Computer Networks 57 (2013) 378–403

Contents lists available at SciVerse ScienceDirect

Computer Networks

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

Botnets: A survey

Sérgio S.C. Silva [a,*], Rodrigo M.P. Silva [b], Raquel C.G. Pinto [b], Ronaldo M. Salles [a,b]

[a] Defense Engineering Program, Military Institute of Engineering, Praça Gen Tiburcio 80, Rio de Janeiro, 22290-270, Brazil
[b] Computar Science Program, Military Institute of Engineering, Praça Gen Tiburcio 80, Rio de Janeiro, 22290-270, Brazil

ARTICLE INFO

Article history:
Received 20 December 2011
Received in revised form 20 July 2012
Accepted 21 July 2012
Available online 16 October 2012

Keywords:
Botnet
Bot
Malware
DDoS
Botnet detection
Network security

ABSTRACT

Botnets, which are networks formed by malware-compromised machines, have become a serious threat to the Internet. Such networks have been created to conduct large-scale illegal activities, even jeopardizing the operation of private and public services in several countries around the world. Although research on the topic of botnets is relatively new, it has been the subject of increasing interest in recent years and has spawned a growing number of publications. However, existing studies remain somewhat limited in scope and do not generally include recent research and developments. This paper presents a comprehensive review that broadly discusses the botnet problem, briefly summarizes the previously published studies and supplements these with a wide ranging discussion of recent works and solution proposals spanning the entire botnet research field. This paper also presents and discusses a list of the prominent and persistent research problems that remain open.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Networks that are formed by machines compromised by malware, also called botnets, have become a serious threat to the Internet [1–3]. Experts believe that approximately 16–25% of the computers connected to the Internet are members of botnets [4,5]. Such networks are prepared to conduct illegal activities on a scale large enough to jeopardize the operation of private and public services in several countries around the world. The attacker, also called the botmaster or botherder, takes control of the machines (the bots) to perpetrate various criminal activities, such as information and identity theft, denial of service attacks, unsolicited messaging and other activities [6,7].

Statistics generated by network security companies have called attention to the severity of the botnet problem [8,9]. Some reports indicate that approximately 80% of all email traffic is spam and that most of these messages are sent via botnets, such as the so called Grum, Cutwail and Rustock botnets [10]. Although these unwanted email messages may be filtered at their destinations, all of these messages are generally allowed to travel along the backbones of the Internet, burdening traffic and wasting network resources.

Most importantly, botnet operation can be lucrative for its controllers because bots are inexpensive and relatively easy to propagate. For example, Symantec [8] reported an advertisement on an underground forum in 2010 promoting a botnet of 10,000 bots for US$ 15. This botnet may be used in a spam or rogueware campaign, but could also be used for a distributed denial of service attack.

Botnet activity is responsible for huge financial losses to Internet Service Providers, private companies, governments and home users. According to [11], ITU (International Telecommunication Union) estimates that the financial effects of malware range from US $13.2 billion, in direct damages caused to the global economy in 2006, to US$ 67.2 billion, in direct and indirect damages to US businesses in 2005. The global cost of spam in 2007 has been estimated to have been US$ 100 billion, with US$ 35 billion allocated to the US alone.

* Corresponding author. Tel.: +55 2125467090; fax: +55 2125467092.
E-mail addresses: scardoso@ime.eb.br (S.S.C. Silva), rmpraxedes@yahoo.com.br (R.M.P. Silva), raquel@ime.eb.br (R.C.G. Pinto), salles@ieee.org (R.M. Salles).

1389-1286/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2012.07.021

Most research on botnets is relatively recent and the interest in the topic is growing in the research community. A search of the major scientific databases IEEE, ACM and Elsevier reveals the following about the number of scientific articles devoted exclusively to this topic (see Fig. 1).

There are several surveys on botnets in the literature, but most [12–17] date from before 2009 and therefore cannot cover recent publications in the field. The surveys in [12,13] clarify certain issues regarding the botnet phenomenon and cover the bot life cycle and botnet detection techniques. Zhu et al. [12] also presents a taxonomy for bot types, metrics for determining botnet size and enterprise solution countermeasures against spam, while [13] classifies command and control architectures as centralized or decentralized and further breaks down botnet detection techniques into signature-based, anomaly-based, DNS-based, and mining-based.

The survey in [14] offers a brief look at certain aspects of botnet research, including the evolution and future of botnets, propagation speed and mechanisms, design complexity, detectability and population size. This survey also classifies command and control channels as centralized, decentralized or unstructured, then sorts research on botnets into two broad categories, botnet detection techniques and botnet measurement studies.

Shin and Im [15] provides a short explanation about botnet topologies and activities. Although it offers an interesting discussion about the consequences, defenses and challenges regarding botnets and DDoS, this paper is somewhat limited in scope and does not make a comprehensive review of the related literature.

A short survey on botnets is also presented in [16], where the authors show botnet evolution through a timeline of different malware used from 1993 to 2007. They classify botnet research into the categories of papers addressing infection mechanisms, malicious behavior, command and control models, communication protocols, botnet detection and defense against botnets. The paper also presents a case study of the SpyBot Worm [18], based on its main features: infection, command and control architecture, exploits and attack mechanism.

The survey in [17] addresses botnet formation and exploitation, the bot life cycle, and describes IRC-based and P2P-based bots. It also describes characteristics of certain well-known bots, such as Agobot [19], SDBot [20], SpyBot [18] and GT Bot [21], and discusses malicious activities carried out by botnets. It discusses the detection of botnets only as it relates to honeynets, IRC and DNS.

The technical report presented in [22] is most likely the most comprehensive survey to date. Its authors discuss the botnet problem with respect to measurement and detection techniques, countermeasures, and recommendations for good practice and then briefly describe likely future trends. Despite the undeniable quality of the report, it focuses on the operational issues of existing systems and networks.

A common observation in the previous research is that botnets are moving targets, i.e., all aspects of a botnet's life cycle may change and evolve over time. No detection or mitigation technique will be permanently effective, and different types of actors (e.g., governments, enterprise networks, ISPs) will approach botnet problems with different goals and from different perspectives.

To the best of our knowledge, previous research has not yet been clearly summarized in an article that also augments such research. Moreover, a deeper and wider study is needed to account for recent algorithms developed for detection, analysis and mitigation of botnet activities. With this in mind, this paper offers the following contributions.

1. A presentation of a comprehensive tutorial-like study discussing the botnet problem in general.
2. A summary of previously published surveys supplemented by a review of a wide scope of recent literature and solution proposals spanning the entire botnet research field.
3. A brief discussion of prominent and persistent open research issues.

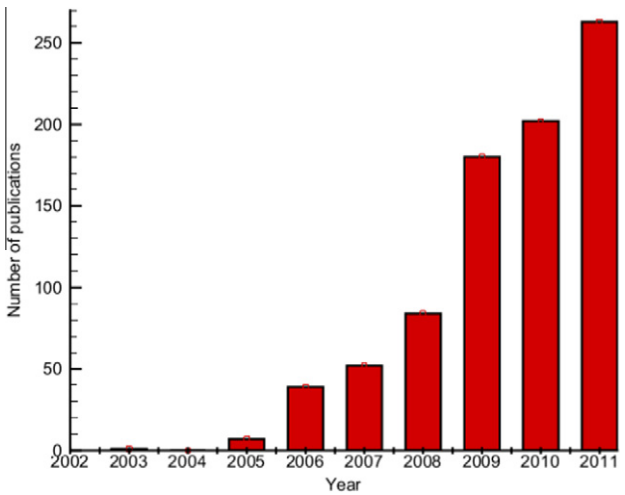

Fig. 1. Number of publications on botnets per year.

2. Botnet scrutiny

2.1. Definition and potential damage

Botnets are networks formed by "enslaving" host computers, called bots (derived from the word robot), that are controlled by one or more attackers, called botmasters, with the intention of performing malicious activities [6,7]. In other words, bots are malicious codes running on host computers that allow botmasters to control the host computers remotely and make them perform various actions [2,13,23–25]. The primary purpose of botnets is for the controlling criminal, group of criminals or organized crime syndicate to use hijacked computers for fraudulent online activity.

The coupling of the growing number of machines connected to the Internet through full-time broadband links with substantial system vulnerabilities[1] has created a perfect environment for the dissemination, infection and formation of botnets. Several works in the literature are dedicated simply to estimating the real size of botnets [1,26–29].

Although the sizes of botnets remain under investigation, reports say that Rustock, the largest known botnet in 2010, had well over 1 million bots under its control [8]. After monitoring approximately 180 botnets over a five-month period, the work in [30] tracked more than 300,000 unique IP addresses associated with at least one of the monitored botnet channels. The authors concluded that more than one million machines were compromised and controlled remotely by botmasters.

According to Satniford et al. [26], if a determined attacker or criminal organization controls one million machines, there may be no effective and complete defense against an offensive launched from such a botnet. Such an attack has the potential to cause immense damage, to assume a significant role in war scenarios between nations and to be a weapon of terrorism. It is no wonder that botnets are regarded as one of the biggest threats in cyber space [3,13,22] and that enormous effort and cooperation is needed to fight this threat.

Botnets can be utilized for a variety of disruptive activities, including spamming, performing DDoS attacks, distribution of malicious software (Trojan horses, spyware, keyloggers), software piracy, information harvesting, extortion, identity theft and manipulating online games or surveys, to name a few [6,7,13,22,30–33].

The growing number of botnets makes them extremely attractive and effective in orchestrating destructive DDoS attacks [34,35]. The sum of the transmission rates of all nodes of a botnet overwhelms most targets, making this type of attack extremely effective [1,30]. Actions taken against malicious activities, especially DDoS attacks and spamming, can be either reactive or preventive.

Reactive defenses are the most common, according to Freiling et al. [30]. The basic strategy is to first detect the malicious activity and then react to the attack by trying to reduce malicious traffic to acceptable levels. This approach has two primary disadvantages. First, it requires building a complete infrastructure with substantial computing power and data storage to analyze (preferably in real time) the large amount of monitoring information. The second problem relates to timing; because the attack is already underway by the time it is detected, legitimate users and ISPs have already suffered at least some of its effects.

Preventive techniques, on the other hand, try to avert the possibility of carrying out the malicious activities, especially in the cases of DDoS attacks and spamming, or to help the victim survive the attack by, for example, increasing its victims' resources or modifying the network infrastructure by forcing users to be authenticated. However, this approach may lead to an "arms race" because the attacker can also enhance its tools and resources. In addition, changing network infrastructures may require substantial financial resources.

It is necessary to analyze the root of the problem to make a preventive mechanism truly effective, i.e., the determination of what permits the realization of an attack or the implementation of illicit activities [1,35]. This methodology detects the machines involved in the preparation of an attack, culminating in their deactivation.

In recent years, an increasing number of studies have been carried out to learn to detect botnets and paralyze them [13–17,22,34,36,37]. The threats posed by botnets are just beginning to appear. Internet community researchers, law enforcement authorities, individual users and businesses have just begun to discuss methods of combating botnets, which represent perhaps the single greatest security threat to the Internet community today [1–3].

2.2. Botnets history

Historically, botnets originated from Internet Relay Chat (IRC), a text-based chat-system that organizes communication in channels, and the concept of bots did not necessarily include harmful behavior. The main idea behind botnets was to control interactions in Internet Relay Chat (IRC) chat rooms. They were able to interpret simple commands, provide administration support, offer simple games and other services to chat users, and retrieve information about operating systems, logins, email addresses, aliases, etc.

The first known IRC bot, Eggdrop, was published in 1993 [38] and developed further thereafter. Next, malicious IRC bots appeared that were developed by adopting the same basic idea, but were created for the primary purpose of attacking other IRC users or even entire servers. Shortly after, denial of service (DoS) and then distributed denial of service (DDoS) attacks were implemented in these bots.

New bots evolved that used complex mechanisms for communication with the botmaster, that exploited other available protocols and that integrated new, powerful methods of attack, all of which made the bots sophisticated and robust. They could propagate like worms, remain hidden as viruses and could launch large, coordinated attacks. Some examples of these bots are AgoBot [19] and SDBot [20]. The development of AgoBot and its variants is consid-

[1] In 2010, 6253 new vulnerabilities were cataloged according to [8].

ered as the point from which botnets became a major threat to the Internet [39].

The current generation of bots can spread through file-sharing networks, peer-to-peer (P2P) networks, email attachments and infected websites, or may be previously installed in backdoors. Communication between bots can be accomplished by several protocols, such as IRC, HTTP and P2P.

Table 1 presents a timeline of some important bots and some of their main features.

2.3. Components of a botnet

For a better understanding of how a botnet operates, we will next present its basic elements. Although there are botnets with different structures, the components illustrated in Fig. 2 are similar [23,39].

A bot is a software program (malware) installed in a vulnerable host that is capable of performing a series of actions, normally malicious. Such software can be installed on victim machines in many ways, including viral mechanisms or by accessing infected sites. Bots are typically configured so that each time the victim boots their computer, the bot is initialized.

Actions are initiated from specific commands sent from the botmaster through a command and control channel.

Importantly, bots are not vulnerabilities in operating systems or applications, but programs that are spread by worms or that are used to install backdoors on compromised machines. What distinguishes a bot from other

Table 1
Botnets timeline.

Year	Name	Architecture/protocol	Estimated size	Comments	Refs.
1993	EggDrop	Centralized/IRC	–	Recognized as one of the first popular IRC bots	[38]
1998	GTbot	Centralized	–	IRC-based bot that uses mIRC scripts	[7,21,40]
2002	SDbot	Centralized/IRC	–	Uses its own IRC client for better efficiency. Can also use instant-messaging programs and has reached more than 4000 variants	[7,20,40,41]
	Agobot	Centralized/IRC	–	Robust, modular, flexible and uses a persistent C&C channel	[7,19,42]
2003	Spybot	Centralized	–	Derived from Agobot with more features	[7,16,18]
	Sinit	P2P	–	Use random scanning to find others peers	[41,43]
2004	Bagle	Centralized	230,000		[41,44]
	Forbot	Centralized	–	Derived from Agobot	[45]
	Phatbot	P2P	–	Based on the WASTE P2P network	[7,41]
2006	SpamThru	P2P	12,000	SpamThru uses a custom P2P protocol to share information with other peers	[46]
	Nugache	P2P	160,000	Connect to a predefined list of peers	[47,48]
	Jrbot	Centralized	–	IRC-Based bot with a persistent channel	[7]
	Rxbot	Centralized/IRC	–	IRC-Based bot with a persistent channel	[49]
	Rustock	Centralized/HTTP	150,000	Bot responsible for 30 billion messagesper day, the largest botnet observed in 2010. Was deactivated in 2011.	[46,50,51]
2007	Storm	Centralized	160,000	Was considered one of most powerful botnets, with high processing power, capable of disconnecting entire countries	[46,48,52,53]
	Peacomm	P2P	160,000	Storm variant based on Kademlia network	[48,52,53]
	Pushdo	Centralized/HTTP	175,000	Encrypts C&C messages and capable of sending 4.500 messages in an hour per bot	[50,54]
	Srizbi	Centralized/HTTP	400,000	In 2008, it was one of the main botnets responsible for sending spam, approximately 50% of all traffic, approximately 80 to 60 billion messages per day	[46,55]
	Zeus/Zbot	Centralized/HTTP	3,6 millions	Allows the creation of new bots, with more than 3000 variants	[56–58]
	Mega-D	P2P	500,000	Became responsible for 1/3 of all spam traffic, was shut down in 2008	[50,59]
2008	Lethic	Centralized	260,000	Initially discovered in 2008, mainly involved in pharmaceutical and replica spam, was responsible for 8–10% of all the spam sent worldwide.	[44]
	Asprox	Centralized/HTTP	15,000	In addition to sending spam, it is able to perform SQL Injection on legitimate websites	[60]
	Bobax	Centralized/HTTP/UDP	185,000	Employs dynamic DNS and an algorithm for generating domains	[41,46]
	Kraken	Centralized	400,000	A variant of Bobax	[61,62]
	Torpig	Centralized	180,000	Typically targets bank account, credit-card data and also steals a variety ofother personal information	[63]
	Conficker	P2P	10,5 millions	In 2009, a coalition of security researchers was created to study Conficker, although some researchers do not consider it a bot/botnet	[64,65]
2009	Waledac	P2P	80,000	Successor of the Storm bot, was used for sending spam (7000 posts per day), shut off in 2010	[50,66,67]
	Donbot	Centralized/TCP	125,000	It uses a specific protocol for the C&C server using TCP ports above 2200	[50]
2010	Festi	Centralized/HTTP	–	Sends an HTTP request message to the C&C, which responds with encrypted templates of spam and/or a list of addresses	[44]
2011	TDL-4	P2P	4,5 millions	Has infected up to 4.5 million PCs in 2011, identified as one of the most sophisticated threats today. "It is virtually indestructible", according to security researchers	[68]

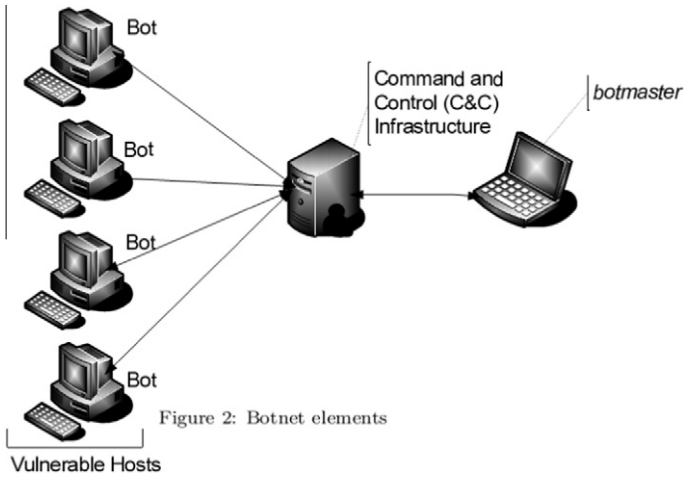

Fig. 2. Botnet elements.

types of malware is the command and control channel [1,32,69].

A botnet is a collection of bots connected to a command and control channel, i.e., a network of bots that is waiting for a command to perform malicious activities.

Botmasters and botherders are malicious users who control botnets by issuing commands to bots to perform illegal activities. Botmasters may obtain financial advantages by renting the network for other users to send spam, for example [1,8,70,71].

Vulnerable hosts are machines in the Internet that have been compromised with malicious software disseminated by a botmaster through a propagation mechanism. After being infected, these machines becomes ‘‘zombies’’ or ‘‘slaves’’ and may be used as attacking platforms against other vulnerable hosts or to carry out denial of service attacks.

The most critical component of a botnet is the so-called command-and-control infrastructure (C&C), consisting of the bots and a control entity that can be either centralized or decentralized. One or more communication protocols are used by the botmaster(s) to command slave computers and coordinate their actions. The instruction sets and functionality of botnets vary widely with the motivation behind their use.

The C&C infrastructure typically serves as the only way to control bots within the botnet and is necessary for maintaining a stable connection within this infrastructure to operate efficiently. Therefore, the architecture of the C&C infrastructure determines the robustness, stability and reaction time [14] of a botnet. In general, botnets may be classified as centralized or decentralized.

2.4. Desirable characteristics of a bot

Bots are applications that propagate themselves, as did previous generations of viruses and worms, to infect vulnerable hosts. According to Rajab et al. [25], bots are outfitted with several modern types of dissemination vectors to increase their numbers.

Botmasters mainly look for victims that have favorable features, such as high transmission rates, easy availability, low levels of security, low monitoring rates, and distant locations [23].

Given that one possible activity of compromised machines is to engage in a denial of service attack, bots should ideally be installed on machines that have high-speed Internet access so that the aggregate bandwidth available to the botmaster will be high enough to enable the paralysis of practically any service available in the Internet [1]. In addition, a bot's use of a vulnerable host with high-speed access allows malicious activities to be carried out almost unnoticed. If the bot uses just a small portion of the nominal network access, legitimate users do not notice suspicious activity and do not attempt to uninstall the malware [28].

According to Dagon et al. [72], botnets have diurnal behavior because computers are usually turned off at night. Moreover, certain activities may have regional characteristics that correspond to different time zones.

Geographical distance between botmaster and bots makes it difficult for law enforcement to track the activities of either. The distances among machines taking part in a typical botnet span different nations and different autonomous systems, making it difficult for the agencies responsible for monitoring such traffic to counteract it [13,25,73]. This feature supports the idea that a central cross-border monitor is necessary to verify traffic exchanged between nations and/or autonomous systems to identify command and control channels in all main networks [1,26].

Finally, most of the malware used by botnets runs only on MS Windows operating systems, making MS Windows machines the main targets [30]. However, recently malware has targeted a large variety of operating systems and platforms.

2.5. Life cycle of botnets

In order for an infected host to become an active bot and part of a botnet, the host must go through a cycle of

phases that integrate what is known as described in Refs. [12,13,33]. Such phases are sometimes identified by different names, but in general, the botnet life cycle is described as in Fig. 3.

The first phase is the Initial Injection, wherein a host is infected and becomes a potential bot. This phase is characterized by a regular computer infection procedure, which may be carried out in different ways as a typical virus infection would be, for instance, through unwanted downloads of malware from websites, infected files attached to email messages, infected removable disks, etc. [12,13,25,39,74].

The second phase, the secondary injection, requires that the first phase be successfully completed. In this phase, the infected host runs a program that searches for malware binaries in a given network database. When downloaded and executed, these binaries make the host behave as a real bot (or zombie). Downloading bot binaries is usually performed by FTP, HTTP or P2P protocols [12,13,25,74].

Although bot life cycles may vary under different implementations, at some point in time the new botnet client (bot) must contact a command and control (C&C) server to receive instructions or updates. Rallying [33] is known as the process of establishing a connection with the C&C; some authors call this procedure the connection phase [13]. In fact, this phase is scheduled every time the host is restarted to ensure the botmaster that the bot is taking part in the botnet and is able to receive commands to perform malicious activities. Therefore, the connection phase is likely to occur several times during the bot life cycle [45].

Because they must contact C&C servers, bots may be vulnerable during this phase. Bots often establish connections with C&C servers by default, allowing mechanisms to be created to identify traffic patterns and hence identify the components of the botnet or even of the command and control server.

The secondary injection and connection phases are related and, according to some authors, may be grouped into a single phase. For example, if the C&C server is also the repository of binaries, the two phases will most likely occur simultaneously. Whether these phases should be grouped together or not, they conduct similar activities, such as contacting servers.

To find the victim's binary repository or the C&C server, the malware installed during the initial infection (or the bot itself) should contain the address of the machines. These addresses may be encoded directly as a list of static IP addresses (hard-coded IP) or through the use of a list of domain names, which can be static or dynamic, making it harder to disable the command and control channel. While this makes it more difficult to take down or block a specific C&C server, the use of only a static domain name constitutes a single point of failure.

The more advanced techniques employ dynamic domain names as described in Ref. [63]. Several recent botnets, including Torpig, use domain flux instead, in which each bot independently uses a domain generation algorithm (DGA) to compute a list of domain names. It is important, however, that the algorithm used for generating names be very efficient, or else name patterns may be inferred and future names could be registered to take over the botnet, as in [63]. Another trick that is also applied in such architecture is the use of an intermediate point, where the infected machine first calls a static domain and then receives a list of current server addresses [45].

Where domain names are used, it is also necessary to call the DNS protocol [75] to find the correct address of the machine to be contacted (either for seeking binaries containing the bot and/or to contact the C&C server). Therefore, monitoring the DNS service may also help identify the botnet [76].

After establishing the command and control channel, the bot waits for commands to perform malicious activities [12,13,25,74]. Thus, the bot passes into phase 4 and is ready to perform an attack.

At this point, message exchanges may occur more intensely, with several messages being exchanged over a short period of time. However, C&C traffic is not high volume and does not cause high network latency. Therefore, anomaly-based techniques may not identify botnet C&C traffic [13].

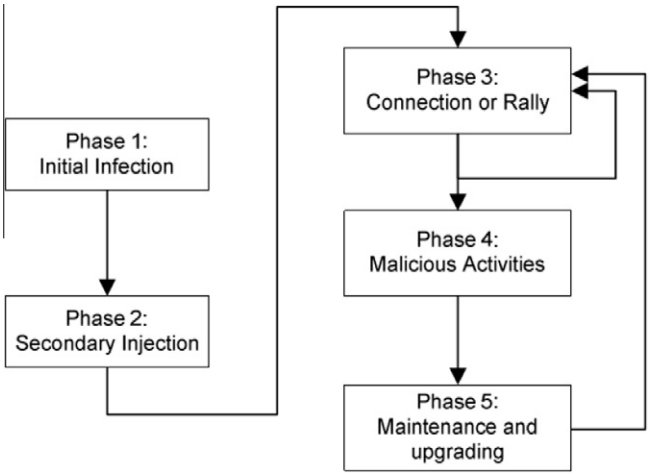

Fig. 3. The life cycle of a botnet.

Malicious activities may be as wide ranging as information theft, performing DDoS attacks, spreading malware, extortion, stealing computer resources, monitoring network traffic, searching for vulnerable and unprotected computers, spamming, phishing, identity theft, manipulating games and surveys, etc. [6,12,23,32,36,69].

The last phase of the bot life cycle is the maintenance and updating of the malware. Maintenance is necessary if the botmaster wants to keep his army of zombies. It may be necessary to update codes for many reasons, including evading detection techniques, adding new features or migrating to another C&C [12,13,74]. This phase is also usually considered a vulnerable step. As the botmaster intends to disseminate updates as soon as possible, some behavioral patterns of the stations belonging to the network may emerge and make the botnet detectable. Changes in behavior are typically observed, for instance, in DNS queries and file sharing, among other areas. After bots are updated, they must establish new connections with the C&C infrastructure.

2.6. Architectural designs

The command and control channel (C&C), the means by which individual bots form a botnet [77], may be classified according to its specific architecture [1,32,35,45,77] and operational modes [45], whether it is centralized, decentralized, hybrid or random architectures, and persistent or periodic (sporadic) modes.

2.6.1. Centralized C&C

The centralized approach is similar to the classic client-server network model. Typical examples of this type of botnet architecture are those implemented through the Internet Relay Chat (IRC) protocol [78]. In a centralized C&C infrastructure, all bots establish their communication channel with one, or a few, connection points. These are usually command and control servers that are responsible for sending commands to bots and to provide malware updates.

The advantages of a centralized architecture are quick reaction times and good coordination. Direct feedback enables easy monitoring of the status of the botnet by the botmaster, providing important information about certain fundamental properties, such as the number of active bots or their global distribution.

The main problem with centralized architecture is that the C&C server itself is a central point of failure [32,35], allowing a detected botnet to be turned off quite easily. This weakness motivated the development of decentralized architectures.

The main protocols used in centralized architectures are Internet Relay Chat (IRC) [78] and HyperText Transfer Protocol (HTTP) [79]. In the case of IRC botnets, a botmaster creates IRC channels on the command and control (C&C) server, to which the zombie machines will connect and then wait for commands to perform a malicious activity. This architecture was, and still is, widely used by botmasters [13,25,32,77,80,81]. According to the 2010 Symantec Internet Security Threat Report [8], 31% of the centralized C&C servers that were observed in 2009 used IRC as their main communication protocol.

An interesting feature of the IRC protocol is the possibility of multicast communication through groups called "communication channels" or of private unicast communication between two members. This feature allows the botmaster to have flexible control of his botnet; for instance, he can select a specific group of bots to carry out an attack. In addition, there are several open source implementations for IRC servers, allowing the botmaster to tailor the protocol to suit his needs. Other advantageous properties for attackers found in IRC botnets are redundancy, scalability and versatility, all of which allow code reuse for bots and the creation of new botnets [39].

In addition, contemporary IRC botnets have evolved from simple dialects to the obfuscation of C&C content in IRC messages by using, for example, a custom dialect, a foreign language, or a naive obfuscation technique such as simple XOR, substitution, or hashing. By using obfuscated IRC messages (e.g., "hello" instead of "scan"), these botnets may evade signature-based detection [82,83] and honeypot-based tracking approaches [76]. Indeed, a substantial collection of current IRC botnets have been observed utilizing obscure C&C communications [81,84].

Even though the IRC protocol is very flexible and suitable for use in a C&C channel, it has serious limitations because it is normally easy to detect and interrupt the operation of an IRC botnet. Detection is facilitated because IRC traffic is not common and is rarely used in corporate networks; in fact, it is usually blocked. So a network administrator may prevent IRC botnet activity simply by detecting IRC traffic in the network and blocking it with firewalls [32,77].

Due to the restrictions on IRC traffic in corporate networks, the HyperText Transfer Protocol (HTTP) [79] became popular as a mechanism for implementing C&C communication [32,45,77,79]. Its chief advantage over IRC communication is that HTTP traffic is permitted in most networks, disguising the communication between bots and botmaster. However, this protocol still suffers from the problem of being the central point of failure because it also employs a centralized architecture.

2.6.2. Decentralized C&C

Modern botnets require great flexibility and robustness to be able to handle large numbers of bots and to maximize profits. In [1], decentralized and random architecture for the C&C channel was addressed for the first time. Botnets that have a decentralized architecture are more difficult to be disarticulated because the discovery of several or even many bots does not necessarily mean the loss of the entire botnet because there is no central C&C server to be found and disabled.

Such decentralized botnets are commonly based on a variety of P2P protocols and work as an overlay network that can be classified according to [85] as follows:

- Unstructured P2P Overlays: The broad class of unstructured overlays refers to random topologies with different degrees of distribution, such as power-law networks or uniform random networks. They offer no

possibility for routing or key lookup. They support flooding, random walk, and variations of the foregoing as search methods. Gossip protocols are also well supported.

- Structured P2P Overlays: In structured P2P networks, mapping is created between the content and its location. A distributed hash table (DHT) for routing is usually implemented in this type of network. CAN, Chord, Pastry, and Tapestry [86] are the first four DHTs introduced. Kademlia [39,87,88] is another DHT algorithm that has been used in Overnet, eMule and BitTorrent. Contemporary P2P botnets are based on structured overlays, such as Kademlia.
- Superpeer Overlays: In superpeer networks, all peers are not equal, and a small subset of the peers are automatically selected as temporary servers to support network functions, such as search and control. Several P2P applications such as Skype, FastTrack and Gnutella [89,90] apply superpeers. Because superpeer networks are more visible and more vulnerable to targeted attacks, we assume that the most efficient botnets are not likely to adopt this design. Bots that belong to this class usually have a valid IP address and are not under firewalls or DHCP.

The construction of a P2P botnet generally involves the following two steps: (i) the selection of peer candidates and (ii) the implementation of the necessary actions to make the peer candidates members of the botnet. For the selection of candidates, three different subcategories of P2P botnets are mentioned in [91], and each subcategory focuses on a certain type of candidate (see Table 2):

- Parasite: In a parasite P2P botnet, all bots are selected from vulnerable hosts within an existing P2P network. However, the number of vulnerable hosts in the existing P2P network limits the scale of a parasite botnet. Such an approach is not flexible and greatly reduces the number of potential bots under the botmaster's control.
- Leeching: A leeching P2P botnet refers to a botnet whose members join an existing P2P network and depend on this P2P network for C&C communication. In this case, bot candidates may be vulnerable hosts that were either inside or outside an existing P2P network. A botnet of this type was the early version of the Storm [52,53,92].
- Bot-only: A bot-only P2P botnet builds its own network in which all members are bots, such as Stormnet and Nugache [48,52,53,92].

After the initial infection is successfully accomplished, the second step, or the "bootstrap procedure", makes the bot able to receive and pass commands, integrating it into the entire botnet. In the bootstrap procedure, botnets take advantage of specific P2P protocols in use, as mentioned by Wang et al. [91]. Contemporary P2P file-sharing networks provide the following ways for new peers to join the network.

1. An initial list of peers is hard-coded in each P2P client. When a new peer turns up, it will try to contact each peer in that initial list to update its neighboring peer information. Unstructured P2P networks commonly adopt such a bootstrap procedure.
2. There is a shared web cache, such as the Gnutella web cache, stored at some place on the Internet, and the location of the cache is put inside the client code. Thus new peers can refresh their neighboring peer lists by retrieving the latest updates from the web cache. Such a procedure is common in structured P2P networks.
3. If the P2P network employs superpeers, a new bot joining the network will attempt to access them so that it can update his "peer list". Superpeer addresses are usually hard-coded in the bots.

The bootstrap procedure may be considered the weak point of a P2P botnet because discovery of the initial list compromises network growth, and the botnet can be overthrown. An alternative mechanism to avoid this weakness was proposed in [93]. When an infected host *i* infects a new victim *j*, its peer list is passed to the victim. Host *i* chooses with a given probability whether to replace one IP address in its own peer list with host j's IP address. If host *j* has already been infected, host *j* updates a part of its own peer list with the new one sent from host *i*.

One example of a bot that uses P2P architecture for communication is Nugache [43], which employs a list of 22 alternative addresses to be contacted during secondary injection to receive the list of peers in the network. However, not all P2P botnets need to work with lists of previously determined servers. Sinit does not connect to any server to retrieve its list of peers, performing instead a search of random IP addresses to accomplish this task [43].

2.6.3. Hybrid model C&C

Hybrid architectures employ characteristics from both centralized and decentralized botnets. A typical example is the use of P2P architecture with superpeers, as discussed in Section 2.6.2. However, more advanced hybrid P2P architectures have been investigated in the literature.

In Ref. [35], bots belonging to a hybrid P2P botnet are classified into two distinct groups, servant bots and client bots. Servant bots behave as both clients and servers; they are configured with static and routable IP addresses. Client

Table 2
Types of P2P botnets – Wang et al. in [91].

Features	Parasite	Leeching	Bot-only
Infection vectors	P2P malware	Any type of malware	Any type of malware
Bot candidates	Vulnerable hosts inP2P networks	Vulnerable hosts in Internet	Vulnerable hosts in Internet
Bootstrap procedure	None	Required	Optional
Members in the network	Legitimate peers &Bots	Legitimate peers &bots	Only bots
Communications protocols	Existing P2P protocols	Existing P2P protocols	Self-designed or existing P2P protocols

bots, on the other hand, do not accept incoming connections and are configured with dynamically designated or non-routable IP addresses. They can also be located behind firewalls without a global connection to the Internet.

The proposed architecture has the following features. Servant bots are the only candidates that can have their IP addresses on the peer lists. They listen to a determined port for incoming connections and use a self-generated symmetric encryption key for communication, which makes botnet detection more difficult. All bots must periodically connect to the servant bots on their peer list to retrieve commands issued by their botmaster. When a bot receives new commands that it has not previously observed, it quickly forwards the command to all servant bots on its peer list.

2.6.4. Random model C&C

The random model was introduced by Cooke et al. [1] as a model for future botnets that wish to operate for a long time. In this proposal, the bot does not actively contact the botmaster or other bots. Instead, it waits for connection attempts by the botmaster. To perform an attack, the botmaster scans the network to find zombies, and if it finds one, it sends commands to the bot.

This model has the advantage of being easy to implement and fairly resilient, as there is no common communication characteristics between bot and botmaster, making it harder to detect and interrupt. Due to the need for scanning, there are scalability issues and attack coordination problems in this architecture. No real bot currently uses this strategy; it is only considered a theoretical model.

3. Detection techniques

Botnet detection is perhaps one of the first actions that should be taken when combating network security threats. Given the potential power of botnets to conduct different malicious activities and cyber warfare, detection techniques play an important role in this process. Researchers have developed several architectures for detecting such threats and have proposed a number of botnet detection taxonomies [13,17,69].

According to a previous study Ref. [69], detection techniques are classified into two main categories: those based on setting up honeynets [7,94] or Intrusion Detection systems (IDSs). The last category has been further divided into two subcategories: signature- and anomaly-based detection systems.

Honeynets are best suited for collecting information from bots. After collecting information, it is possible to learn and understand the technology used and perform a complete analysis of the main botnet characteristics. It is often viable to provide a detection system with a bot signature to discover C&C servers, previously unknown vulnerabilities, tools and techniques used by the attacker and the attacker's motivation.

Honeynets can also be used to obtain bot binaries and infiltrate those botnets [7,25,30]. Some techniques employ honeynets to capture bots [1,25,30,72,95–104].

Honeynets are essential to understanding botnet characteristics and technology, but they have several limitations:

- Limited scale of exploited activities that they can track.
- Cannot capture bots that do not use propagation methods other than those based on scanning, spam and web-driven downloads.
- Only able report information about the infected machines placed as traps.

As honeynets become increasingly popular in monitoring and defense systems, intruders also begin to seek ways to avoid honeynets traps [105,106].

IDS botnet detection is classified as either a signature- or anomaly-based technique. Signature-based detection techniques apply signatures of current bots into the IDS detection system [82,107,108], such as SNORT [109]. The basic idea is to extract feature information from packets of monitored traffic, mark such patterns and register them in a knowledge database of existing bots.

From the signature database, it is not difficult to perform the necessary detection tasks by simply comparing every byte in the packet. Although this technique has some advantages, signature-based detection approaches can only detect well-known botnets (those mapped before). Consequently, this solution is not functional for unknown bots, as it cannot detect zero-day bot attacks. More importantly, the detection mechanism may miss similar bots with slightly different signatures.

Another disadvantage in signature-based detection techniques is that there should always be an effort to update the knowledge base with new signatures, which enhances the management cost and reduces the overall performance. New bots may launch attacks before the knowledge base is patched [107].

Anomaly-based detection can be considered the main research area for botnet detection techniques. The idea is to perform botnet detection considering several different network traffic anomalies, including high network latency, high traffic volume, traffic on unusual ports and unusual system behavior that could indicate the presence of malicious bots in the network [24,110]. Anomaly-based techniques are further divided into host- and network-based categories.

In a host-based approach [111], the individual machine (host) is monitored to find any suspicious behavior, including its processing overhead, and access to suspicious files. Despite the importance of this approach, it is usually not scalable because all machines in the network must have the monitoring tool installed to be effective.

Conversely, network-based techniques analyze network traffic either passively or actively. In active monitoring, packets are injected into the network to measure network response. An example of active monitoring is BotProbe [112]. A disadvantage of this approach is that it generates additional network traffic.

Although there are many botnet detection techniques, detection is an arduous task [2]:

Table 3
Botnet detection techniques .

<table>
<tr><td colspan="8">Botnet Detection Techniques</td></tr>
<tr><td>Honeynet-Based</td><td colspan="7">Intrusion Detection system (IDS)</td></tr>
<tr><td rowspan="4">[1, 7, 25, 30, 72, 94–104]</td><td>Signature-Based</td><td colspan="6">Anomaly-Based</td></tr>
<tr><td rowspan="3">[82, 107, 108]</td><td>Host-Based</td><td colspan="5">Network-Based</td></tr>
<tr><td rowspan="2">[45, 103, 111, 113, 114]</td><td>Active Monitoring</td><td colspan="4">Passive Monitoring</td></tr>
<tr><td rowspan="2">[112]</td><td>IRC</td><td>DNS</td><td>SMTP</td><td>P2P</td><td>Multiporpose</td></tr>
<tr><td>[1, 82, 110, 115–117]</td><td>[2, 25, 72, 76, 118]</td><td>[9, 119–122]</td><td>[52, 53, 123– 127]</td><td>[5, 41, 49, 77, 104, 108, 128– 131]</td></tr>
</table>

- Botnet traffic is similar to normal traffic and may apply encryption in the communication channels to evade detection, especially in signature-based techniques and packet content analyses.
- Botnets are evolving rapidly as more users fail to protect their machines. Botmasters also use techniques such as fast-flux hosting to evade detection mechanisms.
- It is usually necessary to analyze a massive amount of data, which is difficult to perform in real time, thus making detection in large-scale networks a prohibitive task.

Table 3 summarizes the main detection technique classifications.

3.1. Host-based

Host-based techniques analyze the machine behavior. When a bot is running, it performs call sequences to system libraries (e.g., changes in the system registry, file system, changes in network connections) different from those executed by legitimate processes [32]. For example, when an antivirus cannot perform update of its database, it may indicate malware infection. One advantage in using host-based detection approaches is that they are much more effective against download attacks and onset infections in general [114].

Despite being an important approach to minimize malware spread, performing individual station analyses and monitoring them is a complex costly and non-scalable task.

A previous study [111] tests the hypothesis that the behavior of installed bots is distinguished from the behavior of innocuous processes in the host. Each participating bot independently executes commands received from the C&C network, where commands take a certain number of parameters (possibly zero), each of a particular type, in some fixed order. Using parameterized commands separates bot behavior from the normal execution of innocuous programs. To perform this check, a tool called BotSwat is implemented to monitor an environment containing home users' PCs running Windows XP or 2000 [7]. BotSwat monitors the execution of an arbitrary Win32 binary and intercepts the run-time library calls (including system calls) made by a process. This tool is implemented to detect generic botnets, regardless of particular command-and-control communication protocol (e.g., IRC) or botnet structure (e.g., centralized or peer-to-peer).

Masud et al. [113] propose effective flow-based Botnet traffic detection by mining multiple log files. Bots respond much faster than humans, so this type of feature could be captured by correlating host-based log files. The authors propose several log correlation for C&C traffic detection and show that the proposed method could also be applied to detect non-IRC Botnets.

Another study [45] models a typical bot using three invariant features during its onset: (1) the bot startup is automatic without requiring any user actions; (2) a bot must establish a command and control channel with its botmaster; and (3) a bot performs local or remote attacks eventually. These invariants indicate three indispensable phases (e.g., startup, preparation, and attack) for a bot attack. A tool called BotTracer is implemented to detect these phases with the assistance of virtual machine techniques.

Because a bot must actively contact a rendezvous point to build a command and control channel with its controller, BotTracer captures these channels and compares them to known characteristics of bot command and control channels. This can significantly narrow the detection space. BotTracer also constantly monitors system-level activities and traffic patterns of processes that have been identified as suspicious. A fundamental assumption of BotTracer is that a bot cannot detect the virtual machine. However, in practice, many techniques can detect virtual machines.

Previous work [103] presents a hybrid model for bot detection, which combines a host intrusion detection system and an operating system event log analyzer. The proposed solution, called Model of Multi-Agent Bots Detection System (MABDS), is built using multi-agent systems. MABDS comprises the user agent, an administrative agent, a central knowledge database, agent collections, system analysis, network analysis, and honeypots.

The work in [114] implemented a tool called DeWare, a host-based security tool for detecting the malware infection onset through the novel use of rules that enforce the correct dependency characteristics in an operating system. This tool forces policies to control how processes are created and file systems are accessed. It can be used to detect any type of drive-by downloads, including the browser-based exploits. The detection process is based on observing stealthy download-and-execution patterns, which many

active malware packages exhibit at their onset, including the recent Hydraq malware [132]. The results demonstrate that DeWare could correctly distinguish legitimate download events from unauthorized system events with a low false positive rate (<1%).

3.2. Network-based

Detection techniques based on network monitoring are the most used today [74]. These techniques are classified into active and passive monitoring. Some techniques are specifically created for some protocols, and others try to be more generic, involving multiple protocols and architectures.

3.2.1. Active monitoring

The work in [112] presents an active technique named BotProbe. The idea is to dynamically inject packets that probe the internal client to determine whether a human or bot is managing that side of the communicating/chatting session. Such an approach works in a cause-effect correlation because bots are non-human agents and present a deterministic command-response pattern. The authors observe that bots are pre-programmed to respond to certain predefined commands, and responses are consistent across command repetition. Different from normal human chatting, the command-response pattern has a strong cause-effect correlation, i.e., the command deterministically causes the response. The system was validated on several malicious IRC bots in a study of approximately 100 real users.

An advantage of this technique is related to response time, as it requires at most one round of actual C&C to provide a result. Other approaches usually have slow detection responses because it is often required to observe the following phases: multiple different infection stages (BotHunter) [128], multiple instances/rounds of communications/activities (Botsniffer) [49] and a long communication/activity time (BotMiner) [77]. BotProbe is not intended to replace existing passive detection approaches, but it may complement them from a new perspective.

Active methods have the great disadvantage of increasing network traffic with additional packets sent to suspicious machines. Moreover, and most importantly, injecting packets facilitates detection tracking tools and may be subject to legal issues.

3.2.2. Passive monitoring

Passive monitoring techniques observe data traffic in the network and look for suspicious communications that may be provided by bots or C&C servers. Data traffic is analyzed, employing pre-recorded signatures or anomaly detection techniques.

An important premise of passive monitoring is that bots in the same botnet tend to present the same communication patterns [32,77] in both centralized and decentralized architectures (e.g., P2P). This occurs because bots are pre-programmed to perform the same routine communication with the C&C server and for the same botmaster.

Because botmasters must communicate with bots to perform the attack, there are common traffic patterns in the network linked to each stage of the bot life cycle. Moreover, the same network protocols will be used for communication and performing malicious activities [32]. Several papers explore such similarity in both WAN and LAN traffic to perform bot detection [2,74,77].

Passive detection employs a myriad of different techniques and methods, including statistics approaches [74,123,124,133–135], traffic mining [77,113,125,136], visualization [101,129,136–138], graph theory [28,138–146], clustering [124,133,147–151], correlation [128,151–153], stochastic models [9,154], entropy [133,155,156], neural networks [157], decision trees [158], discrete Fourier transform [126,159], CUSUM [160], machine learning [113,152,153,161,162], discrete time series [5], group analysis [49,128,162], and a combination of existing techniques, in some cases.

The next subsections discuss passive detection techniques oriented with respect to the network protocols employed by botnets.

3.2.3. IRC protocol

One of the first techniques proposed to detect IRC bots can be found in [1]. The work outlines the origins and structure of bots and botnets, studies the effectiveness of detecting botnets by directly monitoring IRC communication and shows that a more comprehensive approach was required at that time. The proposal correlates the monitored traffic with additional features observed in the infected machines. Tracking traffic features that do not correspond to human standards may indicate the presence of bots in the network.

Another study [110] proposes an anomaly-based algorithm for detecting IRC-based botnet meshes. The algorithm can also reveal bot servers. It combines an IRC mesh detection component with a TCP scan detector. However, simply using a minor cipher to encode the IRC commands could easily crush that approach.

Another work [162] uses a correlation algorithm to detect the presence of a botnet and identify C&C servers using passive traffic analysis. The method contains three stages, each of which uses flow characteristics: filtering traffic unlikely to be bot C&C (e.g., number of packets, bps, packet size, and duration), classifying traffic as likely to be IRC or not (e.g., duration, role, bpp, bps, and pps), and clustering related flows (e.g., characteristics relating to inter-arrival time and packet size). Flows are then clustered and classified according to IRC-like traffic patterns in a five-dimensional space considering packet inter-arrival times and packet sizes. Flow perturbation could be used to defeat each stage; the simplest approach targets the filtering of high bitrate flows by injecting packet- and flow-level noise.

Additionally, [115] studied network flow-level IRC botnet controller detection for backbone networks. The proposal combines heuristics that assume the network flow of IRC communication, scanning behavior, and known botnet communication models for backbone networks. All flow records for suspected bots are fetched and pruned, keeping only the following flows: which server port is one of the standard IRC ports or which involves a hub server. The flow records for server IP, i.e., the server port tuples with the most suspected bots, are aggregated, and a

correlation algorithm is used to identify suspicious bots according to a defined bot infection dialog model. Those that sufficiently resemble the model for IRC in their average fpa, bpp, and ppf values undergo a heuristics analysis, which entails identifying the number of peers and idle clients. As in previous techniques, flow perturbation could be used to defeat this approach.

Rishi [82] is a signature-based IRC botnet detection system that tracks IRC bots using well-known IRC bot nickname patterns as signatures. Rishi is primarily based on passive traffic monitoring for odd or suspicious IRC nicknames, IRC servers, and uncommon server ports. It employs analysis and a scoring system to detect bots that use uncommon communication channels, which classical intrusion detection systems have not detected. The disadvantage of that method is that it cannot detect encrypted communication or non-IRC botnets.

Strayer et al. [116] propose a network-based approach, which they evolved from [162], to detect botnet traffic using machine-learning techniques. The detection process uses two main steps: first, traffic that is unlikely to be part of a botnet is eliminated; the remaining traffic is then classified into groups and correlated to find common communication patterns that would suggest botnet activity. They show that botnet activity evidence was extracted from a traffic trace containing over 1.3 million flows. Nevertheless, it has the same weakness as the previously mentioned work.

Additionally, [117] proposed a similar approach to [113] for detecting and characterizing IRC botnets. They first classify the network traffic into different applications using payload signatures and a novel clustering algorithm; they then analyze the specific IRC application community based on temporal-frequent flow characteristics. That approach differentiates malicious IRC channels created by bots from normal IRC traffic generated by human beings. A specific IRC anomaly detection procedure is to measure IRC response times. A human cannot respond as quickly as malicious software. Consequently, it is possible to compare all response times and find bot communities.

A difficulty in detecting IRC bots is that real-world IRC-based botnet C&C communications can be quiet, i.e., some have infrequent C&C interactions because the botmaster is not always online to command the bot army. If the C&C interaction frequency is low enough, botnets could potentially evade detection. Indeed, stealthy botnets with small sizes, obfuscated C&C dialogs, and infrequent C&C interactions pose an ongoing challenge to the malware research community.

3.2.4. DNS protocol

DNS-based detection techniques use DNS [75] information generated by a botnet. As mentioned above, bots normally begin a connection with the C&C server to obtain commands. To access the C&C server, bots perform DNS queries to locate the particular server that a DDNS (Dynamic DNS) provider typically hosts. It is thus possible to create a detection mechanism that monitors DNS traffic and searches for some DNS anomalies [118,163].

Analyzing DNS queries may provide relevant information about botnet existence and C&C server locations [164,165]. As bots perform queries in the same domains, there is a relationship among surveys from several bots in those domains. Other information can also be verified, including domain lifetime, query lifetime (TTL), query frequency, and domain PageRank.

Researchers [164] analyze DNS traces to identify the agents participating in the service: clients, local DNS servers, and authoritative root. Data analysis employs a categorization based on directed graphs, where nodes represent machines (IP addresses) involved in queries and edges represent the queries or responses involving nodes. Several graph metrics are listed, including the degree of input and output nodes, the edges' bi-directionality, the number of connected components, the graph structure, and connectivity. The proposed technique correctly identifies the agents involved in the DNS and the possibility of identifying botnets simply by analyzing large trace sets.

The DNS-based Blackhole List (DNSBL) is used to publish the addresses of computers or networks linked to spamming and other malicious activities. Some botmasters perform lookups against the DNSBL to determine whether their spamming bots have been blacklisted. A previous work [76] develops techniques and heuristics for detecting DNSBL reconnaissance activity, trying to catch the botmaster address and identity it. However, such an approach is not effective because it is not difficult to design evasion strategies.

Dagon et al. [72] identify key metrics for measuring botnet utility and describe various topological structures that a botnet may use to coordinate attacks. The authors claim that it is possible to consider the ability of different response techniques to degrade or disrupt botnets using the proposed performance metrics. The work measures canonical DNS request rates and DNS density comparisons of botnets rallying DNS traffic. The main drawback of the approach is that it could be evaded if botmasters know the mechanism or suspect it is running. Botmasters may also poison the scheme through faked DNS queries, thus generating many false alarms.

Another study, [25] presents a measurement methodology that can be applied to study botnets using honeypots. The authors construct a multifaceted infrastructure to capture and concurrently track multiple botnets in the wild, and they achieve a comprehensive measurement analysis that reflects several important structural and behavioral aspects of botnets. They studied the botnet behavior, botnet prevalence on the Internet, and modeled botnet life cycle.

Choi et al. [2] suggested an anomaly detection mechanism using monitoring group activities in DNS traffic. Based on the group activity model and metric, they developed a botnet detection mechanism, called BotGAD (Botnet Group Activity Detector). They defined some special features of DNS traffic to differentiate valid DNS queries from Botnet DNS queries. BotGAD enables detecting unknown botnets from large-scale networks in real time. The authors also developed a mechanism to detect C&C server migration. The scheme may also detect botnets with encrypted channels, as it uses information from IP headers. The main drawback of the approach is the high processing time required to monitor the huge scale of network traffic [13].

Additionally, [118] evaluated two approaches for identifying botnet C&C servers based on anomalous DDNS traffic. The first approach looks for domain names whose query rates are abnormally high or temporally concentrated. High DDNS query rates may be expected because botmasters frequently move C&C servers, and botnets with as many as 1.5 million bots have been discovered. The second approach looks for abnormally recurring DDNS replies, indicating that the query is for an inexistent name (NXDOMAIN). Such queries may correspond to bots trying to locate C&C servers that have been taken down. The approach based on abnormally high or temporally concentrated query rates is ineffective because many legitimate and popular domains use DNS with short time-to-live (TTL) records, including gmail.com, yahoo.com, and mozilla.com, which may misclassify those names as C&C servers. Conversely, the approach based on abnormally recurring NXDOMAIN replies is effective and can identify several domain names independently reported as being suspicious.

3.2.5. SMTP protocol

Husna et al. [119] investigates the behavior patterns of spammers based on their underlying similarities in spamming. The authors use a Principal Component Analysis (PCA) to identify features, which accounts for the maximum variance in the spamming patterns. They calculate the proximity between different spammers and classify them into various groups, which represent similar proximity.

Zhuang et al. [120] develops techniques to map botnet membership using email spam traces. To group bots into botnets, they looked for multiple bots participating in the same email spam campaigns. The authors applied the proposed technique against an email spam trace from Hotmail web mail services. Through this analysis, they made indirect observations about the sizes and activities of different spam botnets behavioral characteristics (e.g., the amount of spam sent per bot) and the geographical botnet distribution. The authors assume that a spam campaign is realized for one botnet. However, as previously discussed [122], some spam campaigns utilize multiple botnets. Some drawbacks are that such an approach cannot uncover botnets not involved in email spamming. Analyzing "incoming" spam feeds provides valuable information on aggregate botnet behavior, but it does not separate the activities of individual botnets or provide information on the spammers' latest techniques.

BotGraph [121] is a tool to detect web-account abuse attacks targeting major Web email providers. The tool uncovers the correlations among botnet activities by constructing large user–user graphs and looking for tightly connected sub-graph components. The approach can identify stealthy botnet users that are hard to detect when viewed in isolation. BotGraph was implemented as a distributed application on a computer cluster and explored a number of performance optimization techniques.

BotLab [122] is a tool to correlate incoming email spam with outgoing spam collected from known bots in a controlled environment. BotLab gathers multiple real-time information streams about botnets taken from distinct perspectives. By combining and analyzing these streams, BotLab produces accurate, timely, and comprehensive data about spam botnet behavior. It presents interesting behavioral characteristics of spamming botnets derived from their multi-perspective analysis. The authors show that a handful of botnets are responsible for most received spam. Multiple botnets often simultaneously participate in a single campaign, contrary to the assumption made by prior research [120]. "Canadian Pharmacy" is distributed by Kraken [62], MegaD [50], Pushdo [54], Srizbi [46], and Storm [52]. This suggests that the most prominent spammers utilize the services of multiple botnets. Other contributions of the work include creating a real-time botnet monitoring platform, identifying new bot variants, and developing network sandboxing mechanisms that prevent captive bot nodes from causing harm.

Researchers [9] have proposed a tool named BotMagnifier to support identifying and tracking bots that send spam. It takes as input an initial set of IP addresses known to be associated with spam bots and learns their spamming behavior. The approach can effectively model spam behavior; the need to provide an initial IP set list constitutes a serious drawback.

3.2.6. P2P protocols

P2P traffic accounts for over 70% of the overall traffic across the Internet [166,167]. The lack of control in the shared content exchanged in P2P networks along with the increasing bandwidth available to users have made P2P applications a major source of illegal content sharing (e.g., pirated movies or music). Motivated by this fact, several studies have been conducted to identify and block this type of traffic in controlled network environments.

Because most P2P botnets are closely related to common P2P protocols and applications, techniques to detect P2P botnets also benefit from previous approaches. Many authors employ techniques based on payload analysis to identify P2P applications. These techniques focus on information about port numbers, protocols and strings contained in the packets. Constantinou et al. [168] criticized such approaches based on two main points: (a) they cannot be used if payload information is not available and (b) they generally cannot identify unknown traffic classes.

Payload information is not always accessible for several reasons. First, legal and privacy issues may prevent network administrators from reading the actual content of the packets that are sent through the network [169]. Some applications may also encrypt their payloads, thus making them impossible to read. Examining the payload to classify traffic in real time is impractical due to its high overhead, especially if there is high network utilization. Another limitation in using payload for traffic identification is that the classifier can only identify traffic it has already observed. Unknown traffic, such as new or modified P2P applications, cannot be easily identified by reading the payload.

For port number identification, most applications no longer use standard ports and often prefer higher port numbers. Some applications may even use previously defined service ports to try to evade detection mechanisms.

A previous study [170] is one of the first papers to contest the efficiency of techniques based on signature searching in packet payloads. Other techniques have been

proposed, accounting for some inherent characteristics of P2P protocols instead of predefined signatures. Such techniques can also be applied to detect P2P botnets and locate such applications (bots) in the network.

Some common P2P features used to construct detection techniques are described below:

- **TCP/UDP IP pairs:** P2P networks tend to use both protocols (TCP/UDP), TCP to exchange messages and UDP for signaling (PING-PONG messages), to prevent the channel from being idle. To identify P2P hosts, one can thus look for pairs of source–destination hosts that use both transport protocols. There are legitimate applications that also exhibit this behavior (DNS or streaming media), requiring an analysis of other features.
- **(IP, port) pairs:** To connect to distributed networks, each P2P client maintains a starting host cache. Depending on the network, the host cache may contain the IP addresses of other peers, servers or superpeers. This pool of hosts facilitates the initial connection of the new peer to the existing P2P network.
- **Connection responded success rate:** Liu et al. [171] study the nature of P2P traffic in contrast to the traditional Client/Server applications traffic. In a P2P network, every host acts as a client and server at the same time. P2P hosts thus may differ in processing capabilities, connection speeds, local network configurations or operating systems. To keep its download speed stable, a P2P host continually tries to initiate connections with other hosts; due to the dynamics of P2P systems, some hosts may be offline. Conversely, a connection of a typical Client/Server applications has a much higher success rate. This property can be summarized in the following equation:

$$R = \frac{N_a}{N_s} \tag{1}$$

where R represents the connection success rate and the number of IP destinations that responded to the host. The overall number of connection attempts tends to be lower in P2P applications than in non-P2P applications.

- **Mean port numbers used to communicate with external hosts:** A study [172] verified that P2P applications usually set random port numbers for communication. As there are 65,535 total ports, the probability of a given port be selected is 0.015%, and the probability of a lower port (below 1024) being chosen is 15%. In P2P applications, the mean port number value is usually high.
- **Up/Down traffic ratio:** Because P2P nodes may behave as both clients and servers, download and upload rates tend to be more balanced than in other client/server applications, including HTTP and FTP [173]. The ratio of download and upload rates can also be considered to differentiate P2P and non-P2P applications.
- **DNS use:** A previous study [123] shows that P2P nodes rarely use DNS before starting data communication. P2P hosts directly employ IP peer lists to search for online peers to exchange information. DNS requests occur only when the architecture uses superpeers with registered domain names.

Zhang et al. [123] develop a methodology that seeks P2P botnets using some of the parameters described above. The first step of the method is to search for P2P applications and output a list of P2P nodes found. Using the list, some typical bot behavior is considered to mark suspicious nodes (bots) in the list. Fig. 4 shows an outline of the proposed methodology.

Other typical bot behavior [123] is a long application lifetime. Experiments have shown that legitimate P2P applications have a lifetime that is often lower than system uptime. Bots must preserve an active connection with the botnet as long as possible to be useful for the botmaster. In this sense, bots commonly start at system boot time, being active in the background while the system is up.

Some papers [124] employ clustering techniques such as K- and X-means to obtain typical botnet community behavior. The main observation is that legitimate P2P users tend to have a different behavior, while bots act like a community [163] of automated agents. The probability that multiple hosts with P2P legitimate applications do the same data searches and transfers at the same time interval is remote. The main disadvantage of such approaches is the long time required for those algorithms to process data from an entire network. Depending on the data, they can take up to 10 days to be processed [124]. Such a long period of time is not feasible for botnet detection techniques as networks may change their main characteristics faster.

Liao et al. [125] use a methodology based on packet sizes to distinguish P2P botnet traffic from legitimate P2P traffic and the rest of Internet traffic. The research suggests a detection method based on the following three hypotheses. First, P2P botnet communications mimic a P2P structure to set up massive communications sessions among bots to create a multiple controller and mesh network. Second, to not interfere with the Internet, a P2P bot maintains session and data transmission with other bots rather than stopping communicating after getting contacted. Third, for privacy and to avoid being discovered, data are transmitted at a minimum level in bot communication. After many observations, it was found that P2P botnet packets tend to have an average size lower than other regular network packets, as in Fig. 5.

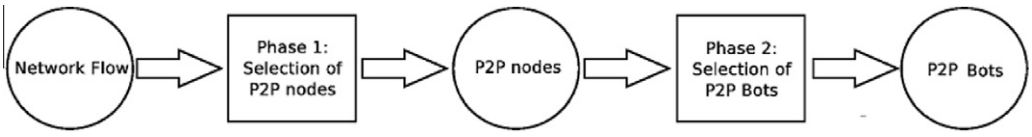

Fig. 4. Proposed methodology in phases [123].

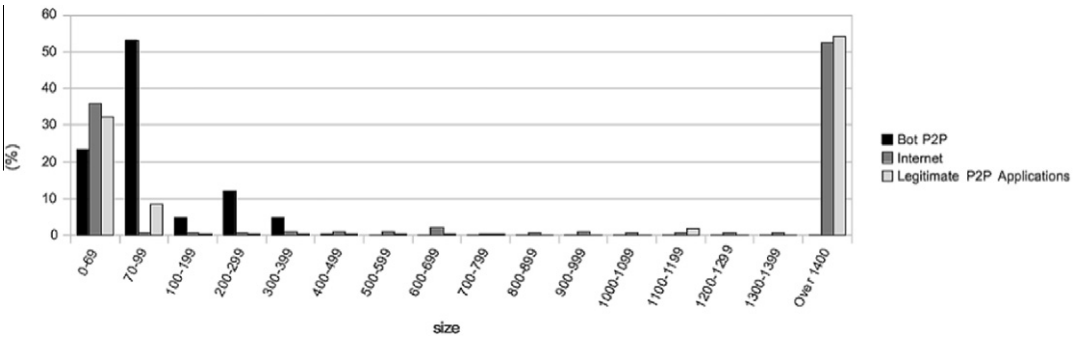

Fig. 5. Percentage of packet size: Botnet P2P, internet and Legitimate P2P applications.

Yu et al. [126] propose a methodology based on DFT (Discrete Fourier Transform) to find communication patterns among bots. The technique has a relatively low processing time being adequate for online implementation as its main advantage. However, its false positive rate is relatively high compared to other techniques (over 10%). The authors use an initial volume reduction filter to remove any traffic notoriously irrelevant for the analysis. Such a filter is based on blacklists, whitelists and protocols used by C&C channels. The remaining flow is processed using DFT, and a graphic communication pattern is obtained. Analyzing successive network graphs generated by the technique over a given period of time allows observing patterns that can be classified as a bot.

Nagaraja et al. developed [127] BotGrep, a tool to detect P2P botnets based on network graph analyses, e.g., the information about which pairs of nodes communicate with one another (communication graph). The approach relies on the fast-mixing nature of the structured P2P botnet C&C graph. The BotGrep algorithm iteratively partitions the communication graph into faster- and slower-mixing pieces, eventually narrowing it to the fast-mixing component. The network graph analysis assumes that hosts belonging to P2P botnets tend to be more connected than other hosts.

Douceur [174] presented the Sybil attack, which explored the lack of a certification authority in P2P networks to assume multiple identities and manage to control the network. A Sybil attack is one in which an attacker subverts the reputation system of a P2P network by creating many pseudonymous entities, using them to gain a disproportionately large influence. A reputation system's vulnerability to a Sybil attack depends on how cheaply identities can be generated, the degree to which the reputation system accepts inputs from entities that do not have a chain of trust linking them to a trusted entity, and whether the reputation system treats all entities identically. Holz et al. adopt this technique [52] to access the Storm botnet and measure a series of features. Davis et al. [53], using previous studies, developed a technique to generate false nodes in the Storm botnet capable of sending unconnected commands to disrupt and neutralize the C&C channel.

Constantinou and Mavrommatis [168] worked on a novel approach for P2P traffic identification that relies on the fundamental characteristics of P2P protocols instead of application-specific details. These characteristics include large network diameters and many entities acting as both clients and servers.

Yen and Reiter [175] have developed a methodology capable of differentiating hosts with P2P applications and those with P2P bots. This methodology is based on analyzing network flows generated by Argus and seeks characteristics related to flow volume, connection time in the P2P network, and host behavior (human-driven and machine-driven). The methodology can identify the Storm botnet in 87.5% of its occurrence and the Nugache botnet in 34%.

3.2.7. Multipurpose techniques

With the evolution of defensive techniques and to detect generic botnets, new approaches have been proposed to detect bots that use different protocols and unknown botnets.

BotHunter [128] is a passive bot detection system that uses IDS dialog correlation to associate IDS events with bot infection models. As BotHunter aims to detect bot behavior at the network level, stealthy bots can dodge detection by evading event timing correlation or conducting local attacks (e.g., deleting files) without any networking activities. The authors model the botnet infection life cycle considering the following activities: target scanning, infection exploit, binary download and execution, command-and-control channel establishment, and outbound scanning. The mechanism then detects botnets employing IDS-driven dialog correlation according to the bot infection life cycle model. Malware not conforming to this model seemingly go undetected. BotHunter employs Snort [109] with various malware extensions to raise an alarm when a sufficient subset of these is detected.

A study [129] analyzes NetFlow flows to generate visual representations of network traffic and help identify malicious activity related events. The view is appropriate in situations where human intelligence and knowledge are to be combined with automated methods. According to the authors, the alerts generated by the various detection tools and sophisticated attacks require a symbiosis between the detection algorithms and human analysis to tackle new opponents. Two visual representations of the data flow are compared: a TreeMap visualization of local network hosts, which are linked through hierarchical edge bundles with the external hosts, and a graph representation using a

force-directed layout to visualize the host communication pattern structures.

Botsniffer [49] uses network-based anomaly detection techniques designed especially for detecting IRC and HTTP botnets in a local area network. Botsniffer observes that bots within the same botnet likely reveal strong similarities in their responses and activities, i.e., scanning and sending spam emails, thus sharing common communication contents. It uses a detection method called spatial-temporal correlation and assumes that all botnets, unlike humans, tend to communicate in a highly synchronized fashion. Botsniffer shares similar concepts with BotGAD [2] in capturing synchronized botnet communication. Differing from BotGAD, Botsniffer performs string matching to detect similar responses from botnets. Nevertheless, botnets may encrypt their communication traffic or inject random noise packets to evade detection.

Botminer [77] is an approach that applies data mining techniques for botnet C&C traffic detection. Botminer improves the previously designed approach called Botsniffer [49]. Botminer clusters similar communication and malicious traffic. It then performs cross-cluster correlation to identify hosts that share both similar communication and malicious activity patterns. Botminer is an advanced botnet detection tool independent of botnet protocol and structure. Botminer can detect real-world botnets, including IRC-based, HTTP-based, and P2P botnets with a low false positive rate.

Botsniffer [49] and Botminer [77] are two botnet detection systems that employ vertical and horizontal correlation to perform a spatio-temporal group analysis across multiple hosts. Vertical correlation means that the network-based detection focuses on individual or single-bot infections. The referring software checks the network traffic using precast patterns for communications between the infected system and the C&C server or other relevant activities [108]. This procedure has the same disadvantage as the signature-based approach because botnet traffic cannot be detected without patterns or signatures.

In contrast to vertical correlation, horizontal correlation tries to detect two or more infected systems in the network. The detection mechanism searches for analogies in the network traffic, e.g., the same C&C server [108]. The key problem with this technique is that individual or different bots cannot be detected inside a network because there may be no affinity between them. Two different bots can thus remain unnoticed in the network.

Entropy and machine learning approaches are supposedly capable of detecting chat bots (not C&C botnet), including IRC, Yahoo! Chats, and MSN [130]. The two classifiers complement each other in chat bot detection. The entropy-based classifier can more accurately detect unknown chat bots, whereas the machine learning classifier is faster at detecting known chat bots. The work has a limitation in that it requires observing many chat messages before making a decision, so it is not suitable for detecting botnets with infrequent C&C interactions.

Yen and Reiter proposed TAMD (Traffic Aggregation for Malware Detection) [41], a system that detects centralized botnets by aggregating traffic with the same external destination, similar payload and that involves internal hosts with similar operating systems. The tool can identify candidate groups of infected computers within the network by analyzing communication flows that share common characteristics. One drawback of this technique is that it uses payload analysis for detection; botmasters could randomly generate payloads to evade the mechanism.

AsSadhan et al. [5] assumes that a bot has a repeated pattern behavior and explores this behavior, looking for periodic components in C&C traffic. It uses periodograms to study the periodic behavior and applies Walker's large sample test to detect whether the traffic has a significant periodic component. If it does, then it is identified as bot traffic. This test is independent of the structure and communication protocol used in the botnet, and it does not require any knowledge of a certain botnet behavior. To confirm the results, the author tests two variants of botnet C&C communication traffic generated by SLINGbot [176] and shows that the traffic in both variants exhibits periodic behavior.

Wurzinger et al. [108] propose an automatic method to generate a network-level botnet signature (model) of a given bot binary based on the botnet command-response pattern. They aim to generate detection models by observing bot behaviors captured in the wild, launching a bot in a controlled environment and recording its network activity (traces). The work can then identify points in a network trace that likely correlate with command-response activities. Detection models are generated for 18 different bot families: 16 controlled via IRC, one via HTTP (Kraken), and one via a P2P network (Storm Worm).

Giroire et al. [131] propose a method to identify suspicious C&C destinations by tracking the persistence of new connections that are not yet whitelisted. They introduced the notion of "destination traffic atoms". A destination atom is an aggregation of destinations intended to capture the "service" the user seeks. They compute the "persistence", which measures temporal regularity for individual destination atoms. Very persistent destination atoms are added to a host whitelist during a training period. With this whitelist in place, the method tracked the persistence of new destination atoms to identify suspicious C&C destinations. Destinations with high enough persistence are flagged as potential C&C endpoints. This approach aims to capture regular patterns to uncover C&C activity for the bot classes that employ a high degree of centralization in their infrastructure and where the communication channel lasts for an extended period. The main drawback is the need to track persistence over multiple timescales simultaneously.

Previous research [104] presents Botzilla, a system that uses vertical correlation to detect individual machines that behave like infected ones. When the malicious software contacts its maintainer by a process called "phoning home", signatures are automatically generated from monitored malware traffic, without human intervention or full network payload examination. The signature could be generated even if the malware is observed on a single infected host. The analysis is limited to the first bytes of network flows and attains sufficient detection accuracy.

3.3. Evasion techniques

Bot detection remains a challenging task because bot developers continuously adopt advanced techniques to make bots stealthier [45]. Stealthy malware, e.g., botnets and spyware, are hard to detect because their activities are subtle and do not disrupt the network, in contrast to DDoS attacks and aggressive worms.

Bots have become increasingly sophisticated, so evasion techniques have been developed to deceive detection mechanisms allowing botnets to have long operating times [32]. Stinson and Mtichell [177] propose a systematic framework for evaluating the evasion ability and fitness of a detection method. To be effective, a technique must consider two costs: implementation complexity and effect on botnet utility. Implementation complexity is related to how easily bot developers can change source code to evade detection. If the complexity is high, the evasion technique is not interesting.

Botnet usefulness relates to its functionality. If a mechanism to evade botnet detection no longer performs its function, its usefulness is reduced and it becomes less effective. The evasion technique is thus discarded and a new one should be developed, according to the level of complexity to justify the effort.

Several different evasion techniques are commonly employed, including tunneling through HTTP, ICMP or VoIP protocols [89,178], IPv6 tunneling [179], fast-flux service networks (FFSN), changes in statistical patterns, using dynamic DNS entries, encrypted traffic, assigning different tasks to bots in the same network, and randomizing bot communication patterns [32,77,146]. Developing new evasion techniques leads to developing new detection techniques, creating a conflict between attackers and defenders.

Initial detection techniques were based on payload inspection [110], looking for suspicious messages between bots and botmasters. To defeat these techniques, bots evolved and employed a cypher algorithm. Payload inspection is thus no longer effective.

As stated in Section 2.2, the first bots used an IRC protocol in a C&C channel because it facilitates developing new bots. However, blocking IRC bots is a simple task because IRC is not a common protocol used in enterprise networks; it can be prohibited, thus disrupting the botnet. To evade disruption, botmasters developed new bots that use tunnels through protocols commonly allowed in enterprise networks, e.g., HTTP or SIP.

A previous work [89] proposed a novel botnet model that exploits an overlay network like Skype to build a parasitic overlay, making it extremely difficult to track the botmaster and disrupt the botnet without damaging legitimate Skype users. This technique takes advantage of the reliability and capability that the Skype Network has to easily bypass firewalls and NAT devices. Bots have been developed using the Skype API, and messages exchanged between bots and the master flow through the network like application-legitimate messages. This makes the botnet traffic unrecognizable among the legitimate Skype traffic. Although that work used Skype, other evasion techniques can be used that employ and exploit other widely used applications.

Although clustering-based botnet detection schemes are effective, they can be evaded simply by randomizing communication patterns and assigning different roles to each bot. As Nugache indicates [48], the studied bot uses random high-numbered ports. Each peer chooses its own randomly generated high-numbered port on which to listen, ranging from 1025 to 65,535, and it could evade detection in most cases. In the same way, Storm picks a random high port for communication and advertises that port in every packet that it sends.

3.3.1. Fast-flux service networks – FFSNs

Most recent botnets use fast-flux service networks (FFSNs) [2,12,77,180,181] as their C&C mechanism. Fast flux is a DNS technique used to hide phishing and malware delivery sites behind an ever-changing network of compromised hosts acting as proxies. It can also refer to combining P2P networking, distributed C&C, web-based load balancing and proxy redirection used to make malware networks more resistant against discovery and counter-measures.

To provide anonymity, bots first connect to a compromised host, which serves as a proxy, to forward bot commands to the real C&C server and send responses from the C&C server to bots. In this technique, DNS records of a real website indicate computers in a fast-flux network. Fast-flux service networks combine round-robin IP addresses and very short Time-To-Live (TTL) values for any given particular DNS Resource Record (RR) to distribute bot commands to many compromised hosts. To simultaneously manage content availability for thousands of domains on a single host, nodes in fast-flux networks always host both DNS and HTTP services [17]. The Storm Worm [182] is a recent malware variant that uses this technique.

Perdisci et al. [147] proposes a passive approach for detecting and tracking malicious flux service networks. This detection system is based on passively analyzing recursive DNS (RDNS) traffic traces collected from multiple large networks. It can detect malicious FFSNs in-the-wild, i.e., as they are accessed by users who have fallen victim to malicious content advertised through a blog, instant messaging, social networking, and email spam.

Huang et al. [155] proposed a real-time FFSN detection system named Spatial Snapshot Fast Flux Detection system (SSFD). SSFD can reveal FFSNs by capturing the geographic traffic patterns of hosts and mapping the IP address of a DNS response in a geographic coordinate system. It uses spatial distribution estimation to estimate the uniform geographic distribution of infected hosts. Spatial service relationship evaluation is used to improve the misclassification between FFSN and Content Distribution Networks.

4. Defense techniques

When a botnet is detected, the next step is to either stop the bot or paralyze the whole botnet. Shutting down a single bot is not enough to solve the problem because there are many other infected machines in the network. More effective mechanisms are related to disabling the network as a whole [1,30].

Defense techniques against bots focus on two main activities: propagation and bot communication. Combating bot/worm spread directly affects the number of compromised machines in a botnet, thus reducing the network power and utility to the botmaster. Conversely, considering already infected machines, another form of defense is to stop the communication between bots and C&C servers. By disarticulating the communication, the botmaster cannot send commands or perform malicious activities [30].

The two defense approaches mentioned above cover three main areas: prevention, treatment and containment [183]. For the defense to be effective, there must be actions performed by users, network administrators, and Internet Service Providers (ISPs).

For propagation, the most important features for worm spreading are the number of vulnerable machines, infection duration, and infection rate [183]. Prevention techniques thus aim to reduce the vulnerable population, limit the worm spread, and reduce the botnet size.

Preventive actions are related to secure software development, updated system maintenance, vulnerability removal, antivirus program use, training, and user socioeconomic power [183]. Mechanisms employed in malware control are also effective in combating botnets, but that action alone is not enough to stop these threats: there is no guarantee that software does not have security weaknesses and users have sufficient knowledge and carefully use their network devices.

Treatment is related to disinfecting zombies to reduce the number of bots and perform system updates to reduce vulnerable hosts and the worm spread rate. However, the time required to release new mechanisms and updates to disinfect compromised machines and execute tests is limited by the human time scale, which sometimes does not react quickly enough to combat malicious botnet activities and worm spread epidemics. During the Code-Red epidemic, it took up to 16 days for most systems to be updated; even after six weeks, some systems had not yet been updated [183].

The containment phase can be split into two stages: botnet detection and response [32]. Botnet detection is generally performed by monitoring stations and/or monitoring the network, which were discussed in Section 3.

The response stage is related to using mechanisms to stop the traffic between bots and C&C servers and, as a more effective final action, server deactivation. This stage can be accomplished using automated mechanisms that integrate firewalls, content filters, address blacklists and routes to block communications between bots and malware spread to reduce or stop the infection, disrupt the botnet communication or definitely deactivate the C&C centers.

Analyzing a cyber-warfare scenario, the possibility of taking over botnet control may also be considered to conduct counter-attack actions. An example of this action is shown in previous work [63,184].

The main reasons for using containment and blocking strategies include the following:

- Blocking can be automated after bot detection, without depending on human action.
- These strategies may be implemented directly on the network without an overall solution to all hosts on the Internet.

Containment mechanisms must be considered in three aspects:

- Detection and reaction time.
- Strategy used to identify and contain bots.
- Solution topology and scope.

Moore et al. [183] analyzed aspects of malware spread and detection, which are important for reducing botnet growth rate. The detection and reaction times should be low enough to prevent high growth rates. These times vary according to the detection strategies used by malware searching for new victims, ranging from an average of 4 h to an average time of 2 min, in the worst case.

Moore et al. [183] proposed two basic strategies for malware detection: using blacklists and content filtering. The latter strategy proved to be more efficient, allowing a longer detection and reaction time with no infected population growth. A topology analysis showed that adopting these containment mechanisms must be held within a network-wide approach, with as much ISP participation as possible. Simulation results showed that if the 100 largest autonomous systems adopt a containment solution, there would be a decrease of approximately 90% in the number of machines infected in 24 h.

Such results corroborate the statement [26] that integrated malware control centers are required, similar to those for human diseases and viruses, and can become a more effective defense against malicious activity and malware spread.

Freiling et al. [30] proposes to prevent DDoS attacks using preventative action, i.e., disabling the botnet. This goal was achieved in three steps: infiltrate the C&C network, analyze the network in detail and disable the C&C channel. To accomplish this task, the researchers captured a bot, collected information for the infiltration, and infiltrated and disabled the network. Unfortunately, the approach presented is only applicable to IRC bots. The information needed to infiltrate the C&C network was

- DNS or IP address, and IRC server port.
- Password to connect to the IRC server (optional).
- Bot nickname and struct.
- IRC channel name and password, if applicable.

Two techniques were used to gather such information: a honeypot and the software developed by the authors called mwcollect. The first technique allows honeypots or honeynets to be compromised and join a botnet [1,7,30]. Behaving as normal “bots” in the botnet, these honeypot spies provide valuable information about the monitored botnet activities.

The software was designed to be infected and automate the information gathering tasks. It acts like a low-interactive honeypot, emulating services and vulnerabilities. Such emulations increase the likelihood that the botmaster

would realize that it is a monitoring device and evade the system.

After botnet infiltration, the authors propose a way to turn the botnet off: changing the DNS server configuration [30]. If the name resolution is redirected to private addresses, bots could not connect to the C&C server anymore, and the botnet itself would be disarticulated. This approach requires the cooperation of the DNS provider. With the involvement of the DNS provider, one may also take control of the entire network by simply configuring a C&C server.

With the help of Dynamic DNS providers, Dagon et al. [72] presented an effective botnet sinkhole that can change domain names by mapping a detected bot controller and indicating a monitoring machine. The monitor thus receives connection requests from most (if not all) bots in the botnet. Conceptually, the monitor becomes a hijacked bot controller, which is similar to a honeypot in its functionality.

5. New trends/platforms

Many traditional devices such as mobile phones, televisions, and cars incorporate new features, software, and applications. Internet access is almost mandatory, and connections to social networks, email, and web sites are guaranteed. Such devices are known as smart devices, such as smartphones and smartTVs. A realm of opportunities has arisen for the botmaster, allowing bots to migrate from conventional platforms like traditional computers to these new devices.

The installed base of smartphones and other mobile devices has grown to an attractive size. The devices run sophisticated operating systems that come with inevitable vulnerabilities. Fossi et al. [8] reports approximately 163 vulnerabilities in 2010. Porras et al. [185] analyzes the iKee.B (duh), an Apple iPhone bot client and one of the new malware applications emerging in this area. A previous study [186] demonstrates a proof-of-concept for an Android botnet, presenting an exploit that turns smartphones running the Android OS into SMS-spamming zombies. With full-day Internet connections for multiple mobile devices (e.g., BlackBerry, Windows Mobile, Symbian), malicious code may soon target those devices (a "mobile device botnet").

According to Plohmann et al. [22], another trend is improvements in attackers' security measures. The first botnets frequently have simple malicious code and infrastructure that builds the entire scenario, but this is changing. Attackers are becoming more cautious in every step they take. When they notice something strange, they use public key cryptography, distributed VPN, fast-flux, PHP encoding, JavaScript obfuscation, kernel packers, covert channels and auto-removal.

5.1. Socialbot network (SbN)

Another platform that has attracted botmasters attention is social networks. Botmasters have recently begun to exploit social network websites (e.g., Twitter.com) to behave as their C&C infrastructures. This procedure turns out to be stealthy because it is hard to distinguish the C&C activities from normal social networking traffic [187]. They have developed socialbots, which are computer programs that control social networks accounts and mimic real users to build a Socialbot Network (SbN), which is a group of adaptive socialbots orchestrated in a C&C fashion.

Koobface [188] is, perhaps, the first botnet on online social networks (OSNs). Koobface first compromises user accounts on OSNs and uses these accounts to promote a provocative message with a hyperlink. The link points to a phishing website that asks the user to install a Flash plugin which is, in fact, the Koobface executable. Koobface evolved from a SbN that does not rely on hijacked profiles. Koobface thus requires infecting many initial "zombie" machines through OSN-independent distribution channels.

In 2009, a user updated his Twitter account by reading updates via a RSS feed controlled by NazBot. This bot decoded messages, which were Base-64 encoded URLs, and downloaded malicious payload using trusted popular websites as a C&C server, such as Twitter and Facebook. It uses port 80 with legitimate HTTP requests and responses to become stealthy and not suspicious. To become invisible, the bot uses common features that are indistinguishable from normal traffic, such as RSS feeds [187].

A previous study [189] evaluates how vulnerable OSNs are to a large-scale socialbot infiltration. They design and build a Socialbot Network (SbN) and operate it on Facebook for approximately 8 weeks. They show that social networks can be infiltrated with a success rate of up to 80%; depending on users' privacy settings, a successful infiltration can result in privacy breaches where even more users' data are exposed. In practice, security defenses, such as the Facebook Immune System [190], are not effective enough in detecting or stopping a large-scale infiltration as it occurs.

Socialbots can be used to influence users in a social network [191] and can perform some activities, including posting messages and sending a connection request. They are designed to be stealthy and compromise the social graph of a targeted OSN by infiltrating (i.e., connecting to) its users to reach an influential position. It can be then exploited to spread misinformation and propaganda to bias public opinion.

The Facebook Socialbot Network, NazBot and Koobface were first examples of Socialbot Network (SbN). With online social networks growing and more users depending on these networks, more cases are expected to appear in the next years making OSN very attractive for botmasters.

5.2. Mini-botnets

Large-scale botnets, e.g., Srizbi, Kraken/Bobax, Rustock, gained notoriety for their sizes and malicious activities they are capable of performing, including sending spam or performing DDoS attacks. However, a new type of botnets has emerged, called mini-botnets.

Mini-botnets are small-scale and highly specialized botnets [192]. They are mainly used for information thefts, either personal or corporative, which have high financial or political value. Unlike large-scale botnets, mini-botnets

Table 4
Botnets × mini-botnets.

Feature	Botnet	Mini-botnet
Owner/Intention	Botmaster/Create generic botnet	Individual/Create botnet to a specific purpose
Target	Generic	Specific (e.g., vulnerable machines in a given company network)
Goal	Generic	Specific (e.g., stealing information)

do not generate much information during attacks, but they preserve discretion by being silent and stealthy [1]. A study that tracked over 600 botnets for 500 days showed that 57% of the networks had fewer than 100 elements [193]. Table 4 presents a comparative summary of mini-botnets and generic botnets.

Vogt et al. present another use of mini-botnets [194], describing a new botnet architecture called the super-botnet. Their architecture suggests that rather than having one large botnet, the botnet contains many smaller botnets with predefined sizes. The group of smaller botnets routes commands to each other and can collectively achieve the same results as a large botnet but with increased resilience.

The malware employed by mini-botnets is automatically generated by specialized software, generating codes that perform specific activities and indicate specific targets. Some software used include Zeus [195,196], SpyEye [197] and Butterfly [198].

Among the various activities that can be set for the new bot, it is usually programmed to steal the following information [192,195]:

- Credentials for access to FTP and POP services.
- X. 509 certificates.
- HTTP cookies.
- Intercept HTTP forms (ability to perform man-in-the-middle attack).

Over 3000 bot variants have been identified [196], with an estimation of approximately 3.6 million infected machines in the United States [192]. In February 2011, a C&C server was turned off. During the action to disrupt the botnet, 95% of the requests to the C&C server were from Mexico, although other requests came from other nations [199,200].

New techniques have emerged in the field. The SpyEye botnet, besides being a successor of Zeus, is also considered a competitor: when it detects that Zeus is installed in the machine, Zeus is deactivated [192,201].

Such software packages come with two basic programs: one to generate malware and another to implement the C&C server. The package thus allows people with low technical knowledge to customize their networks as they must only set up the C&C server address and define the activities they want to accomplish.

6. Challenges

Several challenges surround the botnet study. Botnets are a global phenomena; botmasters have at their disposal thousands of vulnerable hosts in domains around the world. Unfortunately, researchers do not have the same facility to access hosts in various domains for two main reasons [169]:

- First, most administrative domains consider any detailed information about their networks to be a business secret, and researchers (unlike botnet controllers) can only obtain such information through explicit collaboration agreements, which must be manually negotiated with each domain.
- Second, network traces, the most detailed and potentially useful type of information, may contain sensitive information and are thus treated like information plutonium: something to be mined only when absolutely necessary, carefully controlled, and never shared with outsiders or even others within the same organization.

The above problems have created a great disadvantage for researchers who ultimately use traces of academic networks where they pursue their research or generate botnet synthetic traces for their experiments.

Academic networks often do not reflect the reality of heterogeneous networks and can camouflage many important aspects that are unknown to a network in a relatively safe and controlled environment. The performance of a detecting bot method can be overestimated when applied to a particular network scenario.

Synthetic traces generation may consider some approaches to model botnet behavior:

- Epidemiological models that attempt to compare malware and virus spreads in populations.
- Stochastic Activity Network (SAN) models [154,202], used to generate a set of interconnected states that the host follows after its infection, with each state transition probability defined in advance. Not all botnets follow a standard set of states, so this model may not be considered a universal solution.

Botnet emulators often use protocols in which bots are developed to try to obtain some behavioral features for study.

Another challenge [169] is the difficulty of estimating how much a novel detection technique enhances overall botnet detection. A quantitative comparison of existing works would be easier if there were a standard methodology or widely accepted benchmark for evaluating botnet detectors. No such methodology or benchmark exists, presumably due to the pervasive privacy concerns and the resulting difficulties for data sharing.

With the fast development of computing capabilities and Internet access (e.g., using WiFi, GPRS and 3G) for smartphones and other mobile devices, another threat has risen in the area: botnets in mobile devices. Research

is just beginning in this area. Traynor et al. [203] demonstrate the ability of a botnet comprising as few as 11,750 compromised mobile phones to degrade service in area-code-sized regions by 93%. The authors reveal bottlenecks in cellular networks and showed how an attack could make the network unavailable for an entire region. Other authors [186,204,205] describe a cellular botnet (Andbot, IBot) in detail, including a C&C channel, commands and spread models.

Another challenge raised by mobile devices is that although most ISPs are implementing security measures to protect their customers from infections, these devices do not use static locations (i.e., static addresses or addresses within a specific dynamic range) because they connect to unknown wireless networks, thus making the providers' efforts useless.

A further challenge for mobile platforms is the lack of certification for applications created by programmers and placed in repositories for users to download. It is not difficult to imagine that a backdoor or vulnerability can be deployed in such applications, aiming to disseminate a botnet. The situation becomes even more alarming with the advent of mobile payment systems that aim to replace credit cards.

6.1. Botnet mitigation

Botnets will continue to grow unless effective actions are taken to mitigate both technical and non-technical factors that allow such threats to exist. Non-technical factors are related to distributed environment, legal issues and low user awareness. Technical factors are related to insecure software, passive ISPs and improper ingress and egress filtering.

Botnets are widespread in a distributed environment, so each botnet may involve several countries. Botnet activity could be legally restrained according to a specific local law issued by a nation. Agreements between countries are thus needed to prosecute cyber-crime in a consistent and coordinated way. Steps should also be taken to raise awareness among political decision-makers about the severity of the botnet problem.

Concurrently, there is no clear sign of a rise in user awareness. People still do not know that their computers are infected, and if they do, they often do not know what to do about it. User education about botnet threats should also be extended.

As Internet users do not know how to harden their devices' security, software vendors (e.g., operating systems and software applications) should make strenuous efforts to increase their products' security and improve the update and patch management process.

ISPs are among the most important actors in botnet mitigation, as they can detect and block botnet communication. They could apply proper ingress and egress filtering, blocking inbound and outbound malicious users connections (e.g., spam, malicious code, and attacks), thus blocking C&C communication and disrupting botnets. Such filters often conflict with blocking normal user traffic, and thus ISPs would need to inspect the user traffic, which could lead to privacy issues.

As most ISPs do not block bot communications, another challenge is to develop algorithms to hijack botnets [63]. Hijacking could allow law enforcement agencies to closely monitor botnets. If such monitoring indicates high botnet activity, switching off the bot host may be considered, assuming that privacy issues and potential side effects are resolved.

7. Conclusion

A great deal of recent research has examined botnets; despite some real advances, few results have been adopted and implemented in real network scenarios. This survey aims to first identify this serious security problem, review and classify recent research works and results, and finally indicate some challenges posed to future work on the topic.

Botnets have played an important role as a major security threats on the Internet. It is estimated that over 80% of spam messages originate from these overlay networks. Attackers may take control of an army of hosts (bots) using a C&C channel to perform illegal activities (e.g., click frauds, email spam, DDoS attacks). Botnets' architectures, life cycles, mechanisms, protocols, destructive power and utility are discussed in greater detail above.

The first necessary step towards combating botnet threats is developing efficient detection techniques. As discussed above, such techniques can be classified into two categories: honeynet-based and intrusion detection system (IDS). The first aims to collect information from bots to understand their behavior and how to detect them. The second employs IDS approaches and can be further subdivided into signature- and anomaly-based techniques. Most work in botnet detection considers anomaly-based techniques.

After detection, the next important step is to derive ways to dismantle botnets' infrastructures and disrupt their operations. The most common techniques applied to accomplish this task focus on breaking the C&C channel and preventing the botmaster from continuing to send commands to the botnet. Several different proposals are also discussed above.

From the botmaster's perspective, evasion techniques are also being developed that make current botnets stealthy, fast changing and difficult to track. The latest moves also migrate botnets to operate on new platforms, including smartphones, tablets and other mobile devices. Another innovative action is the emergence of mini-botnets, which are small-scale networks specialized in information theft that, due to their small sizes, are even harder to detect.

Many challenges remain in this area. One main issue researchers face is the difficulty of testing their proposals in a real scenario or using real data. Some initiatives, e.g., creating trace repositories, have been implemented with some success, but data access is sometimes controlled or limited to some particular cases.

Because botnets may span the globe, it is paramount that different network players and administrative entities take cooperative actions. Research challenges also include designing distributed, collaborative and Inter-AS detection

systems as well as global botnet countermeasures. It is also important to discuss legal international issues and establish global policies to thoroughly combat the botnet threat.

References

[1] E. Cooke, F. Jahanian, D. McPherson, The zombie roundup: understanding, detecting, and disrupting botnets, in: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop, USENIX Association, Berkeley, CA, USA, 2005, p. 6.

[2] H. Choi, H. Lee, H. Kim, BotGAD: detecting botnets by capturing group activities in network traffic, in: Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE, COMSWARE '09, ACM, New York, NY, USA, 2009, pp. 21–28.

[3] J.M. Ceron, L.Z. Granville, L.M.R. Tarouco, Uma arquitetura baseada em assinaturas para mitigaç˜ao de botnets, in: X Simpósio Brasileiro em Seguranç˜a da Informaç˜ao e de Sistemas Computacionais (SBSeg), pp. 105–118.

[4] W. Sturgeon, Net pioneer predicts overwhelming botnet surge, 2007 <http://news.cnet.com/2100-73483-6154221.html>.

[5] B. AsSadhan, J. Moura, D. Lapsley, C. Jones, W. Strayer, Detecting botnets using command and control traffic, in: Eighth IEEE International Symposium on Network Computing and Applications, 2009. NCA, 2009, pp. 156–162.

[6] N. Ianelli, A. Hackworth, Botnets as a Vehicle for Online Crime – Coordination Center, CERT cMellon University, Carnegie CERT, 2005.

[7] P. Bacher, T. Holz, M. Kotter, G. Wicherski, Know Your Enemy: Tracking Botnets (using honeynets to learn more about bots), Technical Report, The Honeynet Project, 2008.

[8] M. Fossi, G.Y. Egan, K. Haley, E. Johnson, T. Mack, T. Adams, J. Blackbird, M.K. Low, D. Mazurek, D. McKinney, P. Wood, Symantec Internet Security Threat Report – Trends for 2010, Technical Report Volume 16, Symantec, 2011.

[9] G. Stringhini, T. Holz, B. Stone-Gross, C. Kruegel, G. Vigna, BOTMAGNIFIER: locating spambots on the internet, in: Proceedings of the 20th USENIX conference on Security, SEC'11, USENIX Association, Berkeley, CA, USA, 2011, p. 28.

[10] T.M.S. Labs, M86 Security, Spam Statistics, 2011 <http://www.m86security.com/labs/spamstatistics.asp>.

[11] J.M. Bauer, M.J.G. van Eeten, Y. Wu, ITU Study on the Financial Aspects of Network Security: Malware and Spam, Technical Report, ITU -International Telecommunication Union, 2008.

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

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

[14] M. Bailey, E. Cooke, F. Jahanian, Y. Xu, M. Karir, A survey of botnet technology and defenses, in: Conference for Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology, pp. 299–304.

[15] Y. Shin, E. Im, A survey of botnet: consequences, defenses and challenges basic knowledge of botnet, Challenges (2009).

[16] C. Li, W. Jiang, X. Zou, Botnet: survey and case study, in: Fourth International Conference on Innovative Computing, Information and Control (ICICIC), 2009, pp. 1184–1187.

[17] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng, J. Zhang, Botnet: classification, attacks, detection, tracing, and preventive measures, EURASIP Journal of Wireless Communication Networks 2009 (2009) 91–911.

[18] Symantec, Spybot worm, 2003 <http://www.symantec.com/securityresponse/writeup.jsp?docid=2003-053013-5943-99>.

[19] T. Micro, Worm AgoBot, 2004 <http://about-threats.trendmicro.com/ArchiveMalware.aspx?language=us&name=WORMAGOBOT.XE>.

[20] T. Micro, Worm SDBot, 2003 <http://about-threats.trendmicro.com/ArchiveMalware.aspx?language=us&name=WORMSDBOT.AZ>.

[21] G. Macesanu, T. Codas, C. Suliman, B. Tarnauca, Development of GTBoT, a high performance and modular indoor robot, in: IEEE International Conference on Automation Quality and Testing Robotics (AQTR), vol. 1, 2010, pp. 1–6.

[22] D. Plohmann, E. Gerhards-Padilla, F. Leder, Botnets: Detection, Measurement, Disinfection & Defence, Technical Report, The European Network and Information Security Agency (ENISA), 2011.

[23] R. Puri, Bots & Botnet: An Overview, SANS Institute InfoSec Reading Room, 2003.

[24] S. Basudev, A. Gairola, Botnet: An Overview, CERT-In White Paper CIWP-2005-05, 2005.

[25] M.A Rajab, J. Zarfoss, F. Monrose, A. Terzis, A multifaceted approach to understanding the botnet phenomenon, in: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC'06, ACM, New York, NY, USA, 2006, pp. 41–52.

[26] S. Staniford, V. Paxson, N. Weaver, How to own the internet in your spare time, in: Proceedings of the 11th USENIX Security Symposium, USENIX Association, Berkeley, CA, USA, 2002. p. 149 and 167.

[27] D. Dagon, G. Gu, C. Zou, J. Grizzard, S. Dwivedi, W. Lee, R. Lipton, A taxonomy of botnets, in: In: Proceedings of CAIDA DNS-OARC Workshop, 2005.

[28] D. Dagon, G. Gu, C.P. Lee, W. Lee, A taxonomy of botnet structures, in: Twenty-Third Annual Computer Security Applications Conference, ACSAC 2007, 2007, pp. 325–339.

[29] M.A Rajab, J. Zarfoss, F. Monrose, A. Terzis, My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging, in: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, USENIX Association, Berkeley, CA, USA, 2007, p. 5.

[30] F.C Freiling, T. Holz, G. Wicherski, Botnet tracking: exploring a root-cause methodology to prevent distributed denial-of-service attacks, in: S. de Capitani di Vimercati, P. Syverson, D. Gollmann (Eds.), Computer Security ESORICS 2005, Lecture Notes in Computer Science, vol. 3679, Springer, Berlin/Heidelberg, 2005, pp. 319–335. 10.1007/11555827 19.

[31] B. News, Hacker threats to bookies probed, 2004 <http://news.bbc.co.uk/2/hi/technology/3513849.stm>.

[32] T. Micro, Taxonomy of Botnet Threats, Technical Report, Trend Micro White Paper, 2006.

[33] C. Schiller, J. Binkley, Botnets: The Killer Web Applications, Syngress Publishing, 2007.

[34] N. Paxton, G. Ahn, B. Chu, Towards practical framework for collecting and analyzing network-centric attacks, in: IEEE International Conference on Information Reuse and Integration, IRI 2007, 2007, pp. 73–78.

[35] P. Wang, S. Sparks, C.C Zou, An advanced hybrid peer-to-peer botnet, in: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, USENIX Association, Berkeley, CA, USA, 2007, p. 2.

[36] H. Zeidanloo, A. Manaf, Botnet command and control mechanisms, in: Second International Conference on Computer and Electrical Engineering ICCEE 09, vol. 1, 2009, pp. 564–568.

[37] P. Marupally, V. Paruchuri, Comparative analysis and evaluation of botnet command and control models, in: 24th IEEE International Conference on Advanced Information Networking and Applications (AINAs), 2010, pp. 82–89.

[38] EggHeads, EggHeads.org-eggdrop development, 1993 <http://eggheads.org/>.

[39] J.B Grizzard, V. Sharma, C. Nunnery, B.B Kang, D. Dagon, Peer-to-peer botnets: overview and case study, in: Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets, USENIX Association, Berkeley, CA, USA, 2007, p. 1.

[40] Dumbledore, Well Known Bot Families, 2001 <http://dumbledore.hubpages.com>.

[41] T. Yen, M.K Reiter, Traffic aggregation for malware detection, in: Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA'08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 207–227.

[42] Symantec, W32.Gaobot.CEZ, 2002 <http://www.symantec.com>.

[43] R. Schoof, R. Koning, Detecting Peer-to-Peer Botnets, Technical Report 1, University of Amsterdam, 2007.

[44] Symantec, Messagelabs Intelligence, in: Security Response, Symantec, 2010.

[45] L. Liu, S. Chen, G. Yan, Z. Zhang, BotTracer: Execution-Based Bot-Like Malware Detection, in: T. Wu, C. Lei, V. Rijmen, D. Lee (Eds.), Information Security, Lecture Notes in Computer Science, vol. 5222, Springer, Berlin/Heidelberg, 2008, pp. 97–113. 10.1007/978-3-540-85886-7 7.

[46] G. Keizer, Top botnets control 1 m hijacked computers, 2008 <http://www.computerworld.com>.

[47] T. Wilson, Competition may be driving surge in botnets, spam, 2008 <http://www.darkreading.com>.

[48] S. Stover, D. Dittrich, J. Hernandez, S. Dietrich, Analysis of the storm and nugache: P2p is here, in: Proceedings of the 4th USENIX Workshop on Cyber Security Experimentation and Test (CSET'11), USENIX Association, 2007.
[49] G. Gu, J. Zhang, W. Lee, BotSniffer – detecting botnet command and control channels in network traffic, in: 15th Annual Network & Distributed System Security Symposium, The Internet Society (ISOC), San Diego, 2008.
[50] Spam botnets to watch in 2009, 2001 <http://www.secureworks.com>.
[51] C. Miller, The Rustock Botnet Spams Again, 2008.
[52] T. Holz, M. Steiner, F. Dahl, E. Biersack, F. Freiling, Measurements 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, USENIX Association, Berkeley, CA, USA, 2008.
[53] C. Davis, J. Fernandez, S. Neville, J. McHugh, Sybil attacks as a mitigation strategy against the storm botnet, in: 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008, 2008, pp. 32–40.
[54] S. Works, Pushdo – Analysis of a Modern Malware Distribution System, 2008 <http://www.secureworks.com>.
[55] J. Kirk, Spammers Regaining Control Over Srizbi Botnet, 2008 <http://www.pcworld.com>.
[56] E. Messmer, America's 10 Most Wanted Botnets, 2009 <http://www.networkworld.com>.
[57] Symantec, Zeus: King of the bots, in: Security Response, Symantec, 2008.
[58] K. Stevens, D. Jackson, Zeus Banking Trojan Report, 2010 <http://www.secureworks.com>.
[59] J. Hruska, New Mega-d Menace Muscles Storm Worm Aside, 2008 <http://arstechnica.com>.
[60] R. Borgaonkar, An analysis of the asprox botnet, in: Fourth International Conference on Emerging Security Information Systems and Technologies (SECURWARE), 2010, pp. 148–153.
[61] K.J. Higgins, New Massive Botnet Twice the Size of Storm, 2008 <http://www.darkreading.com>.
[62] A. Moscaritolo, Kraken botnet re-emerges 318,000 nodes strong, 2010 <http://www.scmagazineus.com>.
[63] B. Stone-Gross, M. Cova, B. Gilbert, R. Kemmerer, C. Kruegel, G. Vigna, Analysis of a botnet takeover, Security Privacy, IEEE 9 (2011) 64–72.
[64] Experts bicker over conficker numbers, 2001 <http://news.techworld.com>.
[65] Symantec, The downaduo codex, in: Security Response, Symantec, 2009.
[66] D.-I. Jang, M. Kim, H.-C. Jung, B. Noh, Analysis of HTTP2P botnet: case study waledac, in: IEEE 9th Malaysia International Conference on Communications (MICC), 2009, pp. 409–412.
[67] G. Sinclair, C. Nunnery, B. Kang, The waledac protocol: the how and why, in: 4th International Conference on Malicious and Unwanted Software (MALWARE), 2009, pp. 69–77.
[68] Tdl-4 top bot, 2011 <http://www.securelist.com>.
[69] H.R. Zeidanloo, M.J. Shooshtari, P.V. Amoli, M. Safari, M. Zamani, A taxonomy of botnet detection techniques, in: 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 2, 2010, pp. 158–162.
[70] T. Holz, M. Engelberth, F. Freiling, Learning more about the underground economy: a case-study of keyloggers and dropzones, in: M. Backes, P. Ning (Eds.), Computer Security – ESORICS 2009, Lecture Notes in Computer Science, vol. 5789, Springer, Berlin/Heidelberg, 2009, pp. 1–18. 10.1007/978-3-642-04444-1 1.
[71] B. Stone-Gross, T. Holz, G. Stringhini, G. Vigna, The underground economy of spam: a botmaster's perspective of coordinating large-scale spam campaigns, in: 4th Usenix Workshop on Large-Scale Exploits and Emergent Threats.
[72] D. Dagon, C. Zou, W. Lee, Modeling botnet propagation using time zones, in: Proceedings of the 13th Network and Distributed System Security Symposium NDSS.
[73] G.P. Schaffer, Worms and viruses and botnets, oh my! Rational responses to emerging internet threats, Security Privacy, IEEE 4 (2006) 52–58.
[74] A. . Barsamian, Network characterization for botnet detection using statistical-behavioral methods, Ph.D. thesis, Thayer School of Engineering – Dartmouth College, Hanover, New Hampshire, 2009.
[75] P. Mockapetris, RFC 1034-domain names-concepts and facilities, 1987 <http://tools.ietf.org/html/rfc1034>.
[76] A. Ramachandran, N. Feamster, D. Dagon, Revealing botnet membership using DNSBL counter-intelligence, Proceedings of the 2nd Conference on Steps to Reducing Unwanted Traffic on the Internet, vol. 2, USENIX Association, Berkeley, CA, USA, 2006, p. 8.
[77] G. Gu, R. Perdisci, J. Zhang, W. Lee, BotMiner: clustering analysis of network traffic for protocol-and structure-independent botnet detection, in: Proceedings of the 17th Conference on Security Symposium, USENIX Association, Berkeley, CA, USA, 2008, pp. 139–154.
[78] C. Kalt, RFC 2810-internet relay chat: architecture, 2000 <http://tools.ietf.org/html/rfc2810>.
[79] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, RFC 2616-hypertext transfer protocol – HTTP/1.1, 1999 <http://tools.ietf.org/html/rfc2616>.
[80] Microsoft, What is a Botnet? Security Intelligence Report, Microsoft, 2010.
[81] ShadowServer, ShadowServer, 2004 <http://shadowserver.org>.
[82] J. Goebel, T. Holz, Rishi: identify bot contaminated hosts by IRC nickname evaluation, in: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, USENIX Association, Berkeley, CA, USA, 2007, p. 8.
[83] V. Paxson, Bro: a system for detecting network intruders in real-time, Computer Networks 31 (1999) 2435–2463.
[84] V. Yegneswaran, P. Porras, H. Saidi, M. Sharif, A. Narayanan, SRI honeynet and BotHunter malware analysis, 2005 <http://www.cyber-ta.org/releases/malware-analysis/public/>.
[85] M. Jelasity, V. Bilicki, Towards automated detection of peer-to-peer botnets: on the limits of local approaches, in: USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09), USENIX Association, Boston, MA, 2009.
[86] T. Risse, P. Knezevic, A. Wombacher, P2p evolution: from file-sharing to decentralized workflows, Information Technology 46 (2004) 193–199.
[87] S.A. Crosby, D.S. Wallach, An analysis of bittorrent?s two kademlia-based dhts, 2007.
[88] P. Maymounkov, D. Mazi'eres, Kademlia: a peer-to-peer information system based on the xor metric, in: Revised Papers from the First International Workshop on Peer-to-Peer Systems, IPTPS'01, Springer-Verlag, London, UK, 2002, pp. 53–65.
[89] A. Nappa, A. Fattori, M. Balduzzi, M. Dell'Amico, L. Cavallaro, Take a deep breath: a stealthy, resilient and cost-effective botnet using skype, in: Proceedings of the 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 81–100.
[90] D. Zhang, C. Zheng, H. Zhang, H. Yu, Identification and analysis of skype peer-to-peer traffic, in: Fifth International Conference on Internet and Web Applications and Services (ICIW), 2010, pp. 200–206.
[91] P. Wang, L. Wu, B. Aslam, C. Zou, A systematic study on Peer-to-Peer botnets, in: Proceedings of 18th International Conference on Computer Communications and Networks, ICCCN 2009, 2009, pp. 1–8.
[92] J. Yu, Z. Li, J. Hu, F. Liu, L. Zhou, Using simulation to characterize topology of peer to peer botnets, International Conference on Computer Modeling and Simulation 0 (2009) 78–83.
[93] S. Chang, L. Zhang, Y. Guan, T. Daniels, A framework for P2P botnets, in: WRI International Conference on Communications and Mobile Computing, CMC'09, vol. 3, 2009, pp. 594–599.
[94] N. Provos, A virtual honeypot framework, Proceedings of the 13th Conference on USENIX Security Symposium SSYM'04, vol. 13, USENIX Association, Berkeley, CA, USA, 2004, p. 1.
[95] B. McCarty, Botnets: big and bigger, Security Privacy, IEEE 1 (2003) 87–90.
[96] A. Ramachandran, N. Feamster, Understanding the network-level behavior of spammers, SIGCOMM Computer Communication Review 36 (2006) 291–302.
[97] P. Barford, V. Yegneswaran, An inside look at botnets, in: M. Christodorescu, S. Jha, D. Maughan, D. Song, C. Wang (Eds.), Malware Detection, Advances in Information Security, vol. 27, Springer, US, 2007, pp. 171–191. 10.1007/9780-387-44599-1 8.
[98] J. Oberheide, M. Karir, Z.M. Mao, Characterizing dark DNS behavior, in: Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 140–156.
[99] Z. Li, A. Goyal, Y. Chen, V. Paxson, Automating analysis of large-scale botnet probing events, in: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS'09, ACM, New York, NY, USA, 2009, pp. 11–22.

[100] B.B Kang, E. Chan-Tin, C.P Lee, J. Tyra, H.J Kang, C. Nunnery, Z. Wadler, G. Sinclair, N. Hopper, D. Dagon, Y. Kim, Towards complete node enumeration in a peer-to-peer botnet, in: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS'09, ACM, New York, NY, USA, 2009, pp. 23–34.
[101] M. Cremonini, M. Riccardi, The dorothy project: an open botnet analysis framework for automatic tracking and activity visualization, in: European Conference on Computer Network Defense (EC2ND), 2009, pp. 52–54.
[102] V. Pham, M. Dacier, Honeypot traces forensics: the observation viewpoint matters, in: Third International Conference on Network and System Security, NSS'09, 2009, pp. 365–372.
[103] M. Szymczyk, Detecting botnets in computer networks using multi-agent technology, in: Fourth International Conference on Dependability of Computer Systems, DepCos-RELCOMEX'09, 2009, pp. 192–201.
[104] K. Rieck, G. Schwenk, T. Limmer, T. Holz, P. Laskov, Botzilla: detecting the "phoning home" of malicious software, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC'10, ACM, New York, NY, USA, 2010, pp. 1978–1984.
[105] J. Bethencourt, J. Franklin, M. Vernon, Mapping internet sensors with probe response attacks, Proceedings of the 14th conference on USENIX Security Symposium, vol. 14, USENIX Association, Berkeley, CA, USA, 2005, p. 13.
[106] C. Zou, R. Cunningham, Honeypot-Aware advanced botnet construction and maintenance, in: International Conference on Dependable Systems and Networks, DSN 2006, 2006, pp. 199–208.
[107] Y. Kugisaki, Y. Kasahara, Y. Hori, K. Sakurai, Bot detection based on traffic analysis, in: The 2007 International Conference on Intelligent Pervasive Computing, IPC, 2007, pp. 303–306.
[108] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, E. Kirda, Automatically generating models for botnet detection, in: M. Backes, P. Ning (Eds.), Computer Security – ESORICS 2009, Lecture Notes in Computer Science, vol. 5789, Springer, Berlin/Heidelberg, 2009, pp. 232–249. 10.1007/978-3-642-04444-1 15.
[109] Snort, Snort 2006 <http://www.snort.org>.
[110] J.R Binkley, S. Singh, An algorithm for anomaly-based botnet detection, Proceedings of the 2nd Conference on Steps to Reducing Unwanted Traffic on the Internet, vol. 2, USENIX Association, Berkeley, CA, USA, 2006, p. 7.
[111] E. Stinson, J.C Mitchell, Characterizing bots' remote control behavior, in: Proceedings of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 89–108.
[112] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, W. Lee, Active botnet probing to identify obscure command and control channels, in: Computer Security Applications Conference, ACSAC'09, Annual, 2009, pp. 241–253.
[113] M. Masud, T. Al-khateeb, L. Khan, B. Thuraisingham, K. Hamlen, Flow-based identification of botnet traffic by mining multiple log files, in: First International Conference on Distributed Framework and Applications, DFmA 2008, 2008, pp. 200–206.
[114] K. Xu, D. Yao, Q. Ma, A. Crowell, Detecting infection onset with behavior-based policies, in: 5th International Conference on Network and System Security (NSS), 2011, pp. 57–64.
[115] A. Karasaridis, B. Rexroad, D. Hoeflin, Wide-scale botnet detection and characterization, in: Proceedings of the First conference on First Workshop on Hot Topics in Understanding Botnets, USENIX Association, Berkeley, CA, USA, 2007, p. 7.
[116] W. Strayer, D. Lapsely, R. Walsh, C. Livadas, Botnet detection based on network behavior, in: W. Lee, C. Wang, D. Dagon (Eds.), Botnet Detection, Advances in Information Security, vol. 36, Springer, US, 2008, pp. 1–24. 10.1007/978-0-387-68768-1 1.
[117] W. Lu, A. Ghorbani, Botnets detection based on IRC-Community, in: Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008, IEEE, pp. 1–5.
[118] R. Villamarin-Salomon, J. Brustoloni, Identifying botnets using anomaly detection techniques applied to DNS traffic, in: 5th IEEE Consumer Communications and Networking Conference, CCNC 2008, 2008, pp. 476–481.
[119] H. Husna, S. Phithakkitnukoon, S. Palla, R. Dantu, Behavior analysis of spam botnets, in: 3rd International Conference on Communication Systems Software and Middleware and Workshops, COMSWARE 2008, 2008, pp. 246–253.
[120] L. Zhuang, J. Dunagan, D.R Simon, H.J Wang, J.D Tygar, Characterizing botnets from email spam records, in: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, USENIX Association, Berkeley, CA, USA, 2008, pp. 2:1–2:9.
[121] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, E. Gillum, BotGraph: large scale spamming botnet detection, in: Proceedings of the 6th USENIX Symposium on NETWORKED SYSTEMS DESIGN and Implementation, NSDI'09, USENIX Association, Berkeley, CA, USA, 2009, pp. 321-334.
[122] J. P. John, A. Moshchuk, S.D. Gribble, A. Krishnamurthy, Studying spamming botnets using botlab, in: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09, USENIX Association, Berkeley, CA, USA, 2009, pp. 291–306.
[123] J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, X. Luo, Detecting stealthy P2P botnets using statistical traffic fingerprints, in: DNS 2011, IEEE Computer Society, Los Alamitos, CA, USA, 2011, pp. 121–132.
[124] D. Liu, Y. Li, Y. Hu, Z. Liang, A P2P-Botnet Detection Model and Algorithms Based on Network Streams Analysis, IEEE Computer Society, Changzhou, China, 2010. pp. 55–58.
[125] W.-H. Liao, C.-C. Chang, Peer to peer botnet detection using data mining scheme, in: International Conference on Internet Technology and Applications, IEEE Computer Society, 2010, pp. 1–4.
[126] X. Yu, X. Dong, G. Yu, Y. Qin, D. Yue, Y. Zhao, Online botnet detection based on incremental discrete Fourier transform, Journal of Networks 5 (2010).
[127] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, N. Borisov, Botgrep: finding p2p bots with structured graph analysis, in: Proceedings of the 19th USENIX Conference on Security, USENIX Security'10, USENIX Association, Berkeley, CA.USA, 2010, p. 7.
[128] G. Gu, P. Porras, V. Yegneswaran, M. Fong, W. Lee, BotHunter: detecting malware infection through IDS-driven dialog correlation, in: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, USENIX Association, Berkeley, CA, USA, 2007, pp. 12:1–12:16.
[129] F. Mansmann, F. Fischer, D.A Keim, S.C North, Visual support for analyzing network traffic and intrusion detection events using TreeMap and graph representations, in: Proceedings of the Symposium on Computer Human Interaction for the Management of Information Technology, CHiMiT 09, ACM, New York, NY, USA, 2009, pp. 3:19–3:28.
[130] S. Gianvecchio, M. Xie, Z. Wu, H. Wang, Measurement and classification of humans and bots in internet chat, in: Proceedings of the 17th Conference on Security Symposium, USENIX Association, Berkeley, CA, USA, 2008, p. 155.
[131] F. Giroire, J. Chandrashekar, N. Taft, E. Schooler, D. Papagiannaki, Exploiting temporal persistence to detect covert botnet channels, in: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, RAID '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 326–345.
[132] A. Lelli, Trojan. Hydraq, in: Symantec Security Response, Symantec, 2010.
[133] W. Chun-dong, L. Ting, W. Huai-bin, Botnet detection based on analysis of mail flow, in: 2nd International Conference on Biomedical Engineering and Informatics, BMEI '09, 2009, pp. 1–4.
[134] K. Kaemarungsi, N. Yoskamtorn, K. Jirawannakool, N. Sanglerdsinlapachai, C. Luangingkasut, Botnet statistical analysis tool for limited resource computer emergency response team, in: Fifth International Conference on IT Security Incident Management and IT Forensics, IMF'09, 2009, pp. 27–40.
[135] B. Wang, Z. Li, H. Tu, J. Ma, Measuring Peer-to-Peer botnets using control flow stability, in: International Conference on Availability, Reliability and Security, ARES'09, 2009, pp. 663–669.
[136] A. Shahrestani, M. Feily, R. Ahmad, S. Ramadass, Architecture for applying data mining and visualization on network flow for botnet traffic detection, in: International Conference on Computer Technology and Development, ICCTD '09, vol. 1, 2009, pp. 33–37.
[137] S. Mukosaka, H. Koike, Integrated visualization system for monitoring security in large-scale local area network, in: 6th International Asia-Pacific Symposium on Visualization, APVIS'07. 2007, 2007, pp. 41–44.
[138] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, G. Varghese, Network monitoring using traffic dispersion graphs (tdgs), in: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC '07, ACM, New York, NY, USA, 2007, pp. 315–320.
[139] J. Fran¸cois, S. Wang, R. State, T. Engel, BotTrack: tracking botnets using NetFlow and PageRank, in: Proceedings of the 10th International IFIP TC 6 Conference on Networking – Volume Part I, NETWORKING'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 1–14.

[140] N. Jiang, J. Cao, Y. Jin, L.E. Li, Z. Zhang, Identifying suspicious activities through DNS failure graph analysis, in: 18th IEEE International Conference on Network Protocols (ICNPs), 2010, pp. 144–153.
[141] J. Leonard, S. Xu, R. Sandhu, A first step towards characterizing stealthy botnets, in: International Conference on Availability, Reliability and Security, ARES'09, 2009, pp. 106–113.
[142] M.P Collins, M.K Reiter, Hit-list worm detection and bot identification in large networks using protocol graphs, in: Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection, RAID'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 276–295.
[143] D. Ha, G. Yan, S. Eidenbenz, H. Ngo, On the effectiveness of structural detection and defense against P2P-based botnets, in: IEEE/IFIP International Conference on Dependable Systems Networks, DSN'09, 2009, pp. 297–306.
[144] W. Wang, B. Fang, Z. Zhang, C. Li, A novel approach to detect IRC-Based botnets, in: International Conference on Networks Security, Wireless Communications and Trusted Computing, NSWCTC'09, vol. 1, 2009 pp. 408–411.
[145] Y. Zeng, G. Yan, S. Eidenbenz, K. Shin, Measuring the effectiveness of infrastructure-level detection of large-scale botnets, in: IEEE 19th International Workshop on Quality of Service (IWQoS), 2011, pp. 1–9.
[146] B. Coskun, S. Dietrich, N. Memon, Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts, in: Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC 10, ACM, New York, NY, USA, 2010.
[147] R. Perdisci, I. Corona, D. Dagon, W. Lee, Detecting malicious flux service networks through passive analysis of recursive DNS traces, in: Annual Computer Security Applications Conference, ACSAC '09, 2009, pp. 311–320.
[148] W. Lu, M. Tavallaee, A.A Ghorbani, Automatic discovery of botnet communities on large-scale communication networks, in: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS '09, ACM, New York, NY, USA, 2009, pp. 1–10.
[149] S. Chang, T.E Daniels, P2P botnet detection using behavior clustering & statistical tests, in: Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence, AISec'09, ACM, New York, NY, USA, 2009, pp. 23–30.
[150] F. Yu, Y. Xie, Q. Ke, Sbotminer: large scale search bot detection, in: Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM'10, ACM, New York, NY, USA, 2010, pp. 421–430.
[151] O. Thonnard, M. Dacier, A strategic analysis of spam botnets operations, in: Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, CEAS'11, ACM, New York, NY, USA, 2011, pp. 162–171.
[152] F. Chen, S. Ranjan, P. Tan, Detecting bots via incremental LS-SVM learning with dynamic feature adaptation, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11, ACM, New York, NY, USA, 2011, pp. 386–394.
[153] F. Sanchez, Z. Duan, Y. Dong, Blocking spam by separating end-user machines from legitimate mail server machines, in: Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, CEAS '11, ACM, New York, NY, USA, 2011, pp. 116–124.
[154] E. Van Ruitenbeek, W. Sanders, Modeling Peer-to-Peer botnets, in: Fifth International Conference on Quantitative Evaluation of Systems, QEST'08, 2008, pp. 307–316.
[155] S. Huang, C. Mao, H. Lee, Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection, in: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS '10, ACM, New York, NY, USA, 2010, pp. 101–111.
[156] J. Kang, J. Zhang, Application entropy theory to detect new peer-to-peer botnet with multi-chart CUSUM, in: Second International Symposium on Electronic Commerce and Security, ISECS'09, vol. 1, 2009, pp. 470–474.
[157] P. Salvador, A. Nogueira, U. Franca, R. Valadas, Framework for zombie detection using neural networks, in: Fourth International Conference on Internet Monitoring and Protection, ICIMP'09, 2009, pp. 14–20.
[158] W. Lu, M. Tavallaee, G. Rammidi, A. Ghorbani, BotCop: an online botnet traffic classifier, in: Seventh Annual Communication Networks and Services Research Conference, CNSR'09, 2009, pp. 70–77.
[159] X. Yu, X. Dong, G. Yu, Y. Qin, D. Yue, Y. Zhao, Online botnet detection by continuous similarity monitoring, in: International Symposium on Information Engineering and Electronic Commerce, IEEC'09, 2009, pp. 145–149.
[160] J. Kang, J. Zhang, Q. Li, Z. Li, Detecting new P2P botnet with multi-chart CUSUM, in: International Conference on Networks Security, Wireless Communications and Trusted Computing, NSWCTC'09, vol. 1, 2009, pp. 688–691.
[161] C. Livadas, R. Walsh, D. Lapsley, W. Strayer, Using machine learning techniques to identify botnet traffic, in: Proceedings 2006 31st IEEE Conference on Local Computer Networks, pp. 967–974.
[162] W. Strayer, R. Walsh, C. Livadas, D. Lapsley, Detecting botnets with tight command and control, in: Proceedings 2006 31st IEEE Conference on Local Computer Networks, pp. 195–202.
[163] H. Choi, H. Lee, H. Lee, H. Kim, Botnet detection by monitoring group activities in DNS traffic, in: 7th IEEE International Conference on Computer and Information Technology, CIT 2007, 2007, pp. 715–720.
[164] C.D Cranor, E. Gansner, B. Krishnamurthy, O. Spatscheck, Characterizing large DNS traces using graphs, in: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, IMW 01, ACM, New York, NY, USA, 2001, pp. 55–67.
[165] C.E Wills, M. Mikhailov, H. Shang, Inferring relative popularity of internet applications by actively querying DNS caches, in: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement, IMC 03, ACM, New York, NY, USA, 2003, pp. 78–90.
[166] A. Madhukar, C. Williamson, A longitudinal study of p2p traffic classification, in: 14th IEEE International Symposium on Modeling Analysis and Simulation, 2006, pp. 179-188.
[167] J. Erman, A. Mahanti, M. Arlitt, C. Williamson, Identifying and discriminating between web and peer-to-peer traffic in the network core, in: Proceedings of the 16th International Conference on World Wide Web, WWW'07, ACM, New York, NY, USA, 2007, pp. 883–892.
[168] F. Constantinou, P. Mavrommatis, Identifying known and unknown peer-to-peer traffic, in: Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications, IEEE Computer Society, Washington, DC, USA, 2006, pp. 93–102.
[169] A. J. Aviv, A. Haeberlen, Challenges in experimenting with botnet detection systems, in: Proceedings of the 4th USENIX Workshop on Cyber Security Experimentation, and Test (CSET'11), 2011.
[170] T. Karagiannis, A. Broido, M. Faloutsos, K. claffy, Transport layer identification of p2p traffic, in: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, IMC'04, ACM, New York, NY, USA, 2004, pp. 121–134.
[171] F. Liu, Z. Li, Q. Nie, A new method of p2p traffic identification based on support vector machine at the host level, Proceedings of the 2009 International Conference on Information Technology and Computer Science, vol. 02, IEEE Computer Society, Washington, DC, USA, 2009, pp. 579–582.
[172] A. Ulliac, B.V Ghita, Non-intrusive identification of peer-to-peer traffic, in: Proceedings of the 2010 Third International Conference on Communication Theory, Reliability, and Quality of Service, CTRQ '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 116–121.
[173] W. Chunzhi, J. Wei, C. Hong, W. Luo, H. Fang, Research on a method of p2p traffic identification based on multi-dimension characteristics, in: 5th International Conference on Computer Science and Education (ICCSE), 2010, 2010, pp. 1010–1013.
[174] J.R Douceur, The sybil attack, in: Revised Papers from the First International Workshop on Peer-to-Peer Systems, IPTPS'01, Springer-Verlag, London, UK, 2002, pp. 251–260.
[175] T. Yen, M.K Reiter, Are your hosts trading or plotting? Telling P2P File-Sharing and bots apart, in: Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems, ICDCS'10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 241–252.
[176] A. Jackson, D. Lapsley, C. Jones, M. Zatko, C. Golubitsky, W. Strayer, SLINGbot: a system for live investigation of next generation botnets, in: Conference For Homeland Security, CATCH '09, Cybersecurity Applications Technology, 2009, pp. 313–318.
[177] E. Stinson, J.C Mitchell, Towards systematic evaluation of the evadability of bot/botnet detection methods, in: Proceedings of the 2nd Conference on USENIX Workshop on Offensive Technologies, USENIX Association, Berkeley, CA, USA, 2008, pp. 5:1–5:9.
[178] P. Judge, How bad is the skype botnet threat? 2006 <http://features.techworld.com/security/2199/how-bad-is-the-skypebotnet-threat/>.
[179] CERT, Malware Tunneling in IPv6, Technical Report, US-CERT, 2005.
[180] A. Caglayan, M. Toothaker, D. Drapaeau, D. Burke, G. Eaton, Behavioral analysis of fast flux service networks, in: Proceedings

of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies, CSIIRW'09, ACM, New York, NY, USA, pp. 48:1–48:4.
[181] J. Nazario, T. Holz, As the net churns: fast-flux botnet observations, in: 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008, 2008, pp. 24–31.
[182] T.M.S. Labs, Security Labs Report January – June 2011 Recap, Technical Report, Security Labs, 2011.
[183] D. Moore, C. Shannon, G.M. Voelker, S. Savage, Internet quarantine: requirements for containing self-propagating code, in: IEEE Societies INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications, vol. 3, pp. 1901–1910.
[184] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, G. Vigna, Your botnet is my botnet: analysis of a botnet takeover, in: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09, ACM, New York, NY, USA, 2009, pp. 635–647.
[185] P. Porras, H. Saidi, V. Yegneswaran, An analysis of the ikee.b iphone botnet, in: Security and Privacy in Mobile Information and Communication Systems, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 47, Springer, Berlin Heidelberg, 2009, pp. 141–152.
[186] K. Fogarty, Just what we need: Malware to slave your android to a botnet, 2011.
[187] E. Kartaltepe, J. Morales, S. Xu, R. Sandhu, Social Network-Based botnet Command-and-Control: emerging threats and countermeasures, in: J. Zhou, M. Yung (Eds.), Applied Cryptography and Network Security, Lecture Notes in Computer Science, vol. 6123, Springer, Berlin/Heidelberg, 2010, pp. 511–528, doi:http://dx.doi.org/10.1007/978-3-642-13708-230.
[188] J. Baltazar, J. Costoya, R. Flores, The real face of koobface. The largest web 2.0 botnet explained, in: Trend Micro Research, Trend, Micro, 2009.
[189] Y. Boshmaf, I. Muslukhov, K. Beznosov, M. Ripeanu, The socialbot network: when bots socialize for fame and money, in: Proceedings of the 27th Annual Computer Security Applications Conference, ACSAC'11, ACM, New York, NY, USA, 2011, pp. 93–102.
[190] T. Stein, E. Chen, K. Mangla, Facebook immune system, in: Proceedings of the 4th Workshop on Social Network Systems, SNS'11, ACM, New York, NY, USA, 2011, pp. 8:1–8:8.
[191] D. Misener, Rise of the socialbots: they could be influencing you, in: CBC News Technology & Science, CBC News.
[192] A. Al-Bataineh, G. White, Information loss in enterprise networks: mini-botnes, The ISSA Journal (2011) 36–40.
[193] D. Danchev, Research: small DIY botnets prevalent in enterprise networks, 2005 <http://www.zdnet.com/blog/security/research-small-diy-botnets-prevalent-in-enterprise-networks/4485>.
[194] R. Vogt, J. Aycock, M.J. Jacobson, Army of botnets, in: Proceedings of NDSS'07, pp. 111–123.
[195] N. Falliere, E. Chien, Zeus: King of the Bots, Technical Report, Symantec, 2009.
[196] T. Micro, File-Patching ZBOT Variants: ZeuS 2.0 Levels Up, Technical Report, Trend Micro White Paper, 2010.
[197] M.B. Barcena, Trojan.Spyeye, 2010 <http://www.symantec.com/business/securityresponse/writeup.jsp?docid=2010-0202160135-99>.
[198] J. McDonald, The mariposa butterfly, 2010 <http://www.symantec.com/connect/blogs/mariposa-butterfly>.
[199] D. Sancho, R. Link, Trend micro sinkholes and eliminates a ZeuS botnet C&C, 201 <http://blog.trendmicro.com/trend-microsinkholes-and-eliminates-a-zeus-botnet-cc/>.
[200] D. Sancho, R. Link, Sinkholing Botnets, Technical Report, Trend Micro White Paper, 2010.
[201] P. Coogan, SpyEye bot versus zeus bot, 2010 <http://www.symantec.com/connect/blogs/spyeye-bot-versus-zeus-bot>.
[202] A. Kolesnichenko, A. Remke, P. de Boer, B. Haverkort, Comparison of the mean-field approach and simulation in a peerto-peer botnet case study, in: Proceedings of the 8th European Performance Engineering Workshop, ACM, Borrowdale, UK, 2011, pp. 133–147.
[203] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, P. McDaniel, T. La Porta, On cellular botnets: measuring the impact of malicious devices on a cellular network core, in: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09, ACM, New York, NY, USA, 2009, pp. 223–234.
[204] C. Xiang, F. Binxing, Y. Lihua, L. Xiaoyi, Z. Tianning, Andbot: towards advanced mobile botnets, in: Proceedings of the 4th USENIX Conference on LARGE-SCALE EXPLOITS and Emergent Threats, LEET'11, USENIX Association, Berkeley, CA, USA, 2011, pp. 11–11.
[205] C. Mulliner, J.-P. Seifert, Rise of the ibots: owning a telco network, in: 5th International Conference on Malicious and Unwanted Software (MALWARE), 2010, pp. 71–80.

Sérgio dos Santos Cardoso Silva is a Doctoral Candidate at the Military Institute of Engineering (IME) in the Defense Program and undergraduate lecturer also at IME, Rio de Janeiro, Brazil. Currently serves the Brazilian Army as a major. He received a B.Sc. degree in military sciences from Military Academy of Agulhas Negras (AMAN), Resende, in 1995. Also received a B.Sc. degree in Computer Engineering from the Military Institute of Engineering (IME) in 2001, and received his master degree in Computer Science from also from IME in 2007. His main research areas are network security and cyber-defense.

Rodrigo Mathias Praxedes da Silva is a M.Sc. Student at the Military Institute of Engineering (IME) in the Department of Systems Engineering, Rio de Janeiro, Brazil. Currently serves the Brazilian Army as a Captain. He received a B.Sc. degree in Computer Engineering also from IME in 2005. He worked as network administrator in states of Bahia and Sergipe (2006-2011) and has experience in the management of large network backbones. His main interests are in the areas of network security, network management, P2P networks and cyber-defense.

Raquel Coelho Gomes Pinto received a B.Sc.degree in Computer Science from Universidade Federal Fluminense, Niterói, in 1990. She received M.Sc. and D.Sc. degrees in Systems and Computing Engineering from Federal University of Rio de Janeiro, Rio de Janeiro, in 1994 and 2001 respectively. Currently she holds the position of coordinator of the Computer Engineering course at the Military Institute of Engineering (IME), Rio de Janeiro. Her main research interests include high performance computing, parallel computing, computing systems virtualization and cyber-defense.

Ronaldo Salles is appointed professor at the Military Institute of Engineering (IME), Rio de Janeiro, and currently serves the Brazilian Army as a lieutenant colonel. Since 2007 he holds the position of postgraduate course coordinator in the Department of Computer Engineering. He received his master degree in Computer Science from IME in 1998, and his Ph.D. degree in 2004 from the Imperial College London, UK. His main research interests include performance analysis of computer networks, internet traffic engineering, delay and disruption tolerant networks, network resilience and network security. He received best paper awards at the following conferences: ACM LANC'09, IEEE/IFIP LANOMS'09 and SBRC'08. He has served on the technical program committees of many conferences and has published more than 70 papers in international journals and conference proceedings. He is a member of the IEEE and SBC.