysis, botnets' DNS queries can be easily distinguished from legitimate ones [45]. First of all, only bots will send DNS queries to the domain of C2 servers, a legitimate one never do this [45]. Secondly, botnet's members act and migrate together simultaneously, as well as their DNS queries [45]. Whereas the legitimate one occurs continuously, varying from botnet [45]. Third, legitimate hosts will not use DDNS very often while botnet usually use DDNS for C2 servers [45]. Based on the above features, they developed an algorithm to identify botnet DNS queries [45]. The main idea is to compute the similarity for group activities and then distinguish the botnet from them based on the similarity value. The similarity value is defined as 0.5 (C/A+C/B), where A and B stand for the sizes of two requested IP lists which have some common IP addresses and the same domain name, and C stands for the size of duplicated IP addresses [45]. If the value approximated zero, such common domain will be suspected [45].

There are also some other approaches. Dagon [46] presented a method of examining the query rates of DDNS domain. Abnormally high rates or temporally concentrated were suspected, since the attackers changed their C2 servers quite often [47]. They utilized both Mahalanobis distance and Chebyshev's inequality to quantify how anomalous the rate is [47]. Schonewille and van Helmond [48] found that when C2 servers had been taken down, DDNS would often response name error. Hosts who repeatedly did such queries could be infected and thus to be suspected [48]. In [47], authors evaluated the above two methods through experiments on the real world. They claimed that, Dagon's approach was not as effective since it misclassified some C2 server domains with short TTL, while Schonewille's method was comparatively effective due to the fact that the suspicious name came from independent individuals [47].

In [49], Hu et al. proposed a botnet detection system called RB-Seeker (Redirection Botnet Seeker). It is able to automatically detect botnets in any structure. RB-Seeker first gathers information about bots redirection activities (e.g., temporal and spatial features) from two subsystems. Then it utilizes the statistical methodology and DNS query probing technique to distinguish the malicious domain from legitimate ones. Experiment results show that RB-Seeker is an efficient tool to detect both "aggressive" and "stealthy" botnets.

5. Preventive Measures

It takes only a couple of hours for conventional worms to circle the globe since its release from a single host. If worms using botnet appear from multiple hosts simultaneously, they are able to infect the majority of vulnerable hosts worldwide in minutes [7]. Some botnets have been discussed in previous sections. Nevertheless, there are still plenty of them that are unknown to us. We also discuss a topic of how to minimize the risk caused by botnets in the future in this section.

5.1. Countermeasures on Botnet Attacks. Unfortunately, few solutions have been in existence for a host to against a botnet DoS attack so far [3]. Albeit it is hard to find the patterns of malicious hosts, network administrators can still identify botnet attacks based on passive operating system fingerprinting extracted from the latest firewall equipment [3]. The lifecycle of botnets tells us that bots often utilize free DNS hosting services to redirect a sub-domain to an inaccessible IP address. Thus, removing those services may take down such a botnet [3]. At present, many security companies focus on offerings to stop botnets [3]. Some of them protect consumers, whereas most others are designed for ISPs or enterprises [3]. The individual products try to identify bot behavior by anti-virus software. The enterprise products have no better solutions than nullrouting DNS entries or shutting down the IRC and other main servers after a botnet attack identified [3].

5.2. Countermeasures for Public. Personal or corporation security inevitably depends on the communication partners [7]. Building a good relationship with those partners is essential. Firstly, one should continuously request the service supplier for security packages, such as firewall, anti-virus tool-kit, intrusion detection utility, and so forth. [7]. Once something goes wrong, there should be a corresponding contact number to call [7]. Secondly, one should also pay much attention on network traffic and report it to ISP if there is a DDoS attack. ISP can help blocking those malicious IP addresses [7]. Thirdly, it is better to establish

Table 2: Rules of prevention by home users [18].

Type	Strategies
Personal Habits	Attention while downloading
	Avoid to install useless things
	Read carefully before you click
Routine	Use anti-virus/trojan utilities
	Update system frequently
	Shutdown PC when you leave
Optional Operations	Back-up all systems regularly
	Keep all software up-to-date
	Deploy personal firewall

accountability on its system, together with a law enforcement authority [7]. More specifically, scholars and industries have proposed some strategies for both home users and system administrators, to prevent, detect and respond botnet attacks [16, 18]. Here we summarize their suggestions.

5.2.1. Home Users. To prevent attacks from a botnet, home users can follow the rules described in Table 2. They are classified into three categories: (1) Personal Habits, (2) Routine, and (3) Optional Operations. As personal habits, people should pay attention when downloading, especially for those programs coming from unscrupulous sites. Besides, try to avoid installing useless things on personal computer, which will minimize the possibility of bots infection. If necessary, read the License Agreement and the notes carefully before click the button on the web site. As a routine, use anti-virus software and anti-trojan utilities while system is on. Scan and update system regularly, especially for Windows. When leaving the PC, shutdown the system or it may be remotely controlled by hackers. As the optional operations, home users are recommended to backup system regularly, to keep all software up-to-date and to deploy personal firewall by all means. By doing so, home PCs are shielded from unauthorized accesses, and thus bots cannot compromise them.

To detect an abnormal behavior, taking Windows operating system as an instance, a home user can check the IRC port range from 6000 to 7000 (typically 6667) by command "C:\Windows\netstat-an" [16, 18]. The result can reveal the connection of current IRC client. However, bots may use some other TCP ports [18]. If unusual behavior occurs on a home PC, such as slow network response, unknown ports being used, and something like that, there is possibly a bot attack [16, 18]. Also, home users can use anti-virus software or online services to detect attacks [16, 18]. Once the computer has been compromised, there are strategies to recover it. The following procedure (Figure 5) is a good example for home users.

5.2.2. System Administrator. Similarly, there are corresponding rules for system administrators to prevent, detect, and respond botnet attacks [16, 18]. For a prevention method, administrators should follow vendor guidelines for updating the system and applications [18]. Also, keep informed of latest vulnerabilities and use access control and log files to achieve accountability [18]. As illustrated in Table 3, the following measures can help the system administrator to minimize the possibilities of botnet attacking.

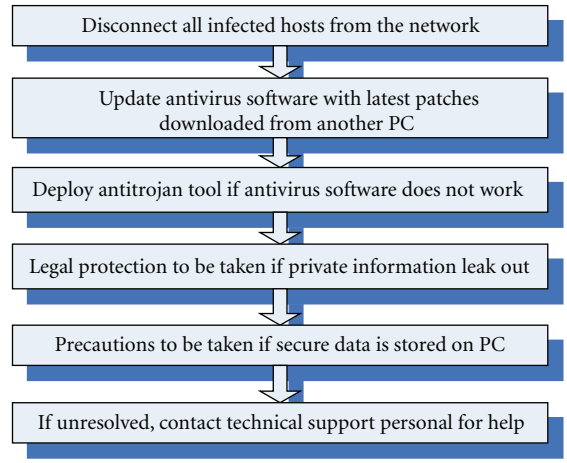

Figure 5: Home users' response to botnet attacks [18].

Table 3: Rules of detection by system administrators [18].

Rules	Notes
Monitor logs regularly	Analyze the internet traffic for anomalies
Use network packet sniffer	Identify the malicious traffic in intranet
Isolate the malicious subnet	Verify IRC activity on host
Scan individual machine	They may contain malware

Once an attack is detected, a system administrator should isolate those compromised hosts and notify the home users [16]. Then preserve the data on those infected hosts including the log files [16]. Besides, identify the number of victims via sniffer tools [16]. Finally, report the infection to security consultant [16].

6. Conclusion and Future Challenges

To better understand the botnet and stop its attack eventually, we provide a survey on existing research on botnets. The survey first discussed botnet formation and exploitation, the lifecycle, and two typical topologies. Aiming at the IRC-based and P2P-based botnet, we give a tutorial of current research on their attacks and countermeasures.

According to the discussion in Section 2, we propose several ideas on different topologies as follows. For IRC-based botnets, the thorny problem is that we cannot get the source code of most of bots. Hence, in-depth analysis at networking level and system level for bots' behaviors are hardly carried out. For P2P-based botnets, the following practical challenges should be further considered: (1) maintaining the rest of bots after some have been taken down by defenders; (2) hiding the botnet topology while some bots are captured by defenders; (3) managing the botnet more easily; (4) changing the traffic patterns more often and making it harder for detection.

Detecting and tracking compromised hosts in a botnet will continue to be a challenging task. Traffic fingerprinting is useful for identifying botnets. Nevertheless, just like previous signature technologies discussed in Section 3, its drawbacks are obvious. We need an up-to-date knowledge base for all released bots in the world, which seems to be an impossible mission. Anomaly detection is another feasible approach. However, when infected hosts do not behave as unusual, it may be unable to detect such a potential threat. Since current detecting technology depends on the happened attacking event, no guarantee for us to find every possible compromised hosts. One interesting issue about anomaly detection is the time efficiency. If an attack occurs and we can capture the anomaly in the first place and fix the relevant problems before it is used for malicious purposes, we say this anomaly detection is time efficient. We need to focus on its time efficiency in the future work.

In wireless context, especially for an ad hoc network, there is not much related research conducted on either attacking or defending. There are lots of open issues: (1) How to find the shortest route to attack a target; (2) How to prevent the compromised hosts from being detected in the wireless network; (3) How to propagate the bots in the wireless network, especially before some compromised hosts become off line.

There are also some other interesting open issues that need to be considered. To the best of our knowledge, DDoS attack derived from botnets cannot be avoided. Even if the attacking has been detected, there is no effective way to trace back or fight against it. Instead, one can only shut down the compromised hosts or disconnect with the network, waiting for further command such as scanning virus or reinstalling the operating system. As a matter of fact, what we need indeed is to avoid the propagation of bots in the first place. Perhaps the only effective approach to eliminate botnets is to deploy new protocols on routers worldwide. It is really a huge and beyond reality project. Then, why not consider installing them on a local gateway? If the gateway could block the communication of bots between several domains, the attacker could not easily manage the compromised hosts worldwide. In the meantime, the gateway could give us information about where the malicious command came from. Based on the available evidence over the network, it would be possible to trace back the initial attack source. Nevertheless, it is very difficult to implement such an idea due to the following reasons: (1) It is hard to distinguish the malicious packages from the regular traffic flow; (2) Cooperating among domains is not very easy, and sometimes even gateways can be compromised; (3) How to trace the potential attack and who should be noticed for further analysis need to be studied.

Acknowledgment

We would like to thank the editor and anonymous reviewers for their useful comments on earlier versions of this paper. This work was supported in part by the National Science Foundation (NSF) under grants CNS-0716211, CNS-0737325, and CCF-0829827.

References

[1] K. Ono, I. Kawaishi, and T. Kamon, "Trend of botnet activities," in *Proceedings of the 41st Annual IEEE Carnahan Conference on Security Technology (ICCST '07)*, pp. 243–249, Ottawa, Canada, October 2007.

[2] Wikipedia, "Internet bot," http://en.wikipedia.org/wiki/Internet_bot.

[3] Wikipedia, "Botnet," http://en.wikipedia.org/wiki/Botnet.

[4] B. Thuraisingham, "Data mining for security applications: mining concept-drifting data streams to detect peer to peer botnet traffic," in *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI '08)*, Taipei, Taiwan, June 2008.

[5] C. Mazzariello, "IRC traffic analysis for botnet detection," in *Proceedings of the 4th International Symposium on Information Assurance and Security (IAS '08)*, pp. 318–323, Napoli, Italy, September 2008.

[6] B. McCarty, "Botnets: big and bigger," *IEEE Security and Privacy*, vol. 1, no. 4, pp. 87–90, 2003.

[7] G. P. Schaffer, "Worms and viruses and botnets, oh my!: rational responses to emerging internet threats," *IEEE Security and Privacy*, vol. 4, no. 3, pp. 52–58, 2006.

[8] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP '02)*, pp. 312–321, Paris, France, November 2002.

[9] P. Bacher, T. Holz, M. Kotter, and G. Wicherski, "Know your Enemy: Tracking Botnets," http://www.honeynet.org/papers/bots.

[10] T. Holz, S. Marechal, and F. Raynal, "New threats and attacks on the world wide web," *IEEE Security and Privacy*, vol. 4, no. 2, pp. 72–75, 2006.

[11] 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 Internet Measurement Conference (IMC '06)*, pp. 41–52, Rio de Janeriro, Brazil, October 2006.

[12] E. Levy, "The making of a spam zombie army: dissecting the sobig worms," *IEEE Security and Privacy*, vol. 1, no. 4, pp. 58–59, 2003.

[13] D. Cook, J. Hartnett, K. Manderson, and J. Scanlan, "Catching spam before it arrives: domain specific dynamic blacklists," in *Proceedings of the Australasian Workshops on Grid Computing and E-Research*, pp. 193–202, Hobart, Australia, January 2006.

[14] J. Jung and E. Sit, "An empirical study of spam traffic and the use of DNS black lists," in *Proceedings of the 4th ACM SIGCOMM Internet Measurement Conference (IMC '04)*, pp. 370–378, Taormina, Italy, October 2004.

[15] A. Ramachandran, N. Feamster, and D. Dagon, "Revealing botnet membership using DNSBL counter-intelligence," in *Proceedings of the 2nd Conference on Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI '06)*, vol. 2, p. 8, San Jose, Calif, USA, 2006.

[16] J. Govil, "Examining the criminology of bot zoo," in *Proceedings of the 6th International Conference on Information, Communications and Signal Processing (ICICS '07)*, pp. 1–6, Singapore, December 2007.

[17] P. Barford and V. Yegneswaran, "An inside look at botnets," in *Proceedings of the ARO-DHS Special Workshop on Malware Detection*, Advances in Information Security, Springer, 2006.

[18] R. Puri, "Bots and botnets: an overview," Tech. Rep., SANS Institute, 2003.

[19] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting botnets with tight command and control," in *Proceedings of the 31st Annual IEEE Conference on Local Computer Networks (LCN '06)*, pp. 195–202, Tampa, Fla, USA, November 2006.

[20] M. Akiyama, T. Kawamoto, M. Shimamura, T. Yokoyama, Y. Kadobayashi, and S. Yamaguchi, "A proposal of metrics for botnet detection based on its cooperative behavior," in *Proceedings of the International Symposium on Applications and the Internet Workshops*, p. 82, Washington, DC, USA, January 2007.

[21] J. R. Binkley and S. Singh, "An algorithm for anomaly-based botnet detection," in *Proceedings of the 2nd Conference on Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI '06)*, p. 7, San Jose, Calif, USA, 2006.

[22] E. Cooke, F. Jahanian, and D. Mcpherson, "The zombie roundup: understanding, detecting, and disrupting botnets," in *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI '05)*, p. 6, Cambridge, Mass, USA, 2005.

[23] 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.

[24] T. Holz, M. Steiner, F. Dahl, E. W. Biersack, and F. Freiling, "Measurement and mitigation of peer-to-peer-based botnets: a case study on storm worm," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pp. 1–9, San Francisco, Calif, USA, April 2008.

[25] P. Wang, S. Sparks, and C. C. Zou, "An advanced hybrid peer-to-peer botnet," in *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets*, p. 2, Cambridge, Mass, USA, July 2008.

[26] R. Lemos, "Bot software looks to improve peerage," http://www.securityfocus.com/news/11390.

[27] I. Arce and E. levy, "An analysis of the slapper worm," *IEEE Security & Privacy Magazine*, vol. 1, no. 1, pp. 82–87, 2003.

[28] J. Stewart, "Sinit P2P Trojan analysis," http://www.secureworks.com/research/threats/sinit/.

[29] J. Stewart, "Phatbot Trojan analysis," http://www.secureworks.com/research/threats/phatbot/?threat=phatbot.

[30] C. Langin, H. Zhou, and S. Rahimi, "A model to use denied Internet traffic to indirectly discover internal network security problems," in *Proceedings of the IEEE International Performance, Computing, and Communications Conference (IPCCC '08)*, pp. 486–490, Austin, Tex, USA, December 2008.

[31] F. C. Freiling, T. Holz, and G. Wicherski, "Botnet tracking: exploring a root-cause methodology to prevent distributed denial-of-service attacks," in *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS '05)*, vol. 3679 of *Lecture Notes in Computer Science*, pp. 319–335, Springer, Milan, Italy, September 2005.

[32] K. Pappas, "Back to basics to fight botnets," *Communications News*, vol. 45, no. 5, p. 12, 2008.

[33] P. Sroufe, S. Phithakkitnukoon, R. Dantu, and J. Cangussu, "Email shape analysis for spam botnet detection," in *Proceedings of the 6th IEEE Consumer Communications and Networking Conference (CCNC '09)*, pp. 1–2, Las Vegas, Nev, USA, January 2009.

[34] K. Chiang and L. Lloyd, "A case study of the restock rootkit and spam bot," in *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets*, p. 10, Cambridge, Mass, USA, 2007.

[35] A. Brodsky and D. Brodsky, "A distributed content independent method for spam detection," in *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets*, p. 3, Cambridge, Mass, USA, 2007.

[36] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming botnets: signatures and characteristics," in *Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM '08)*, vol. 38, pp. 171–182, Seattle, Wash, USA, August 2008.

[37] C. C. Zou and R. Cunningham, "Honeypot-aware advanced botnet construction and maintenance," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN '06)*, pp. 199–208, Philadelphia, Pa, USA, June 2006.

[38] J. Corey, "Advanced honey pot identification and exploitation," 2004, http://www.ouah.org/p63-0x09.txt.

[39] K. Seifried, "Honeypotting with VMware basics," 2002, http://www.seifried.org/security/index.html.

[40] Honeyd security advisory 2004–001, "Remote detection via simple probe packet," 2004, http://www.honeyd.org/adv.2004-01.asc.

[41] J. Bethencourt, J. Franklin, and M. Vernon, "Mapping internet sensors with probe response attacks," in *Proceedings of the 14th Conference on USENIX Security Symposium*, pp. 193–208, Baltimore, Md, USA, August 2005.

[42] N. Krawetz, "Anti-Honeypot technology," *IEEE Security and Privacy*, vol. 2, no. 1, pp. 76–79, 2004.

[43] S. Racine, *Analysis of internet relay chat usage by DDoS zombies*, M.S. thesis, Swiss Federal Institute of Technology, Zurich, Switzerland, April 2004.

[44] Y. Kugisaki, Y. Kasahara, Y. Hori, and K. Sakurai, "Bot detection based on traffic analysis," in *Proceedings of the International Conference on Intelligent Pervasive Computing (IPC '07)*, pp. 303–306, Jeju Island, South Korea, October 2007.

[45] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in DNS traffic," in *Proceedings of the 7th IEEE International Conference on Computer and Information Technology (CIT '07)*, pp. 715–720, Fukushima, Japan, October 2007.

[46] D. Dagon, "Botnet detection and response, the network is the infection," 2005, http://www.caida.org/workshops/dns-oarc/200507/slides/oarc0507-Dagon.pdf.

[47] R. Villamarin-Salomon and J. C. Brustoloni, "Identifying botnets using anomaly detection techniques applied to DNS traffic," in *Proceedings of the 5th IEEE Consumer Communications and Networking Conference*, pp. 476–481, Las Vegas, Nev, USA, January 2008.

[48] A. Schonewille and D. J. van Helmond, *The domain name service as an IDS*, M.S. thesis, University of Amsterdam, Amsterdam, The Netherlands, February 2006.

[49] X. Hu, M. Knyz, and K. G. Shin, "RB-Seeker: auto-detection of redirection botnets," in *Proceedings of 16th Annual Network & Distributed System Security Symposium (NDSS '09)*, February 2009.