A Survey on Botnet Architectures, Detection and Defences

Muhammad Mahmoud[1], Manjinder Nir[2], and Ashraf Matrawy[2]
(Corresponding author: Muhammad Mahmoud)
Computer Engineering Department, King Fahd University of Petroleum & Minerals[1]
Dhahran 31261, Saudi Arabia
(Email: mimam@kfupm.edu.sa)
Department of Systems and Computer Engineering, Carleton University[2]
Ottawa, ON, K1S 5B6, Canada
(Received Sep. 7, 2012; revised and accepted Nov. 15, 2013)

Abstract

Botnets are known to be one of the most serious Internet security threats. In this survey, we review botnet architectures and their controlling mechanisms. Botnet infection behavior is explained. Then, known botnet models are outlined to study botnet design. Furthermore, Fast-Flux Service Networks (FFSN) are discussed in great details as they play an important role in facilitating botnet traffic. We classify botnets based on their architecture. Our classification criterion relies on the underlying C&C (Command and Control) protocol and thus botnets are classified as IRC (Internet Relay Chat), HTTP (HyperText Transfer Protocol), P2P (Peer-to-Peer), and POP3 (Post Office Protocol 3) botnets. In addition, newly emerging types of botnets are surveyed. This includes SMS & MMS mobile botnet and the botnets that abuse the online social networks. In term of detection methods, we categorize detection methods into three main groups, namely: (1) traffic behavior detection -in which we classify botnet traffic into; C&C traffic, bot generated traffic, and DNS traffic, (2) botmaster traceback detection, and (3) botnet detection using virtual machines. Finally, we summarize botnet defence measures that should be taken after detecting a botnet.

Keywords: Botnet, command and control, distributed denial of service attack (DDoS), fast-flux service networks

1 Introduction

A "botnet" is a term used to describe a network of infected hosts (**Bots**) which are running software robots and are being controlled by a human (**botherder**), via one or more controllers (**botmasters**). The botmaster's communication with its bots is called Command and Control (**C&C**) traffic [38, 46]. Botnets are a serious security threat. They are responsible for most of the email spam, identity theft, online phishing, online fraud, adware, spyware, and DDoS attacks [12]. It is estimated that about 15 percent of the computers connected to the Internet are infected and are used by botnets [3]. It has been documented that one botnet has infected and had more than 400,000 computers under its control [57]. Botnets have a very manipulative behavior, which makes their detection a challenging task. They can stay inactive for a very long time, and may generate a very low traffic volume [43, 57].

According to Bacher et al. [7], the attackers often target class B (/16) or smaller network ranges. Once these attackers compromise a machine, they install a botnet code (called *bot* in short) on it. The bot joins a specific communication server and listen to the C&C channel for further commands from its botmaster. This allows the attacker to remotely control this bot. Grizzard et al. [30] described the primary goals of botnets as follows: (1) **Information dispersion**; sending SPAM, Denial of Service (DoS) attack, providing false information from illegally controlled source. (2) **Information harvesting**; obtaining identity, financial data, password and relationship data. (3) **Information processing**; processing data to crack password for access to additional hosts.

Botnets are difficult to detect for many reasons; botnet's C&C traffic is usually low in volume, hidden in existing application traffic -which makes it look like normal traffic. The number of bots in a given network might be very low, or the botnet may use unusual destination port and/or encrypt its C&C traffic to avoid being detected [33, 45].

To demonstrate the potential of botnets Rajab et al. [56] provided multifaceted observations gathered from real world IRC botnets. To extract IRC specific feature, they used a binary analysis testing tool on gathered botnet traffic by using two independent means namely; IRC tracking and DNS cache probing, across the globe. They observed that to resolve IP addresses of their IRC server, most bots issue DNS queries. By tracking the botnet

traffic, they reported the following findings; (1) **Botnet Traffic Share**: The amount of botnet traffic is greater than 27% of all unwanted Internet traffic. They also observed that out of the 800,000 probed servers, 11% (85,000) were involved in at least one botnet activity. (2) **Botnet Spreading Behavior**: Like Worms, botnets continuously scan certain ports by following a specific target selection algorithm. After receiving a command over a C&C channel, botnets with variable scanning behavior start scanning for variabilities and this spreading behavior is more prevalent. (3) **Botnet Structure**: About 60% of all botnet traffic were IRC bots and only small percentage used HTTP for C&C. (4) **Effective Botnet Sizes**: Maximum size of online population is significantly smaller than the botnet footprint (number of hosts infected with the botnet). Moreover, they found that the botnet population depends on different time zones.

Botnets have been surveyed from different perspectives in the literature. Short overviews have been presented in workshops and conferences' surveys [8, 26, 40, 69]. Bailey et al. [8] focused on botnet detection and data sources. Their survey did not focus on botnet architecture or classification and no defence methods were surveyed. However, they surveyed botnet detection by cooperative behavior, attack behavior and signature based detection. Feily et al. [26] focused on botnet detection methods, but neither on botnets' architecture nor their classification, and no defence methods were surveyed. They [26] explained four botnet detection methods, namely; signature-based, anomaly-based, DNS-based and mining-based. Zhu et al. [69] put most of the focus on understanding botnet architecture anatomy where botnets were divided into IRC, HTTP, P2P botnets and fast-flux networks. Detection of botnets by honeynets and by traffic monitoring were mentioned. They [69] referenced a defence against spam and suggested a commercial security service for enterprises. In [40], Chao Li et al. presented a short survey on botnets and their evolution. They outlined botnets' infection mechanism, malicious behavior, C&C methods, communication protocols and they suggested some directions for botnet defence. In [63], Thing et al.'s survey focused on botnets that are used for DDoS attacks, and analyzed four DDoS attack botnets, and the way they launch their attacks. The article by J. Liu et al. [42] surveyed specific known botnets with their malicious activities and some detection methods. They highlighted IRC-based and P2P-based botnets. The majority of their survey focuses on some of the most popular botnet and their malicious activities. Their survey outlined four detection methods based on honeynets, IRC traffic analysis, IRC anomaly activities and DNS. They also referenced countermeasures for the public, home users and system administrators. Shin and Im [59] surveyed the threats and challenges of botnets. They classified botnet detection into C&C-based and P2P-based detection. They also outlined some defences against DDoS attacks. In a short survey [41], Lin and Peng focused on the detection methodologies and detection techniques of botnets. Among the various kinds of botnet attacks, they discussed three types of botnet attacks. Further, they described detection methodologies of botnet and surveyed two detection techniques. In [51], IBM published a report on botnets' risks and prevention. This report lists the security risks of botnets (such as DDoS, privacy, SPAM, phishing, etc.). Then, it focuses on IRC botnets and how they work. Finally, the report suggests some prevention measures against botnets and their risks. In 2011, the European Network and Information Security Agency (ENISA) published a report on botnet threats from industrial perspectives with focus on practical issues [53].

Our Contribution. Besides presenting an extensive survey on botnets and their detection mechanisms, we classified botnets based on underlying (C&C) infrastructure as well. Our survey has elaborated description of Fast-Flux service network and C&C rallying mechanism as both play very important roles in botnets' activities. To the best of our knowledge, this is the first survey that includes new emerging types of botnets like mobile and online social networks botnets. It also gives a comparison between current detection techniques as shown in Table 1 and explains the different attempts to create botnet behavior models.

Objective. The objective of this survey is to shed light on botnets threat by providing a clear background and classification on botnets architectures and their behavior, and to describe some security measures that are used to detect and mitigate botnets threats.

Survey Outline. This survey is outlined as follows: Section 2 lays some background on botnets architectures and mechanisms that botnets use to control other hosts. It analyzes botnets behavior and summarizes botnets models. Furthermore, it outlines some botnets facilitation features. Section 3 classifies botnets based on their underlying communication infrastructure, and surveys mobile botnets and online social networks botnets. In Section 4 we provide detail on botnets detection algorithms and classify them into; behavior-based, botmaster traceback-based, and virtual machine-based detections. We briefly illustrate some post-botnet detection measures. Finally, Section 5 concludes the survey.

2 Botnet Architectural Elements

In this section, we start by explaining what C&C is. Then, botnets infection behaviors and known botnets models are described. After that, we explained how fast-flux service networks work and surveyed botnets' C&C rallying mechanisms.

As Barford et al. [11] described, botnets usually have some of the following architecture features; They use existing protocols for their C&C communication (i.e. IRC, P2P, etc.). They may have the ability to exploit large

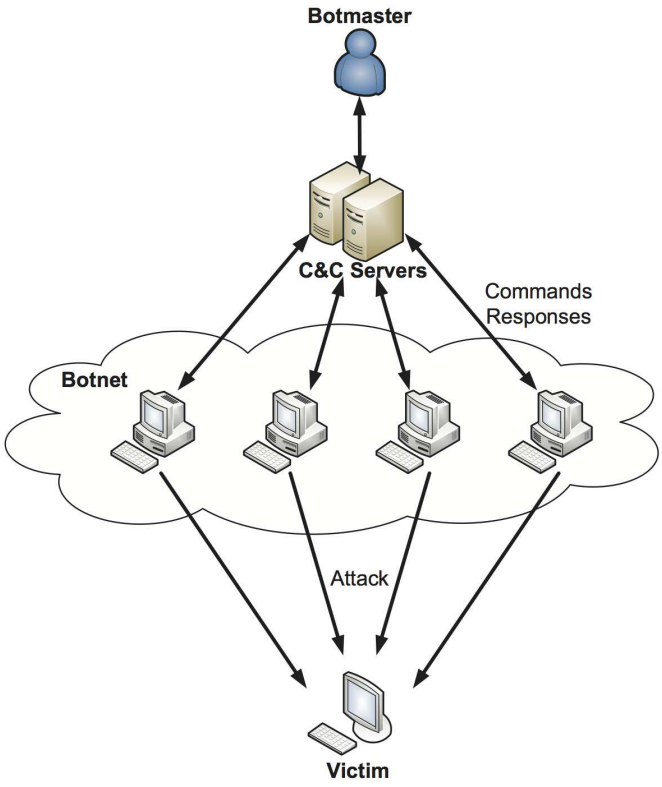

Figure 1: (IRC, HTTP, POP3) C&C architectures are usually centralized

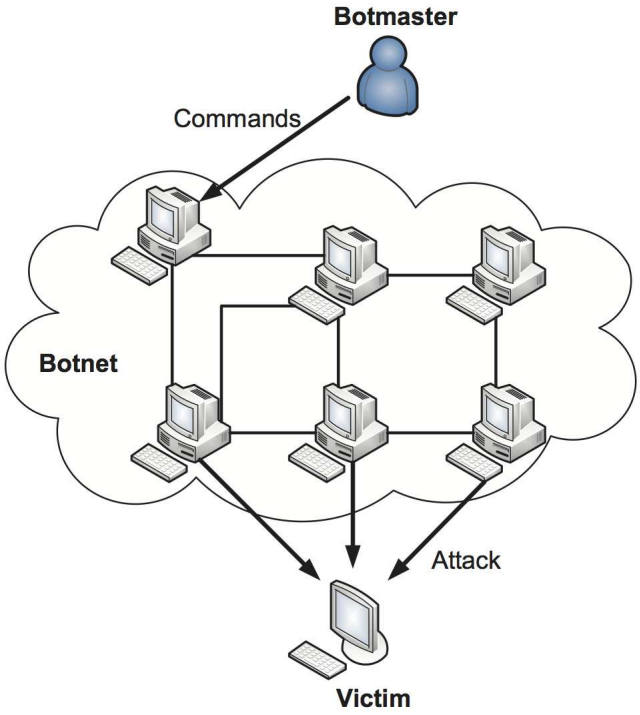

Figure 2: P2P C&C architectures are usually decentralized

number of targets, launch different types of DoS attacks, spy for passwords, fool the defence mechanisms, frustrate disassembly software, hide themselves from the local system, scan ports for vulnerabilities. Furthermore, botnets may encrypt their C&C traffic and/or may come with the C&C only and download other functionalities later on, as needed.

A victim host could be infected by targeting known vulnerability or by infected programs (like viruses). Once the host is infected, the bot can use any of the following mechanisms to control the infected host [11, 32, 52]; (1) secure the system (e.g. close NetBIOS shares, RPCDCOM), (2) spy or steal identity, (3) send SPAM emails, (4) host illegal sites, (5) redirect traffic for the botnet (e.g. fast-flux), (6) kill unwanted process running on the system (e.g. anti-virus, taskmanager, etc.), (7) test for virtual machines and/or debugger software, (8) add or delete autostart applications, (9) run or terminate programs, (10) download and execute files, (11) perform address and port scan, (12) rename files, (13) simulate key presses, (14) run DDoS attacks.

Furthermore, botnet propagation could be through horizontal or vertical scans. The horizontal propagation scan is done on a single port access for some address range. On the other hand, vertical scan is done on a single IP address across a range of ports [11, 32].

2.1 Command and Control

A botmaster's communication with the botnet is carried out via C&C. The C&C is the main feature that distinguishes botnets from other malwares [29]. It allows the botmaster to communicate with the botnet and give commands. Theoretically, the botmaster can command the botnet to do any task including; performing DDoS attacks, spamming, spying, identity theft, etc. [31, 36, 62].

To avoid detection, botnet designers tend to use widely used protocols for their C&C. Most botnets use IRC commands for their C&C communication [23, 51]. However, some botnets use the HTTP, POP3 or P2P protocols for their C&C communication. Newly emerging types of botnets use SMS, MMS, or online social networks for C&C communication.

The IRC, HTTP and POP3 botnets are usually centralized in the sense that their C&C channels depend on specific servers and if they are disabled, botnet will cease to exist. Figure 1 shows an architecture of botnets with centralized C&C server(s). On the other hand, P2P botnets are usually decentralized, as shown in Figure 2. In Section 3, we discuss botnets C&C in more detail as it is the main criterion of our botnets architecture classification.

2.2 Botnet Infection Behavior

Most botnets run in four phases [69]. A node's transition from a clean host to a zombie host, and reacting to its botmaster's commands, goes through four steps.

First, the initial infection starts when botnet nodes scan the network looking for vulnerabilities. They scan for back doors [69], known buffer overflows, known vulnerable network administrator tools. They may run brute force password scanning for some services (e.g. SQL servers, NetBIOS shares, etc.) [11]. Second, the secondary injection starts when a vulnerability is exploited and the victim host downloads and runs the bot's binary code [32]. Then, the bot establishes a connection to the botnet's C&C server, and starts to control the host (e.g. disable anti-virus, change NetBIOS shares, etc.). Finally, the malicious activities begin when the bot starts to act on botmaster's commands (e.g. run DoS attack, send SPAM, etc.) and then the botnet maintains and upgrades itself periodically [11].

In the case of P2P botnet, the first two steps are similar to other botnets. After the initial infection and injection, the P2P botnet uses an initial peer list to contact the initial peers. Once it finds a live peer, phase one starts where botnet updates its peer list and download any available updates. After that the node goes to phase two when it starts its malicious activities [27]. The aforementioned P2P botnet behavior is based on the STORM malware behavior. Other P2P malwares should -to some extend- have similar behavior [61].

2.3 Modeling Botnet Architectures

There has been different attempts at creating models for botnet behavior either to help understand botnets or to give the researchers a head start of possible future botnet designs. The following are examples of these models.

- **Diurnal Propagation Model**: A model by D. Dagon et al. [25] shows propagation dynamics in botnets and describes that time zones and geographical locations play a critical role in malware propagation. All the botnets studies use DNS to locate their C&C servers. However, through binary analysis, this model has confirmed that most botnets do not use hard coded IP addresses. In this model, an approach is used to predict botnet dynamics prior to an attack, and focuses on capturing any network cloud of coordinated attackers rather than tracking individual bots.
- **Super-Botnet Model**: Vogt et al. [67] stated that traditional botnets are easily detectable through their C&C. Therefore, they proposed a possibility of super-botnet, which is a network of independent botnets that can be co-ordinated for large scale attacks. To establish a super-botnet a two phase process is explained. The authors suggest that a super-botnet protects itself from defenders by not allowing individual botnets to have complete information about the super-botnet rather each botnet can have partial routing information to contact a small finite set of its neighbours.
- **Stochastic Peer-to-Peer Model**: Van et al. [58] presented a botnet stochastic model for the creation of a P2P botnet. The model was constructed in the Möbius [4] software tool. In this model, authors examine the growth of botnet size based on different parameters and suggest the development of future anti-malware systems against P2P botnets.
- **Advance P2P hybrid Model**: Keeping in view weaknesses of P2P botnet architecture, Wang et al. [68] proposed a design of an advanced hybrid P2P botnet architecture. The architecture is harder to be shut down or monitored by defenders. In this model, each bot has its individual encryption design and robust connectivity to other bots. The botnet can disperse communication traffic to different service ports in a way that the botnet will not be exposed if one of its bots is captured. Furthermore, the authors alarm us of advanced botnet attack techniques that could be developed by botmasters in the near future and propose honeypot to defend against such advanced botnets.

2.4 Botnet Facilitators

Botnets usually use some techniques to alleviate their activities. In this section, some of these techniques are surveyed.

2.4.1 Fast-Flux Service Networks

DNS is an Internet service which translates names of sites into their numeric IP addresses. Usually DNS do not respond to DNS requests with unique *'A'*[1] record. For every host, DNS has a list of *A* records each with a given Time-to-Live (TTL) value (normally from 1 to 5 days). DNS returns these *A* records in round-robin way [35]. This implementation of DNS is called Round-Robin DNS (RRDNS). Furthermore, in Content Distribution Networks (CDN)[2], the DNS is implemented in a sophisticated way that it finds out the nearest edge router and returns it to the client. CDN uses a much lower TTL value than RRDNS to enable them to react quickly to link changes. On the other hand, a Fast-Flux Service Network (FFSN) is a distributed proxy network -built on compromised machines (flux-agents)- that direct incoming DNS requests to the botnet's desired address on the fly [35].

Nazario et al. [49] and the Honeynet Project [7] discussed fast flux networks that are used as botnet C&C networks. Botnets use fast-flux DNS techniques to host unauthorized or illegal content within a botnet. This is done to allow the botnet's domain name to have multiple IP addresses. In the meantime, involved DNS records are constantly changing every few minutes using a combina-

[1] *'A'* record is a mapping between host name and IP address

[2] Also called content delivery network. It is a system in which many copies of data is placed in different location in the network. The purpose is to maximize the bandwidth, so when a user node tries to access some data, it will be directed to the server closest to it, rather than allowing all users access data on centralized server.

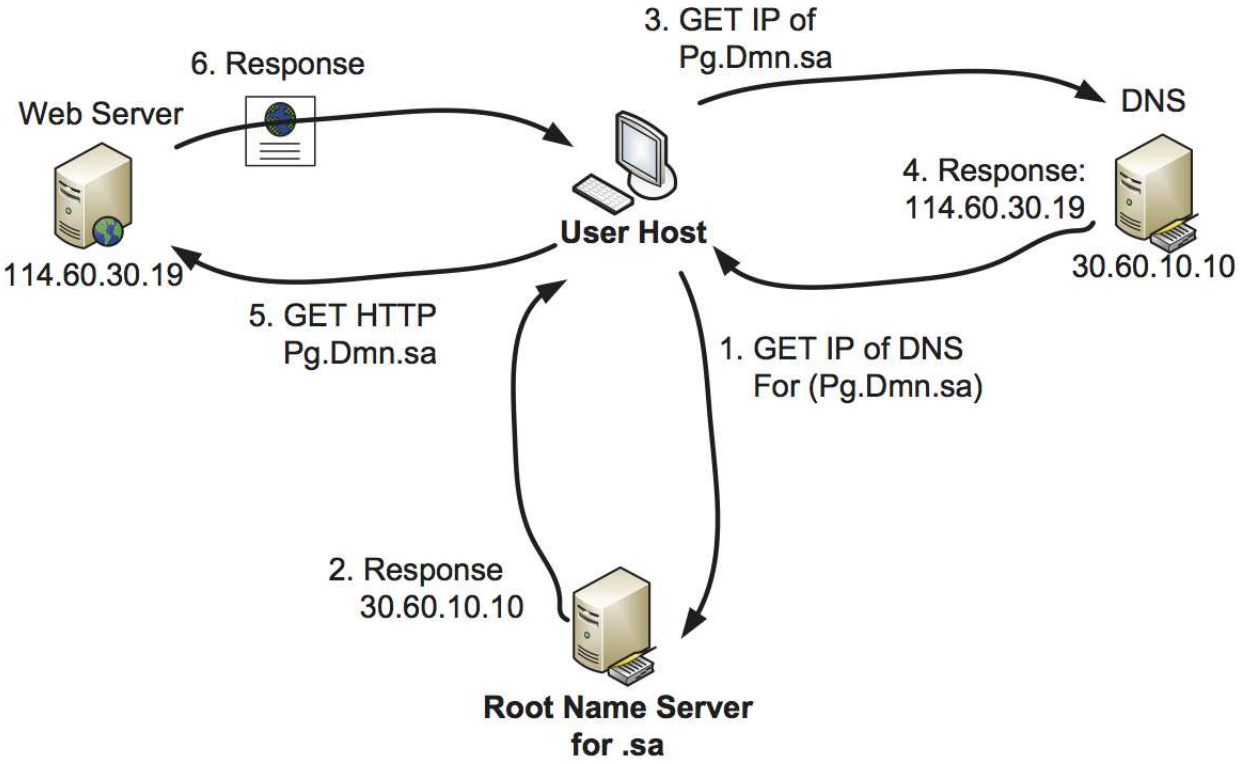

Figure 3: Normal content retrieval process

tion of round robin IP addresses and a very short TTL from any given particular DNS resource record (RR).

In FFSN, the victim client first sends an address query to DNS. Then, the DNS returns the IPs of a subset of active flux-agents. After that, the flux-agent relays the client's request to the mothership[3]. The key factor in FFSN is the combination of a very short TTL and the round-robin answer from a large pool of active agents. Because the TTL is short, the following DNS request will result in a totally different flux-agent. FFSN have high availability because the mothership continuously updates the pool of active agents. To have good understanding of fast-flux we need to learn all the steps for normal DNS query and ignoring steps that are unrelated to the fast-flux concept. The following steps describe the process of content retrieval -for web address "Pg.Dmn.sa"- in normal DNS queries. As Figure 3 shows, the web address is traversed from right to left. (1) The user host asks the ".sa" root name server for the IP address of the DNS responsible of the domain "Dmn.sa". (2) The ".sa" root name server replies with an IP address (30.60.10.10 in this case). (3) The user host then uses this IP address to contact to the DNS and ask it for the IP address of "Pg.Dmn.sa". (4) The DNS replies with an IP address (114.60.30.19 in this case). (5) The user host then uses this IP address to contact to the web server for the HTTP content of "Pg.Dmn.sa". (6) The web server responses with the requested contents [7, 35].

[3]a secret controlling element of the botnet.

As the botmaster tries to hide the IP address of unauthorized or illegal website(s), it tends to fast-flux their IP address(es). Figure 4 illustrates the steps of retrieving the content of fluxed web address "FlxPg.Dmn.sa". Steps (1) to (5) are identical to normal content retrieval steps, except that when the IP address of "FlxPg.Dmn.sa" is requested, the DNS response comes with a short TTL. Therefore, any subsequent DNS query would probably get a different IP address response. After the user host uses the IP address (114.60.30.19 in this case) to contact the "alleged webserver" requesting the contents of "FlxPg.Dmn.sa", this "alleged webserver" will carry out two more hidden steps. (5a) The "alleged webserver" will request the content of "FlxPg.Dmn.sa" from the mothership. (5b) The mothership responses with the requested contents. (6) The "alleged webserver' redirects the response from the mothership to the user host.

Sometimes, botmasters take one more step to make it more difficult to locate them by fluxing the IP address of the DNS too. Figure 5 illustrates the steps of retrieving the content of fluxed web address with fluxed DNS "FlxPg.FlxDmn.sa". (1) The user host asks the ".sa" root name server for the IP address of the DNS responsible of the domain "FlxDmn.sa". (2) The ".sa" root name server replies with an IP address (30.60.10.10 in this case) with short TTL. (3) The user host then uses this IP address to contact to the "alleged DNS" and ask it for the IP address of "FlxPg.FlxDmn.sa". (3a) The "alleged DNS" passes this request to the mothership. (3b) The mothership responses with an IP address (114.60.30.19 in this case) (4)

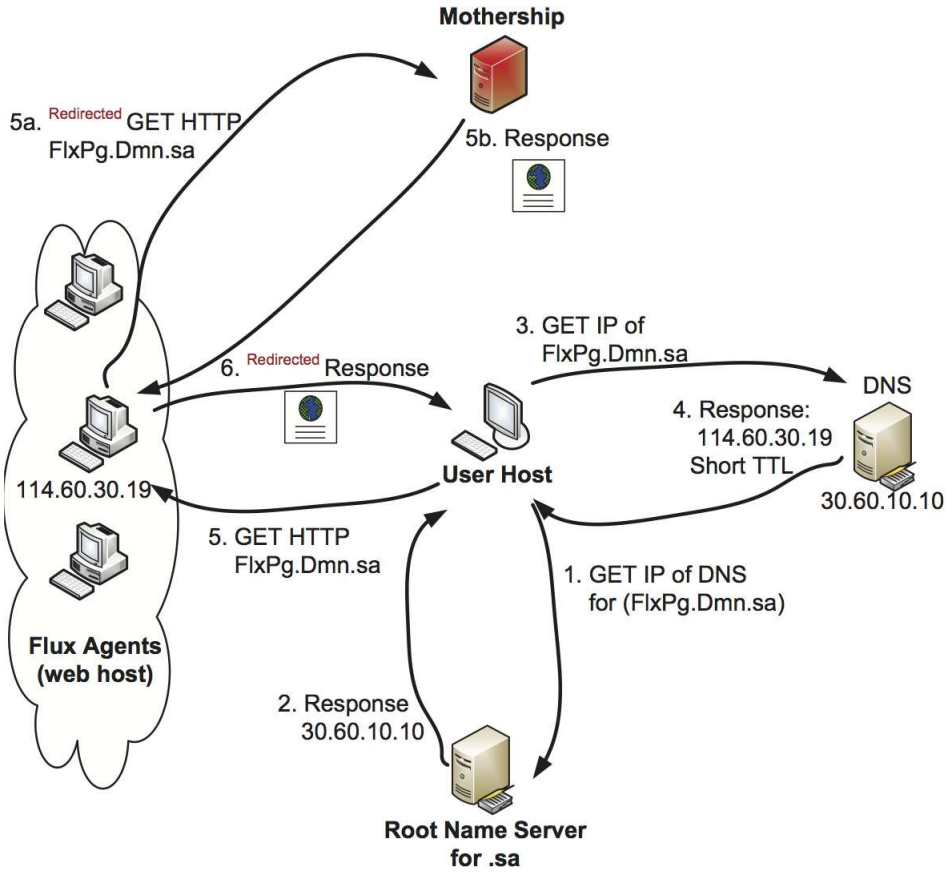

Figure 4: Single-Flux content retrieval process

The "alleged DNS" redirects the mothership's response to the user host. (5) The user host then uses this IP address to contact to an "alleged webserver" for the HTTP content of "FlxPg.FlxDmn.sa". (5a) The "alleged webserver" will request the content of "FlxPg.FlxDmn.sa" from the mothership. (5b) The mothership responses with the requested contents. (6) The "alleged webserver' redirects the response from the mothership to the user host.

To summarize, there are three types of fast-flux. (1) Single-Flux: when IP address of an unauthorized or illegal webpage is fluxed. (2) Name Server (NS)-Flux: when IP address of DNS is fluxed. (3) Double-Flux: when both IP addresses of the webpage and the DNS are fluxed [16]. Figures 3, 4, 5 illustrate the difference in content retrieval process between normal, Single-Flux and Double-Flux Service Networks.

Holz et al. [35] lists some features of FFSN that might help in detecting them. First, legitimate domains return 1 to 3 *A* records, but FFSN return 5 or more *A* records. Second, legitimate domains return a small number of nameserver (NS), but FFSN returns several NS records and several *A* records for the NS records. Third, legitimate domains return a small *A* records only from one autonomous system (AS), but FFSN tends to be located in more ASs. Furthermore, FFSN does not have the freedom to choose hardware and IP address. Therefore, the range of their IP addresses is diverse. Finally, since there is no physical agent control, therefore, there is no guaranteed up time [35, 52].

2.4.2 Command and Control Rallying Mechanisms

According to Choi et al. [20], botmasters want their bots to be invisible but portable, therefore they use different methods for bots rallying. They stated that not all bots can have mobility and invisibility at the same time. They described three rallying methods, namely; hard-coded IP address, dynamic DNS, and distributed DNS.

In hard-coded IP address method; the bot binary has a hard-coded IP address of its C&C server, the server can be detected through reverse engineering, and the botmaster can be quarantined or the botnet can be suspended. As hard-coded IP address cannot be changed, this method cannot provide mobility and does not make the botnet invisible as well. On the other hand, in dynamic DNS botnets migrate their C&C server frequently, upon the instruction of botmaster. Using a list of servers provided in the bot binary, a botmaster uses several C&C servers. It uses dynamic DNS in order not to be detected or suspended, and to keep the botnet portable. When connection to the C&C server fails or shutdown, the bots will perform DNS queries and will be redirected to a new C&C server [2]. This redirection behavior of botnets is known

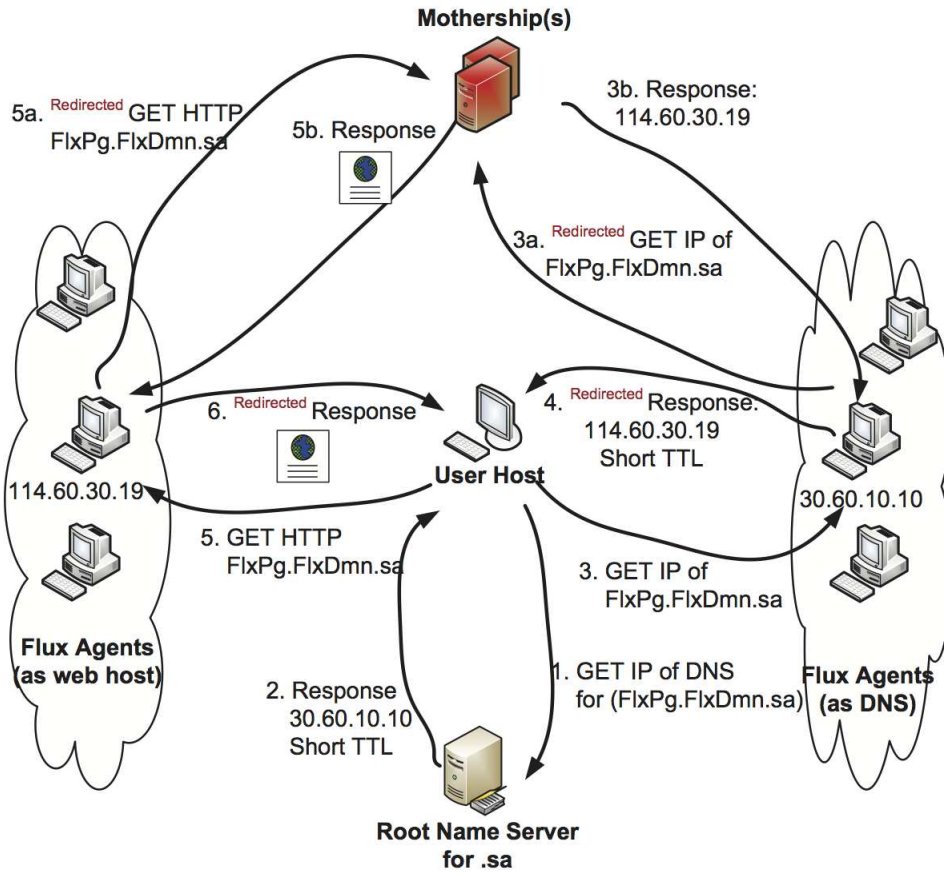

Figure 5: Double-Flux content retrieval process

as "herding". This method provides mobility and some invisibility to the botnets. Finally, with distributed DNS, botnets run their own distributed DNS service at locations that are out of the reach of law enforcement. Bots include the addresses of these DNS servers and contact these servers to resolve the IP address of C&C servers [2]. This method provides both mobility and invisibility to their botnets.

In summary, while the hard-coded IP botnet makes very easy for the newly infected nodes to join the botnet, it also makes easy for law enforcement to track and shutdown the botnet. On the other hand, using DNS to migrate C&C servers make it harder for the newly infected nodes to join the botnet. Some infected nodes might never be able to join the botnet -in case they stay offline long enough for all the addresses in the initial communication list to be obsolete, however, it gives the botnet the flexibility to hide its C&C severs.

3 Botnet Architecture Classification Based on C&C

In this section, we give a classification of botnets. Botnets can be classified based on their C&C traffic protocol.

3.1 IRC Botnets

The IRC [50] protocol was designed to facilitate a chatting environment. Its simplicity and distributed structure enables the earliest and most common botnets to use it for their C&C communication [23, 45]. IRC has many properties, which make it attractive for an attacker, such as, its redundancy, scalability and versatility. Furthermore, due to its long term and wide spread use, there is a large base of knowledge and source code to develop IRC-based bots [30].

As described by Cooke et al. [23], IRC is a well-known public exchange point and enables virtually instant communication, which provides a common, simple, low latency, wide availability and anonymity command and control protocol for bot communication. An IRC network is composed of one or more IRC servers. According to the botnet design, each bot connects to a public IRC network or a hidden IRC server. The bot receives commands from the controller (botmaster) and can be instructed to attack. The simplicity and multicast delivery mechanism of the IRC protocol fascinate attackers to use this protocol to send instructions and commands to bots.

Wang et al. [68] and Gizzard et al. [30] observed that most easily detected botnets use IRC for their C&C communication. They pointed out the weaknesses of IRC botnet because of its centralized server architecture. More-

over, IRC traffic is usually un-encrypted and once the centralized IRC C&C channel is detected by the defender the whole botnet could be disabled by shutting down the centralized server.

3.2 HTTP Botnets

In this botnet architecture, HTTP servers are used to distribute bot commands. Botnet members poll HTTP server(s) from time to time to get new commands [36].

Chiang et al. [19] have described botnets using HTTP as C&C mechanism. According to them, an HTTP bot is setup to communicate with certain webserver(s) using an HTTP post, which contains unique identifiers for the botnet, and in return the webserver will send the HTTP commands that it has been setup with. Afterwards, the bot could download malware files, spam information, or even DDoS instructions. The connection in HTTP botnet cannot be initiated from botmaster(s) as it does with the IRC botnets because botmasters and the bots are not constantly present on the HTTP channel. Compared to IRC, Nazario in [47] mentions that botnet designer could have two benefits of using HTTP for C&C communication. First, HTTP C&C is harder to detect as it blends into majority of traffic. Second, existing firewall policies block IRC C&C botnets, but HTTP based botnets can pass firewall policies. On the other hand, once the bot's HTTP server is identified, it can be isolated and shutdown [36].

3.3 POP3 Botnets

Singh et al. [60] have developed a prototype bot that demonstrates the feasibility of email-based botnet C&C. Jennifer Chandler [17] studied a bot that uses POP3 protocol for C&C communications. In a POP3 C&C architecture, the bot connects to a predefined mail server to retrieve an email message, which contains commands as email attachments and can respond to commands through the same channel. Chandler also mentioned that this traffic will be less detectable than a connection to IRC server. Similarly, Singh et al. [60], demonstrated that botnet commands can remain hidden in spam due to its enormous volume. If email service providers deploy specialized detection of spam-based botnets, botmasters can alternatively communicate with bots via non-spam email that cannot be safely discarded.

3.4 Peer-to-Peer Botnets

Holz et al. [36] explains a new decentralized architecture of botnets that is based on P2P protocols. P2P botnets are relatively new generation of botnets that do not use a central server to send C&C commands to botnet members. P2P botnets usually use publish/subscribe systems to communicate [36]. Unlike IRC botnets, the attacker in P2P botnets cannot send commands directly to bots, instead, a set of commands (C) is defined in the P2P system, and all bots subscribe to this set. When an attacker (any bot) wishes to launch an attack, it publishes a command (c_i) on the P2P system. All bots subscribed to the set will be able to see the command.

Barford et al. [11] have anticipated that future botnet development will include the use of encrypted C&C communication. As stated by Grizzard et al. [30], in a P2P architecture, there is no centralized point for C&C and bots communicate with other peer bots instead of a central server. Nodes in a P2P network act as both servers and clients. Therefore, there is no centralized coordination point that can be incapacitated. The authors analyzed the case study of the "Trojan.Peacomm" bot and observed that the P2P protocol is essentially being used as a name resolution server to upgrade the bot.

Wang et al. [68] pointed out some weaknesses of known P2P bots and proposed a new P2P bot architecture. According to them, botnets such as *Sinit*, *Phatbot*, *Nugache* and *Slapper* have implemented different kinds of P2P control architectures. A *Sinit* bot host finds other *Sinit* bot hosts by using random probing. The extensive probing traffic will make it easy to detect the botnet. *Phatbot* uses Gnutella cache servers for its bootstrap process. This also makes the botnet easy to be shut down. *Nugache* relies on a seed list of C&C IP addresses during its bootstrap process. This makes it weak. *Slapper* does not have encryption and its command authentication enables others to easily hijack it. Keeping in view these weaknesses of P2P botnet architecture Wang et al. propose the design of advanced hybrid P2P botnet architecture, which is much harder to be shut down or monitored. Their hybrid P2P botnet architecture provides robust network connectivity, individualized encryption, controlled traffic dispersion and easy monitoring and recovery by its botmaster. Furthermore, if a bot is captured, the botnet exposure will be limited.

Figure 2 shows an architecture diagram for P2P botnets. Unlike IRC or HTTP botnets, any bot in a P2P botnet can publish a command. Therefore, if one is able to identify a botmaster and bring it down, the P2P botnet will still be functional because any bot can issue botnet commands (i.e. be a botmaster).

3.5 Other Botnet Emerging Types

3.5.1 SMS & MMS Mobile Botnets

In this section, we are going to discuss botnets that spread on mobile devices and could use SMS or MMS for C&C communication.

Challenges. Mobile devices/smartphones pose some challenges which make them unattractive to botnet developers [24, 28]. They usually use the cellular network for communication. Thus they are not continuously available on the Internet. Even when smartphones access the Internet, they are usually either behind firewalls or using

dynamic private IP addresses that are not available on the Internet. This makes difficult to mobile bots to be visible to the botmaster. Therefore, botmasters will not be able to send direct commands to their bots. Furthermore, cellular networks do not use DNS to find mobile nodes. This will make botmaster invisible to botnets to make use of DNS services like Fast-Flux (see Section 2.4.1).

Compared to PCs, smartphones have limited power. Therefore, any abnormal consumption could lead to investigation that might detect the bot. Furthermore, data traffic or messaging cost is very noticeable in cellular networks. Therefore, if abnormal network traffic is generated because of the botnet, it can be easily noticed. Frequent IP change and not having long lived public IPs makes it impractical to deploy P2P-based C&C infrastructure [24]. In mobile botnets, botmaster could use traditional C&C methods (like IRC or HTTP) to communicates with its bots. However, cellular network architecture may limit the bots connectivity [65].

Effect. To overcome the challenge that mobile devices couldn't be reached directly by their botmasters, botmasters could use web servers to post their commands and updates [28]. Then, bots could access these web servers -when the mobile device is available on the Internet- to pull these commands [28]. Some botmaster could rely on the fact that most mobile devices have access to wireless LANs. This enables the botmaster to launch attacks on both, the Internet and the cellular network [64]. Mobile devices could send SMS and MMS to connect to their C&C proxy servers. In such case, these proxy servers would have access to the Internet and would be able to understand and read these SMS and MMS [28].

On the other hand, once a bot has control of a mobile device, it will be able to exploit services in the mobile network. It could monitor incoming messages and delete them before they appear in the inbox. Furthermore, as business users are using their mobile devices to access their banking information, store their sensitive information, order credit reports, and much more, which makes bot spying more threatening.

History. Botnets on mobile phones were almost unheard of until 2009 [24, 28]. In July 2009, Symantec [6] reported a mobile phone botnet that uses a "good old-fashioned social engineering mixed with SMS spam" to propagate. A phone will be infected when the user downloads from the external URL provided in an SMS. The bot hides itself in a process that has a name similar to a legitimate application. To defend itself, the bot is capable of ending applications that could allow the user to manually terminate the bot process. It connects to its server, using HTTP, to update itself and send spy information to the botmaster.

SRI international published a technical report [54] on "iKee.B", an iPhone bot that was captured on 25 November 2009. This bot is capable of checking its C&C server every five minutes to pull additional instructions and run their scripts on the hacked iPhone. It could also run scans on WiFi or some IP address range to infect other vulnerable iPhones. This is a very simple botnet that took very little memory of the iPhone. It has all the functionality that is expected of PC botnets, yet it has a flexible code base, which makes it very dangerous.

During RSA Conference 2010 in San Francisco, Derek Brown and Daniel Tijerina [34] demonstrated how they were able to develop a weather application that provides mobile users with weather forecast. However, to download this application, user had to approve some permission. According to [34], within an hour, there was 126 downloads of this application, then 702 downloads after eight hours. In few days, the number of smartphones running this application was about 8000. To show the danger of botnets on mobile phones, they wrote a malicious version of this application that can send spam, get user's physical addresses and contact information. It could also steal user's files, email, passwords, and access Facebook and Twitter accounts.

Xiang et al. [24] proposed a stealthy, resilient, low cost, Android-based botnet. They called it Andbot. They suggested a new centralized C&C topology called URL Flux. In URL Flux, there are a fixed number of C&C servers that can be accessed by the bot. This list of servers is built using Username Generation Algorithm (UGA). Therefore, the bot connects to a hardcoded Web 2.0 server (one of many) then traverse through a list of users generated by the UGA. Once a life user is found, Andbot commands can be deployed. This provides resilience to the mobile botnet since there are more than one centralized C&C server. To reduce the cost, Andbot avoids using SMS or Bluetooth for C&C. Instead it relies only on IP communication. Andbot uses RSS and GZIP to reduce its traffic.

In [64], Traynor et al. demonstrated how -by attacking the Home Locator Register (HLR) - a relatively small cellular botnet could cause a nation-wide outage in cellular network services. By launching network services requests (like $insert\_call\_forwarding$), the mobile botnet could cause serious degradation in the service without raising the attention of the bot-infected device owner [64].

3.5.2 Social Networks Botnets

In this section, we describe two types of online social networks (OSNs) botnets. The first type is the botnets that use existing (OSNs) for C&C communication (OSNs C&C Botnets). The other type is the botnets that are actually comprised of OSNs accounts (Socialbots)

OSNs C&C Botnets. In 2009, the first actual botnet that used social networks for C&C was reported by Jose Nazario [48]. The botmaster used accounts on Twitter.com and Jaiku.com to command bots to download and run malicious activities. In [39], Kartaltepe et al. studied this bot and found that it works as follows: The bot sends HTTP GET request to the botmaster's RSS. The return RSS feed has encoded message, which is decoded

by the bot to get one or more URLs. These URLs direct to a site with longer URLs each pointing to a malicious zip file. After downloading the zip file, the bot unzips and runs the malicious program. The malicious program collects user information and send it to the botmaster.

In 2009, Trend Micro published a technical report explaining the largest Web 2.0 botnet (KOOBFACE) [10]. KOOBFACE bot is comprised of many components. These components could be as follows: (1) **KOOBFACE downloader**: finds out the victims' social network, connects to C&C channel, and download other components. (2) **Social network propagation components**: this is the KOOBFACE worm that sends out the infection SPAM. (3) **Web server component**: makes the victim a web server for the KOOBFACE botnet. (4) **Ads pusher and rogue antivirus (AV) installer**: installs fake antivirus on the victim's machine and opens an ads window. (5) **CAPTCHA**[4] **breaker**: gets the victim to challenge-response tests. (6) **Data stealer**: steals product IDs, profiles, credentials, etc., and sends them in encrypted form to the C&C server. (7) **Web search hijackers**: intercepts the victim's search queries and redirects them to suspicious sites that returns result with some agenda. (8) **Rogue DNS changer**: modifies the victim's DNS to a fake one. This fake DNS intercepts the victim's web requests, delivers malware, and/or prevents the victim from accessing antivirus websites. KOOBFACE infection starts with a SPAM asking the user to watch a video. By clicking on the video's URL, the user will be directed to YuoTube (not YouTube) and will be asked to download an executable to watch the video. This executable is the aforemention KOOBFACE components downloader. KOOBFACE has an update capability that make it difficult to shutdown [10].

In [46], Nagaraja et al. designed a botnet that uses images shared by OSNs users for C&C communication. Bots in this botnets can communicate if they are hosted on computers for people using an OSN. When the botmaster issues a command, it uses images and post them on Facebook. When users of infected computers log into Facebook and view these images, the bot code on their computers intercepts these images and extracts the required information from them. On the other hand, to send stolen information to the botmaster, the bot waits for the user to post an image. It intercepts the the image and injects the data into it [46].

Socialbots. Socialbot is defined as "*an automation software that controls an adversary-owned or hijacked account on a particular OSN, and has the ability to perform basic activities such as posting a message and sending a connection request*" [13]. It is predicted that about 10% of social network individuals will be robots by 2015 [37]. The main character of the socialbot networks (SbN) is that instead of financial means, they focus on having substantial relationships with human users. They can cause peer effect to promote for a product or social engagements. Therefore, they have the potential for a great future opportunity [37]. For instance, to determine influences, some services score user's activity and their effect in the network. SbN can hijack and change these scores. SbN can learn the social graph, analyze people posts, decide what to say and to whom. They do that by posting and following. The digital space gives the SbN a "near-perfect" world to apply artificial intelligence theories. In Twitter, rate of friendship between user can be greatly changed using SbN [37].

For example, the Realboy Project compilation [22] is about designing a Twitter botnet that imitates human users with three main goals; (1) to repost external users tweets, (2) to follow other users, (3) to get 25% follow-back rate. In addition, Boshmaf et al. [13] designed and analyzed a socialbot. Their Facebook botnet had one botmaster, 102 bots, and they ran it for eight weeks. During this period, the socialbot sent 8570 friendship requests. 3055 out of these requests were accepted. They recorded related data and all accessible profile information. Boshmaf et al. [14] concluded that online social networks (OSN) are vulnerable to large scale infiltration and that SN defence systems do not try to prevent against infiltration campaigns. Furthermore, they [14] concluded that socialbots could be profitable and could cause serious privacy breaches. Therefore, socially-aware software security could be at risk.

Threats. As socialbots infiltrate social campaigns, they could pose some security threats [15]. By polluting the social relationship in OSN, the polluted OSN can no longer be trusted. Socialbots could be used to spread rumors, spread malware, influence trading by giving fake high rates to online products. Furthermore, they could perform online surveillance and harvest users private data to use them for targeted SPAM or phishing campaigns.

Detection. As botnets on OSNs are relatively new, their detection mechanisms are not mature enough. Therefore, some OSN mechanisms are mentioned here instead of Section 4.

Kartaltepe et al. [39] proposed a mechanism to detect botnets that use OSNs for C&C. Their proposal has server-side and client-side countermeasure. The server-side detections is based on the fact that posts are expected to be in plain test. It looks for text attributes and uses a light-weight machine learning algorithm for real-time detection. It also follows any URL in the post to make sure it is from trusted sources. Furthermore, to determine if a process is a bot, the client-side looks for three attributes, namely; self-concealing, dubious network traffic, and unreliable provenance. To determine if a process is self-concealing, they check if it was started without human interaction and it does not have a graphical user interface. Dubious network traffic processes can be detected if they have exclusive requests to social network, encode

[4]CAPTCHA: an automated challenge-response tests to ensure that the response is generated by a human

their text, or download suspicious files (executable, compressed, library). Unreliable provenance processes are processes that do not have reliable origins. These processes can be determined if they are self-reference replication, have dynamic code injection feature, or do not have digital signature.

In [21], Chu et al. proposed a classification system to determine whether tweets on Twitter belong to a human, bot, or cyborg[5]. They studied over 500,000 accounts to find the difference between human, bots, and cyborg in tweeting content and behavior. Their classifier is comprised of the following four components. (1) **Entropy Component**: to detect the regularity and periods of users' tweets. (2) **Machine Learning Component**: to detect spam tweets. (3) **Account Properties Component**: to help identify bots by checking external URL ratio in the tweets. Checking the tweet device (web, mobile, or API) helps in detection bots. (4) **Decision Maker Component**: uses the input of the previous three components to determine if the user is human, bot, or cyborg.

4 Botnet Detection and Defence

Early botnet detection methods were designed to detect botnets using their signatures [1]. Such systems cannot detect unknown botnets. Therefore, signature-based detections become available too late, after a botnet has done its initial damage. However, these detection methods are useful to avoid infection by the same old malware. Daniel et al. [53] classified botnet detection methods into passive and active detections while Trend Micro's report [2] suggests that observing the botnet behavior is an important stage in detecting botnets. It [2] divided botnets' observable behavior into three types, which are as follows:

- **Network Based Behavior**: Botmasters, while communicating with their bots (using IRC, P2P or HTTP C&C), generate observable network traffic. This traffic can be used to detect individual bots and their C&C servers. Many botnet use dynamic DNS to locate their C&C server. Therefore, abnormal DNS queries may be used to detect botnets.
- **Host Based Behavior**: While compromising computers, botnets make sequence of system/library calls (e.g. modifying system registries and/or files, creating network connections and/or disabling antivirus programs). The sequences of system/library calls made by botnets are observable for their detection.
- **Global Correlated Behavior**: The fundamental structures and mechanisms of botnets give a global behavioral characteristics, which are unlikely to change until fundamental structure and mechanism of botnets is not changed. Therefore these global observable behaviors are most valuable to detect botnets.

[5]Cyborg: is either bot-assisted human or human-assisted bot.

In this section, we discuss botnet detection methods available in the literature which do not require signatures and are capable of detecting unknown botnets. These methods are categorized into the following three main categories, provided from the most common to the least common detection methods; (1) Botnet behavior-based detection (Section 4.1): This is the most common technique used to detect botnets based on their abnormal traffic behavior, whether bot generated traffic, or bot DNS queries. The bulk of the surveyed detection methods fall under this section. (2) Botmaster traceback detection (Section 4.2): This is a less common detection method based on tracing botmasters during botnet attacks or when bots report to their botmasters. (3) Detection using virtual machines (Section 4.3): This is an expensive approach based on running virtual machines on hosts in order to detect botnets.

The reader can refer to Table 1 for comparison between various botnet detection methods. This table (will be discussed later) highlights important features of different botnet detections like; the ability to detect encrypted bots, protocol and structure independency, real-time detection ability, computational cost, etc.

4.1 Botnet Behavior Detection

Detecting botnets based on their traffic behavior is further classified into three subsections: (1) C&C Traffic Behavior: to detect abnormality in C&C traffic (C&C communication channel). (2) Bot Generated Traffic: to detect abnormality in the traffic generated because of the botnet (e.g. SPAM, scan, DDoSA, etc). (3) DNS Traffic: to detect abnormality in DNS traffic caused by the botnet.

4.1.1 C&C Traffic

Based on few IRC attributes, Mazzariello [45] modelled IRC user behavior. The author's target was to separate human user generated traffic from automated IRC traffic using language complexity, vocabulary and response times. Support Vector Machine (SVM) [66] and J48 [55] decision trees were used in the experiment. Though the experiment was a success, it was not clear if this was due to the algorithm or the dataset used.

Strayer et al. [62] used filters in pipeline manner to separate botnet traffic. The filtered botnet traffic flows are classified into IRC and non-IRC flows. Then, the algorithm looks for relationships between these flows in the correlation stage. Finally, the Topological Analysis stage takes place in three steps. First, looking for common endpoints by examining clusters IP addresses. Second, correlating traffic clusters by locating traffic in other flows that share the same endpoint. Third, flows are examined to find out which one is between the botmaster and the endpoint.

The authors in [44] propose a Machine Learning (ML) technique to detect IRC-based botnet C&C traffic. After filtering out non-IRC traffic, they tried to identify C&C

hosts by isolating the flows that likely contain C&C traffic, and by correlating them to group flows that belong to the same botnet. To reduce the flow size all port scanning traffic (i.e. TCP Syn or TCP Rst) is eliminated. To avoid software update and rich web page transfer traffic, high bandwidth traffic flows are eliminated too. Furthermore, all short lived flows (few packets or seconds) are eliminated because they can not belong to botnets. This method could result in high false positive rates and could impose considerable computational overhead.

Balram and Wilscy [9] propose a bot detection system for a single host such as PCs, which are vulnerable to phishing, data stealing and data exfiltration. The system filters out the normal traffic generated on the host and analyses the remaining suspicious traffic. Their results suggest that their real time detections system can achieve high detection rate and low false positive rate.

4.1.2 Bot Generated Traffic

Binkley et al. [12] tried to detect IRC botnets based on traffic anomaly. They considered an IRC channel to be malicious if most of its hosts are performing TCP SYN scanning. They collected three tuples for their analysis; (1) TCP SYN scanner to determine types of scanning on the network, (2) IRC channel list to determine IRC channel name and IRC hosts in the channel, (3) IRC node list to determine any IP address that belongs to any IRC channel. Using these tuples, they were able to generate reports of malicious channels, sort IRC channels by maximum number of messages, analyze host statistics of IRC channels, record IRC servers, etc. This algorithm is not signature-based and should work with unknown IRC botnets, but it cannot detect encrypted botnets.

Akiyama et al. [5] suggested that bots of the same botnet have regularities in *relationship*, *response* and *synchronization* and used these measures for botnet detection. Since all bots take commands from the botmaster, there is a one-to-many *relationship* -between the bots and their botmaster- even if there is no direct connection over a single layer. In addition, when bots receive command from the botmaster, they *respond* automatically and without mistakes. This is very different from human responses while chatting. Furthermore, when bots receive a command, they take the same action almost at the same time. For example, when the botmaster sends commands for DDoS attack, all participating bots start the attack at the same time. This *synchronization* is used as a detection metric measure. This detection method could falsely identify high-demand legitimate nodes as botmasters. Furthermore, in order to avoid detection, botnets could adjust their response time to something similar to human response.

4.1.3 DNS Traffic

Botmaster use DNS rallying to make their botnets invisible and portable. Choi et al. [20] proposed botnet detection mechanism by monitoring their DNS traffic. According to the authors, bots use DNS queries either to connect or to migrate to another C&C server. The DNS traffic has a unique feature that they define as group activity. Bots can be detected by using the group activity property of botnet DNS traffic while bots are connecting to their server or migrating to another server. There are three factors that help in distinguishing botnet DNS queries from legitimate DNS queries [20]; (1) queries to C&C servers come only from botnet members (fixed IP address space size), (2) botnet members migrate and act at the same time, which leads to temporary and synchronized DNS queries, (3) botnets usually use DDNS for C&C servers.

For a botmaster to keep its bot hidden and portable, it relies on DNS to rally infected hosts. In botnets, DNS queries can appear for many reasons. They appear during rallying process after infection, during malicious activities like spam or DoS attacks, during C&C server migration, during C&C server IP address change, or after C&C server or network link failure. Based on the aforementioned five situations of DNS query used in botnets, the authors have developed a **Botnet DNS Q Detection** algorithms, which distinguishes the botnet. This algorithm starts by building a database for DNS queries comprised of the source IP address, domain name and timestamp. Then, they group DNS query data using the domain name and timestamp field. After that, they remove redundant DNS queries. Finally, botnet DNS queries are detected using a numerically computed some similarity factor [20] This algorithm cannot detect botnets migrating to another C&C server. Therefore, they developed a **Migrating Botnet Detection** algorithm by modifying the botnet DNS query detection algorithm. Similarly, this algorithm starts by building a database for DNS queries comprised of the source IP address, domain name and timestamp. Then, it groups DNS query data using the domain name and timestamp field. After that, it removes redundant DNS queries. The next step will be to compare IP lists of different domain name with same size of IP list, because bots use two different domain names for the C&C server during migration [20].

These algorithms are protocol and structure independent and are capable of detecting unknown and encrypted botnets. However, these are not for real-time detections and have low accuracy for small networks. Furthermore, they are very sensitive to threshold values which need to be chosen very carefully to balance false positives and false negative rates.

4.2 Botmaster Traceback Detection

Most of the research on botnets focuses on detection and removal of C&C servers and bots in a network [57]. Detection of botmasters is not addressed as often because it is a more challenging task. Botmasters do not need to stay online for long periods of time. As soon as they give their command(s), they can go offline and leave the hardwork to their bots. Therefore, traceback of botmasters

Table 1: Botnet detection methods comparison

No.	Reference	Category	Note	Encrypted Bot	Protocol Independent	Structure Independent	Real Time	Low False +ve/-ve	Active System	Low Cost
1	[45]	Behavior (CCT)	Mining (IRC)					✓		✓
2	[62]	Behavior (CCT)	Honeynets (IRC)				✓			✓
3	[44]	Behavior (CCT)	ML (IRC)							
4	[9]	Behavior (CCT)	ML (HTTP)				✓	✓		∼
5	[12]	Behavior (BGT)	Anomaly					∼		∼
6	[5]	Behavior (BGT)	Anomaly							✓
7	[20]	Behavior (DNS)	Anomaly	✓	✓	✓		∼		
8	[57]	Trace-back	∼	✓	✓		✓	✓	✓	✓
9	[18]	Trace-back	∼				✓	✓	✓	✓
10	[43]	Virtual Machine	BotTracer	✓	✓	✓	✓	✓		
11	[29]	System Example	Rishi							✓
12	[32]	System Example	BotHunter		✓	✓	✓	✓		∼
13	[33]	System Example	BotSniffer	✓				✓		∼
14	[31]	System Example	BotMiner	✓	✓	✓		✓		

∼: Comment or Data are Not Available ✓: The Algorithm Has This Advantage Bhv: Behavior
CCT: C&C Traffic BGT: Bot Generated Traffic TB: Trace-back
ML: Machine Learning VM: Virtual Machine SysEx: System Example

need to be carried out in real-time. Furthermore, botmaster usually connect to their bots via stepping stones in order to hide themselves. Botmaster's C&C traffic is always low-volume and botmaster may hide it even more using encryption [57]. Ramsbrock et al. [57] proposed a unique real-time watermarking botmaster traceback technique that is resilient to encryption and stepping stones. They assumed that their tracer is in control of a bot which is capable of responding to the botmaster. Their approach depends on this bot node injecting watermark when it responses to the botmaster. The watermarking is applied as follows: (1) Random packet pairs are selected. (2) The length of these packets are adjusted by padding in a way that the length difference in each packet pair falls into a predefined range. (3) For encrypted botnet traffic, they developed a hybrid length-timing watermarking method in which the watermarking packet need to be sent at specific time. For their hybrid length-timing watermarking method to work, the assumption that network jitter is limited and knowledge of the availability time of each watermarking packet must hold.

According to Chi et al. [18], once bots receive an attack command, they attack the victim at the same time. So, they proposed a method to detect the botmaster during an attack starting from the victim and working backwards through network nodes. During this detection process, the malicious traffic is blocked router by router. Their work is based on the assumption that routers from the botmaster to the victim are fixed during a given timeframe. It is also assumed that these routers are not compromised. When the IDS that is installed on the victim detects an attack, it sends diagnose request to its edge router setting the TTL to 255. The router starts a marking mode on its interfaces and notifies the victim that it has started the marking mode. As a result, all packets coming to the victim will have their hop-count equals to zero and ID equals the ID of the router's interface that processed the packet. Now the victim sends specific diagnose request to the router's interface that processed the suspicious packets. This process is repeated till the botmaster's router is reached, and the botmaster is detected [18]. This is a real-time detection algorithm that should be capable of detecting unknown botnets. It has low computational power and low false negative rates. However, this algorithm cannot detect encrypted botnets and is designed for IRC-based botnets.

4.3 Detection Using Virtual Machine

Liu et al. [43] proposed BotTracer, a detection technique that is based on virtual machine analysis of program executions. This technique is based on the assumption that bots should have three main features; (1) the bot program starts automatically without user intervention, (2) the bot must start C&C communication, (3) the bot must launch an attack. The BotTracer begins by starting a virtual machine (on the same host) that has identical image of the host system when it starts. This virtual machine will have all autostart processes on the original host but it will be free from any human interaction. Then, the BotTracer will monitor all these processes' automatic communications to detect C&C communications. Finally, BotTracer monitors the processes -that initiated suspected C&C communication- for all system-level activities and traffic patterns. Therefore, once a bot starts malicious activity, it will be detected. This is a real-time technique that is capable of detecting unknown bots regardless of their protocol, with low false positive rate, even if the C&C traffic is encrypted. However, BotTracer has high computational requirement hence virtual machine will degrade the user performance. The BotTracer will not protect against zero day attacks where the bot stay inactive waiting for a specific date and time. Furthermore, for many bots that check for virtual machine presence, the BotTracer will not work.

4.4 Examples of Botnet Detection Systems

Botnet detection system usually use more than one detection approach. For example, a detection system could use signature, C&C and botnet generated traffic to detect botnets. Therefore, it is not feasible to put these detection systems under one classification.

1) **Rishi** [29]: this is an IRC-based botnet detection system that uses IRC channel names for detection. It monitors the network traffic for suspicious IRC channel names. Rishi starts by filtering all TCP packets containing IRC-related headers. These packets are identified by any of these keywords; NICK, JOIN, USER, QUIT and MODE. Then the following information is extracted from the captured packets; connection time, source port and IP address, destination port and IP address, IRC channel and IRC nickname. After that, nicknames are passed to the analyzer where they are scored. Higher scores reflect higher probability of botnet connections. Connections with scores higher than a preset threshold are marked as suspicious and a warning email is generated and sent to the network administrator.
2) **BotHunter** [32]: this is a botnet detection system that is based on a predefined botnet infection lifecycle. This system works in real time and can detect bots regardless of the network protocol or C&C structure as long as the botnet's behavior follows a predefined infection cycle dialog model (i.e. target scanning, infection exploitation, botnet binary downloading, botnet code execution, C&C communication and outbound scanning). BotHunter is comprised of three engines; **S**tatistical s**C**an **A**nomaly **D**etection **E**ngine (SCADE), **S**tatistical pay**L**oad **A**nomaly **D**etection **E**ngine (SLADE) and Signature Engine. SCADE is responsible for the detection of inbound and outbound scan activities. SLADE detects abnormalities in byte-distribution payloads. The signature engine is capable of detecting dialog warnings from a predefined botnet infection warning model. Furthermore, BotHunter uses a correlator to evaluate all messages (dialogs) from the anomaly detection engines (SCADE and SLADE).
3) **BotSniffer** [33]: this is a botnet detection system that is based on traffic anomaly in Local Area Networks (LANs). It is based on the assumption that all the bots respond to a command in crowds and in the same way. It looks for similarities in botnet's traffic spatial-temporal correlations. The BotSniffer algorithm is comprised of two main blocks, monitor engine and correlation engine. The monitor engine is made up of three parts; (1) Preprocessing: to reduce traffic volume using filters and whitelists. (2) C&C-like protocol matcher: to collect suspicious IRC and HTTP traffic using port-independent protocol matcher. (3) Response Detector: to detect abnormally-high scan rates, weighted failed connection rate, MX DNS query and SMTP connections. The correlation engine runs in three phases; (1) Grouping: performing 2-tuple (destination IP and port number) grouping of the nodes. (2) Groups analysis: performing Response-Crowd-Density-Check algorithm, utilizing sequential probability ratio testing, to check for dense response crowds within the groups. It also performs Response-Crowd-Density-Check algorithm looking for crowds with similar responses. (3) Botnet Alert: to issue an alert if any suspicious spatial-temporal correlation C&C is detected.
4) **BotMiner** [31]: this is a botnet detection system that is based on a framework made of three main phases; monitoring, clustering, and correlating. First, in the monitoring phase, two monitoring engines -namely C&C communication traffic engine (C-plane), and activity engine (A-plane)- are used. Each engine keeps logs of its traffic analysis. The C-plane monitors both TCP and UDP flows to determine who is talking to whom. The A-plane monitors network activities to determine who is doing what (e.g. scan, spam) by detecting abnormally-high scan rates or weighted failed connection rate. Second, in the clustering phase, the C-plane clustering is performed by looking for clusters of hosts that share same communication patterns. These clusters are victimized by calculating four random variables, namely; number of flows per hour, number of packets per hour, average number of bytes per packet and average number of bytes per second. In A-plane clustering, hosts are first clustered based on their malicious activities (e.g. scanning) then are clustered based on activity features (e.g. port number). Finally, a cross-plane correlation is performed to find intersection between the two clusters in the previous phase. The intersection means that these hosts are part of a botnet.

To summarize, as Table 1 shows, though Rishi is a low cost botnet detection system, it is a non-real-time passive system that can only detect un-encrypted IRC botnets. The other three detection systems (BotHunter, BotSniffer and BotMiner) are proposed by Gu et al. BotHunter is a real-time, protocol and structure independent detection system capable of detecting unknown botnets with few false positives/negatives. However, it is a passive system that requires botnet to follow a predefined infection cycle dialog model to be detected and it is not capable of detecting encrypted botnets. BotSniffer is capable of detecting encrypted botnets with low false positives/negatives rates, but it is protocol and structure dependent and is not a real-time system and works for LANs only. Finally, BotMiner is a passive non-real-time system. It is a low cost detection system that is capable of detecting encrypted botnets regardless of their protocol or structure with low false positives/negatives rates.

4.5 Detection Methods Summary

To summarize, in this section, we discussed botnet detection methods. These methods are categorized into three categories namely; botnet behavior-based detection, botmaster traceback detection, and detection using virtual machines.

As Table 1 shows, most of behavior-based detection methods (except DNS traffic analysis detections) are protocol dependent, cannot detect encrypted botnets and are neither real-time nor active methods. However, most of them have acceptable false positive/negative rates and acceptable computational cost. DNS traffic analysis detections are capable of detecting encrypted bots, regardless of their protocol.

Traceback detection methods are real-time active techniques that have acceptable false positive/negative rates and acceptable computational cost, but they are not structure independent.

Detection using virtual machines seems to be working for encrypted bots regardless of their protocol or structure. It is a real-time algorithm with acceptable false positive/negative rates. However this system is passive and has high computational overhead.

4.6 Defence and Post-Detection Reactions

According to [36], once a botnet is detected, it needs to be tracked and brought down. First, a copy of the bot needs to be analyzed to understand the bot behavior. To get a copy of the bot, the analyzer needs to use methods similar to honeypots. After that, the bots code needs to be studied to find out; how the communication is done within the botnet, how does new members join the botnet, and find the whereabouts of the botmaster. Finally, the source of the bot is brought down (physically) by the authorities [36].

Very few papers proposed post-detection procedures against botnet. Vogt et al. [67] suggested that superbotnets must be examined by the research community, so that defences against this threat can be developed proactively. They pointed out some weak aspects of C&C mechanism that are exhibited by traditional botnet and suggest defenders to target these weaknesses. They concluded that there are five goals that defenders could take into account to build a defence mechanism against botnets:

1) Locate or identify the adversary: At the time the adversary issues commands through the botnets' C&C, it becomes vulnerable to detection.
2) Reveal all the infected machines: If bots are pooling for botnets' commands from a known location, this polling activity can be used to reveal infected machines.
3) Command the botnet: Once the defender is familiar with the botnets' commands, (s)he can send a command to the botnet to shut it down.
4) Disable the botnet: The botnet could by paralysed by shutting down its C&C channel.
5) Disrupt Botnet Commands: By changing few bits in the adversary's commands is sufficient to disrupt adversary's control of the botnet.

5 Conclusion

Despite the fact that our knowledge about botnets is incomplete; botnets are one of the most serious threats to network security. This survey was conducted to better understand botnets and is an attempt to organize the enormous background available in this area to help researchers who are starting in this area.

In this survey, we explained botnets C&C communication, infection behaviors and models. This survey discussed some of the botnets facilitator services. Fast-Flux service networks were illustrated in great details and botnets' C&C rallying mechanisms were surveyed.

We classified botnets -based on their underlying C&C protocol- to IRC, HTTP, POP3, and P2P botnets. As a new emerging malware, social and mobile botnets' threats and potential were discussed in this survey. As mobile phones with networking capabilities have become more affordable, the threat of mobile botnets have increased. Mobile botnets could spread through SMS or MMS services. Their effect could be very damaging as the security measures against mobile botnets may not have been designed for mobile device.

Furthermore, botnet detection methods are surveyed in detail. Detection methods have been classified into three classes. First, behavior-based detection where botnets are detected using; C&C traffic behavior, bot generated traffic behavior, or DNS traffic behavior. Second, botmaster traceback detection is described. Then, a virtual machine detection method is explained. Finally, examples of botnets detection systems were explained (i.e. *Rishi*, *BotHunter*, *BotSniffer* and *BotMiner*). The survey is concluded with the botnets defence measures that should be taken after detecting a botnet.

Acknowledgments

M. Mahmoud acknowledges funding from King Fahd University of Petroleum & Minerals (KFUPM). This work was done while M. Mahmoud was doing his Ph.D. at Carleton University. M. Nir and A. Matrawy acknowledge funding from Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] "SNORT," Mar. 2006. (https://www.snort.org/)
[2] "Taxonomy of botnet threats," white paper, Trend Micro Incorporated, Nov. 2006. (http://www.cs.ucsb.edu/ kemm/courses/cs595G/TM06.pdf)

[3] "Emerging cyber threats," Technical Report, Georgia Tech. Information Security Center, Oct. 2008. (https://www.gtisc.gatech.edu/pdf/Threats_Report_2008.pdf)

[4] "The möbius tool," Apr. 2011. (https://www.mobius.illinois.edu/)

[5] M. Akiyama, T. Kawamoto, M. Shimamura, T. Yokoyama, Y. Kadobayashi, and S. Yamaguchi, "A proposal of metrics for botnet detection based on its cooperative behavior," in *International Symposium on Applications and the Internet Workshops*, pp. 82–82, Jan. 2007.

[6] I. Asrar, "Could Sexy Space be the Birth of the SMS Botnet?", July 2009. (http://www.symantec.com/connect/blogs/could-sexy-space-be-birth-sms-botnet)

[7] P. Bächer, T. Holz, M. Kötter, and G. Wicherski, "Know your Enemy: Tracking Botnets," Oct. 2008. (http://www.honeynet.org/papers/bots)

[8] M. Bailey, E. Cooke, F. Jahanian, Y. Xu, and M. Karir, "A survey of botnet technology and defenses," in *Cybersecurity Applications Technology Conference For Homeland Security*, pp. 299–304, Mar. 2009.

[9] S. Balram and M. Wilscy, "User traffic profile for traffic reduction and effective bot c&c detection," *International Journal of Network Security*, vol. 16, pp. 46–52, Jan. 2014.

[10] J. Baltazar, J. Costoya, and R. Flores, "The real face of koobface: The largest web 2.0 botnet explained," Technical Report, Trend Micro Incorporated, July 2009.

[11] P. Barford and V. Yegneswaran, "An inside look at botnets," in *Advances in Information Security*, vol. 27, pp. 171–191, Springer, Mar. 2007.

[12] J. R. Binkley and S. Singh, "An algorithm for anomaly-based botnet detection," in *Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet*, pp. 7–7, Berkeley, CA, USA, 2006.

[13] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The socialbot network: When bots socialize for fame and money," in *Proceedings of the ACM 27th Annual Computer Security Applications Conference*, pp. 93–102, New York, NY, USA, 2011.

[14] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "Design and analysis of a social botnet," *Computer Networks*, vol. 57, no. 2, pp. 556–578, Feb. 2013.

[15] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "Key challenges in defending against malicious socialbots," in *Proceedings of the 5th USENIX conference on Large-Scale Exploits and Emergent Threats (LEET'12)*, pp. 12–12, Berkeley, CA, USA, 2012.

[16] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton, "Real-time detection of fast flux service networks," in *Proceedings of the IEEE Cybersecurity Applications & Technology Conference for Homeland Security*, pp. 285–292, Washington, DC, USA, Mar. 2009.

[17] J. A. Chandler, "Liability for Botnet Attack," *Canadian Journal of Law and Technology*, vol. 5, pp. 13–25, Mar. 2006.

[18] Z. Chi and Z. Zhao, "Detecting and blocking malicious traffic caused by IRC protocol based botnets," in *IFIP International Conference on Network and Parallel Computing*, pp. 485–489, Dalian, China, Sep. 2007.

[19] K. Chiang and L. Lloyd, "A case study of the rustock rootkit and spam bot," in *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, pp. 10–10, Berkeley, CA, USA, 2007. Association.

[20] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in dns traffic," in *The 7th IEEE International Conference on Computer and Information Technology*, pp. 715–720, Oct. 2007.

[21] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on twitter: Human, not, or cyborg?" in *Proceedings of the ACM 26th Annual Computer Security Applications Conference (ACSAC'10)*, pp. 21–30, New York, NY, USA, 2010.

[22] Z. Coburn and G. Marra, "Real boy: Believable twitter bots," July 2012.

[23] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, pp. 6–6, Berkeley, CA, USA, 2005.

[24] X. Cui, B. Fang, L. Yin, X. Liu, and T. Zang, "Andbot: Towards Advanced Mobile Botnets," in *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats (LEET'11)*, pp. 11–11, Berkeley, CA, USA, 2011.

[25] D. Dagon, C. Zou, and W. Lee, "Modeling botnet propagation using time zones," in *Proceedings of the 13 th Network and Distributed System Security Symposium NDSS*, 2006.

[26] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," pp. 268–273, Los Alamitos, CA, USA, June 2009.

[27] D. Fisher, "Storm, nugache lead dangerous new botnet barrage," Dec. 2007. Online Article.

[28] A. R. Flø and A. Jøsang, "Consequences of Botnets Spreading to Mobile Devices," in *Proceedings of the 14th Nordic Conference on Secure IT Systems (NordSec 2009)*, Oct. 2009.

[29] J. Goebel and 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*, pp. 8, Berkeley, CA, USA, 2007.

[30] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and 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*, pp. 1, Berkeley, CA, USA, 2007.

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

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

[33] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of 16th Annual Network and Distributed System Security Symposium (NDSS'08)*, Reston, VA, USA, February 2008.

[34] K. J. Higgins, "Smartphone weather app builds a mobile botnet," Mar. 2010.

[35] T. Holz, C. Gorecki, K. Rieck, and F. Freiling, "Measuring and detecting fast-flux service networks," in *The 15th Network and Distributed System Security Symposium (NDSS'08)*, Reston, VA, USA, Feb. 2008.

[36] T. Holz, M. Steiner, F. Dahl, E. biersack, and 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*, pp. 1–9, Berkeley, CA, USA, 2008.

[37] T. Hwang, I. Pearce, and M. Nanis, "Socialbots: Voices from the fronts," *interactions*, vol. 19, pp. 38–45, Mar. 2012.

[38] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, pp. 7–7, Berkeley, CA, USA, Apr. 2007.

[39] E. Kartaltepe, J. Morales, S. Xu, and R. Sandhu, "Social network-based botnet command-and-control: Emerging threats and countermeasures," in *Applied Cryptography and Network Security*, LNCS 6123, pp. 511–528, Springer-Verlag, 2010.

[40] C. Li, W. Jiang, and X. Zou, "Botnet: survey and case study," in *Fourth International Conference on Innovative Computing, Information and Control*, pp. 1184–1187, Dec. 2009.

[41] C. Y. Liu, C. H. Peng, and I. C. Lin, "A survey of botnet architecture and botnet detectection techniques," *International Journal of Network Security*, vol. 16, no. 2, pp. 81–89, Mar. 2014.

[42] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng, and J. Zhang, "Botnet: classification, attacks, detection, tracing, and preventive measures," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, pp. 11, 2009.

[43] L. Liu, S. Chen, G. Yan, and Z. Zhang, "Bottracer: Execution-based bot-like malware detection," in *Information Security*, pp. 97–113, 2008.

[44] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Using machine learning technliques to identify botnet traffic," in *The 31st IEEE Conference on Local Computer Networks*, pp. 967–974, Nov. 2006.

[45] C. Mazzariello, "IRC traffic analysis for botnet detection," in *Fourth International Conference on Information Assurance and Security (ISIAS'08)*, pp. 318–323, Naples, Italy, Sep. 2008.

[46] S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, and N. Borisov, "Stegobot: A covert social network botnet," in *Information Hiding*, LNCS 6958, pp. 299–313, Springer-Verlag, 2011.

[47] J. Nazario, "Blackenergy ddos bot analysis," Technical Report, Arbor Networks, Oct. 2007.

[48] J. Nazario, "Twitter-based botnet command channel," Aug. 2009.

[49] J. Nazario and T. Holz, "As the net churns: Fast-flux botnet observations," in *The 3rd International Conference on Malicious and Unwanted Software*, pp. 24–31, Oct. 2008.

[50] J. Oikarinen and D. Reed, "Internet relay chat protocol," RFC 1459, May 1993.

[51] M. Overton, "Bots and botnets: risks, issues and prevention," in *proceedings of virus bulletin Conference*, Virus Bulletin, Oct. 2005.

[52] E. Passerini, R. Plaeari, L. Martignoni, and D. Bruschi, "Fluxor: Detecting and monitoring fast-flux service networks," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, LNCS 5137, pp. 186–206, July 2008.

[53] D. Plohmann, E. Gerhards-Padilla, and F. Leder, "Botnets: measurement, detection, disinfection and defence," in *ENISA Workshop* (Giles Hogben, ed.), Mar. 2011.

[54] P. Porras, H. Saidi, and V. Yegneswaran, "An analysis of the ikee.b (duh) iphone botnet," Technical Report, SRI International, CA, 94025, USA, Dec. 2009.

[55] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., 1993.

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

[57] D. Ramsbrock, X. Wang, and X. Jiang, "A first step towards live botmaster traceback," in *Recent Advances in Intrusion Detection*, LNCS 5230, pp. 59–77, Springer-Verlag, 2008.

[58] E. Van Ruitenbeek and W. H. Sanders, "Modeling peer-to-peer botnets," in *Fifth International Conference on Quantitative Evaluation of Systems*, pp. 307–316, Sep. 2008.

[59] Y. H. Shin and E. G. Im, "A survey of botnet: consequences, defenses and challenges," in *The fourth Joint Workshop on Information Security (JWIS'09)*, Kaohsiung, Taiwan, Aug. 2009.

[60] K. Singh, A. Srivastava, J. Giffin, and W. Lee, "Evaluating email's feasibility for botnet command and control," in *IEEE International Conference on Dependable Systems and Networks with FTCS and DCC*, pp. 376–385, June 2008.

[61] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, "Analysis of the storm and nugache trojans: P2p is here," *The USENIX Magazine*, vol. 32, pp. 18–27, Dec. 2007.

[62] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting botnets with tight command and control," in *Proceedings the 31st IEEE Conference on Local Computer Networks*, pp. 195–202, Cambridge, MA, Nov. 2006.

[63] V. L. Thing, M. Sloman, and N. Dulay, "A survey of bots used for distributed denial of service attacks," in *New Approaches for Security, Privacy and Trust in Complex Environments*, vol. 232/2007 of *IFIP International Federation for Information Processing*, pp. 229–240, Boston, USA, Nov. 2007.

[64] P. Traynor, M. Lin, M. Ongtang, v. Rao, T. Jaeger, P. McDaniel, and 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)*, pp. 223–234, New York, NY, USA, 2009.

[65] P. Traynor, P. McDaniel, and T. La Porta, "On Attack Causality on Internet-Connected Cellular Networks," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium (SS'07)*, pp. 21:1–21:16, Berkeley, CA, USA, 2007.

[66] V. Vapnik, S. E. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems*, vol. 9, pp. 281–287, 1996.

[67] R. Vogt, J. Aycock, and M. J. Jacobson, "Army of botnets," in *Proceedings of Network and Distributed System Security Symposium (NDSS'07)*, pp. 111–123, Reston, VA, USA, Feb. 2007.

[68] P. Wand, S. Sparks, and 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*, pp. 2, Berkeley, CA, USA, 2007.

[69] Z. Zhu, G. Lu, Y. Chen, Z. J. Fu, P. Roberts, and K. Han, "Botnet research survey," in *32nd Annual IEEE International Conference on Computer Software and Applications*, pp. 967–972, Aug. 2008.

Muhammad Mahmoud is an Assistant Professor at King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia. He received the Ph.D. degree in electrical and computer engineering from Carleton University, Ottawa, Canada. His research interests include network security, and Communication Network Protocols. His research has been supported by KFUPM.

Manjinder Nir is currently a fourth year Ph.D. candidate in the Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada. He received B.Tech. and M.Tech. degrees in Electronics and Communication Engineering from Punjab Technical University, Punjab, India. His research interests include computer networking and pervasive computing.

Ashraf Matrawy is an Associate Professor and the Associate Director of the School of Information Technology at Carleton University. He received the Ph.D. degree in electrical engineering from Carleton University. He is a senior member of the IEEE, serves on the editorial board of the IEEE Communications Surveys and Tutorials journal, and has served as a technical program committee member of a number of international conferences. His research interests include reliable and secure computer networking. His research has been supported by CFI/ORF, NSERC, OCE, Alcatel-Lucent Canada, and Solana Networks.