A Study of Botnets: Systemization of Knowledge and Correlation-based Detection

By
Shehar Bano
2010-NUST-MSCCS-23

Supervisor
Dr. Fauzan Mirza
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Computer and Communication Security (MS CCS)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(October 2012)

Approval

It is certified that the contents and form of the thesis entitled "**A Study of Botnets: Systemization of Knowledge and Correlation-based Detection**" submitted by **Shehar Bano** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Fauzan Mirza**
Signature: _____

Date: _____

Committee Member 1: **Dr. Zahid Anwar**

Signature: _____
Date: _____

Committee Member 2: **Dr. Affan A. Syed**

Signature: _____
Date: _____

Committee Member 3: **Dr. Syed Ali Khayam**

Signature: _____
Date: _____

To Ami & Baba for inspiring me to reach for the stars
and
Shahzad & Shahmeer for making the journey worthwhile.

Certificate of Originality

I here by declare that the research paper titled **A Study of Botnets: Systemization of Knowledge and Correlation-based Detection** my own work and to the best of my knowledge. It contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST or any other education institute, except where due acknowledgment, is made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the projects design and conception or in style, presentation and linguistic is acknowledged. I also verified the originality of contents through plagiarism software.

Author Name: **Shehar Bano**
Signature: _____

Acknowledgment

I am greatly indebted to my university NUST for providing me with several opportunities for both academic and personal growth. The teaching staff was always available for counseling and university administration was very cooperative.

Next, I would like to express my deep gratitude to Dr. Fauzan Mirza (NUST), Dr. Syed Ali Khayam (xFlow research) and Dr. Affan A. Syed (FAST) for their guidance, support and useful input throughout the course of this work. I wish to thank Dr. Syed Ali Khayam for his mentorship, support and availability. His enthusiasm has always been a great motivation. I am extremely grateful to Dr. Affan A. Syed, for admitting me to his research group SysNet at FAST. I have benefited greatly from his useful critique, assistance and feedback. In particular, I am indebted to him for patiently reviewing drafts of research papers I wrote while working on this thesis. I wish to thank Dr. Fauzan Mirza for sharing useful insights that greatly helped in shaping and refining this work. I would also like to thank my committee member Dr. Zahid Anwar. His ideas and comments have significantly added to my work.

My gratitude also extends to my colleagues and friends at WisNet lab (NUST) and SysNet lab (FAST) for sharing knowledge, feedback and support. In particular, I wish to thank Kamran Riaz Khan for our discussions on Linux, Python and programming in general. My thanks are also due to Naurin Rashid Ramay for sharing her knowledge of data processing and statistical concepts. A special thanks to Ayesha Binte Ashfaq for helping me outline goals and tracking my progress. I also wish to thank Badar and Tariq, the vigilant lab administrators at WisNet and SysNet respectively, for providing me with the resources to work smoothly.

I must thank the wonderful Bro team at Berkeley, California for providing support during the development of BotFlex, our botnet detection tool. In particular, I wish to thank Seth Hall for sharing useful insights about intrusion detection and Bro's scripting language.

Last, but not the least, I thank my family for their love and support. In

particular, I wish to express my deep gratitude to my mother and mother-in-law for looking after my son when I couldn't. Thanks to Shahzad and Shahmeer for cheerfully putting up with my hectic schedule. This dissertation could not have been completed without you.

This work was supported by Pakistan National ICT R&D Fund.

Table of Contents

List of Figures

List of Tables

Abstract

Botnets have been the cause of some of the gravest cyberthreats in recent times. Despite research and commercial efforts to curb botnets, they continue to grow in size and sophistication. As a response to this persistent yet rapidly-evolving threat, hundreds of scientific reports have been published on botnet architectures, economics, detection, defense and future trends. Resultantly, attackers have become more aggressive by using robust communication mechanisms, decentralized architecture and a number of elaborate evasion techniques. An obvious by-product of this perennial arms race is that the field of botnets has grown quite complex. This complexity is two-fold. Firstly, the vast body of literature on botnets remains largely unstructured. Secondly, a collaborative platform to carry out botnet research is sorely missing.

In this thesis, we address the issues identified above by first structuring the knowledge in the area of botnets, and then developing a botnet detection tool. We present three taxonomies related to botnet behaviors/architectures, detection mechanisms, and defense strategies. We assert that our proposed taxonomies aid in visualizing the diversity in botnet research, and in making informed decisions when devising new detection and defense mechanisms. Next, we present BotFlex — an open-source, extensible, tool for botnet detection. We use BotFlex to analyze hundreds of GBs of a home ISP's traffic, with ground truth provided by a commercial security company. We observe true and false positive rates of 89.6% and 28.6%, respectively. These are very positive results for a system currently using a handful of features and decisions elements from existing literature. We run BotHunter, a closed-source bot detection tool, over the same data set and observe similar detection rates and performance, validating our implementation against the only freely available tool.

Chapter 1

Introduction and Motivation

'Would you tell me, please, which way I ought to go from here?' 'That depends a good deal on where you want to get to,' said the Cat. 'I don't much care where ' said Alice. 'Then it doesn't matter which way you go,' said the Cat.

— Lewis Carroll, "Alice's Adventures in Wonderland"

Botnet is a highly evolved piece of malware which incorporates components of viruses, worms, spyware and other malicious software [2]. A botnet is a group of remotely controlled, compromised machines. The combined power of these machines can be used for launching a variety of financially motivated crimes such as DDoS attacks, information stealing and spam among others. In addition to financial incentives, botnets are increasingly being used for cyber espionage. The emergence of GhostNet [3], Stuxnet [4] and Flame [5] have marked the beginning of a new era in cyber warfare fueled by botnets.

Despite research and commercial efforts, botnets continue to grow in size and sophistication. Moreover, black markets for renting a botnet have enabled its prevalence irrespective of misfeasor's skill-set. A quarter of the world's computers were estimated to form part of a botnet half a decade ago [6]. The situation is no different today as botnets are still a primary threat to security of the internet [7]. In the last year alone, ShadowServer [8] tracked about 6000 C&C servers, with an average of 2200 active C&C servers per day.

1.1 Motivation

Botnets pose an alarming–and arguably *the* most potent– threat to the security of Internet-connected users and systems. As a response to this per-

sistent yet rapidly-evolving threat, hundreds of scientific reports have been published on botnet architectures, economics, detection, defense and future trends. Resultantly, attackers have become more aggressive by using robust communication mechanisms, decentralized architecture and a number of elaborate evasion techniques. An obvious by-product of this perennial arms race is that the field of botnets has grown quite complex. This complexity is two-fold. Firstly, the vast body of literature on botnets remains largely unstructured. Secondly, a collaborative platform to carry out botnet research is sorely missed. We briefly discuss the motivation for this thesis in the light of the discrepancies identified above.

Realizing the gravity of the botnet threat, the academic community has produced scores of research papers and reports to explain botnet behaviors, topologies, sizes, detection parameters, defense strategies, and future trends. The bulk of information in this area calls for a comprehensive classification of the botnet phenomenon as well as its detection and defense mechanisms. While some surveys and taxonomies of botnet behavior, detection and defense have been proposed [9, 10, 11, 12, 13, 14], these efforts only address a subset of the entire botnet phenomenon. A systematic and comprehensive taxonomy of the botnet phenomenon promises to aid in visualizing the diversity in botnet research, and in making informed decisions when devising new detection and defense mechanisms.

Furthermore, in view of the academia's continued interest in the botnet phenomenon, it is surprising that to date *no* open source botnet detection tool exists. The options available to a researcher are either to purchase services of commercial security companies [15, 8, 16], select from a very small number of closed source tools [17], or implement existing algorithms from scratch. These difficult to realize options have inadvertently resulted in a culture where botnet research is carried out in isolation, without much comparison with other tools or techniques. Thus, botnet research fails to experience the uplift that results from fusion of ideas on an open source platform. A universally accessible platform of this kind offers the additional advantage of channeling time and resources towards improvement of existing technology rather than reproducing what already exists in literature.

1.2 Contribution

In this thesis, we set out to systemize existing knowledge of botnet behavior, detection and defense, and subsequently use this knowledge to develop a flexible, open-source botnet detection tool.

In accordance with our problem statement, we lay out some research

objectives and the corresponding contributions of this thesis.

- **Objective 1:** To systemize existing knowledge related to botnets to better understand their strengths and weaknesses.

  Contribution: We present a systematic analysis of the botnet threat from three aspects: botnet behaviors/architectures, detection mechanisms, and defense strategies. Our first taxonomy aims to demystify the adversary by exploring the botnet phenomenon from different angles, such as propagation, rallying, C&C and purpose. Moreover, we have also enumerated evasion mechanisms employed by a botnet for obscuring its different parts. The second taxonomy classifies botnet detection approaches. We have introduced the notion of 'dimension' in structuring mechanisms relevant to botnet detection. This empowers a defender to evaluate botnet detection approaches by different yardsticks. Our third taxonomy provides a systematic analysis of botnet defense mechanisms. Together, these three taxonomies provide a comprehensive framework that could be utilized to understand the botnet problem and its solution space. The insights gained from this characterization can be used by defenders to identify shortcomings in existing approaches for botnet detection and defense and devise improved strategies.

- **Objective 2:** To extrapolate the knowledge gained from devising the botnet taxonomies to arrive at previously unknown, interesting facts.

  Contribution: The proposed botnet taxonomies reveal an inherent connection between botnet behavioral features and detection approaches. We show that the selection of botnet behavioral features (from our first taxonomy) have a direct impact on the accuracy of the detection approaches (from the second taxonomy). Network security research and products can use this information to evaluate the efficacy of different detection approaches for specific threats. Furthermore, complementary detection approaches can be combined to devise an integrated botnet detection and defense solution.

- **Objective 3:** To develop an extensible open source tool for botnet detection to facilitate research in this area.

  Contribution: This thesis presents BotFlex — an open-source, extensible, tool for botnet detection. In order to evolve with the rapidly-changing botnet threat landscape, we design BotFlex to achieve the goals of *flexibility* in addition and removal of detection features; *extensibility* in adding new decision and correlation elements; support for

forensics and analysis with the capability to *react to events in real-time*. Furthermore, the architecture of BotFlex allows significant flexibility to a researcher to select and compare the performance of a botnet detection system with different combination of techniques and policies.

- **Objective 4:** To compare and evaluate the results of the proposed botnet detection tool with other similar tools using real world datasets with known botnet traffic ("*groundtruth* traffic").

  Contribution: We use BotFlex to analyze hundreds of GBs of a home ISP's traffic, with ground truth provided by a commercial security company. We observe true and false positive rates of 89.6% and 28.6%, respectively. These are very positive results for a system currently using a handful of features and decisions elements from existing literature. We ran BotHunter [17], a closed-source bot detection tool, over the same data set and observed similar detection rates and performance, validating our implementation against the only freely available tool.

  More importantly, we reveal interesting insights that result from parameter tuning and decision policy variations. For instance, we observe a true and false positive rate of 87.3% and 26.6%, respectively, when using just C&C blacklist matching. These rates change to 40% and 3.4% when we activate all detection sensors and declare a bot if a C&C blacklist match is substantiated by another evidence. We also observe, through manual analysis, that 5% of hosts declared false positives based on our groundtruth list were actually malicious hosts. These observations, possible due to the tunability of BotFlex, point to interesting aspects of relying solely on C&C blacklists. Such insights substantiate our thesis that the security research community can benefit significantly from an open source platform for botnet research.

Chapter 2

Background and Literature Review

Every experience is a paradox in that it means to be absolute, and yet is relative; in that it somehow always goes beyond itself and yet never escapes itself.

— T. S. Eliot

A botnet is a collection of compromised machines (*bots*) receiving and responding to commands from a server (the *C&C server*) that serves as a rendezvous mechanism for commands from a human controller (the *botmaster*) (Fig. 6.1). To evade detection, the botmaster can optionally employ a number of proxy machines, called *stepping-stones*, between the C&C server and itself. Machines are infected by means of a malicious executable program referred to as *bot binary*. Bots belonging to the same botnet form the *bot family*. The ultimate goal of a botnet is to carry out malicious activities or attacks on behalf of its controller.

The foundation of botnets was laid by remotely controlled malicious tools, such as *trinoo* [18], *PrettyPark* and *Sub7* in 1999. This was followed by constant innovation in the area of botnets. A number of IRC-based botnets emerged in the following years, such as GTBot, SDBot, Agobot, Spybot and Rbot. In particular, SDBot inspired a number of botnets by making its source code widely available. The susceptibility of IRC traffic to detection reinforced the need for botnet creators to use a more agile and difficult to block protocol for C&C communication. Thus, the trend in botnet C&C gradually shifted from IRC to other protocols, such as HTTP (Rustock, Waledac, Torpig, ZeuS, Pushdo/Cutwail, Bobax) and p2p (Slapper [19], Sinit [20], Phatbot [21], Storm [22] and Nugache [23]). The desire to camouflage C&C

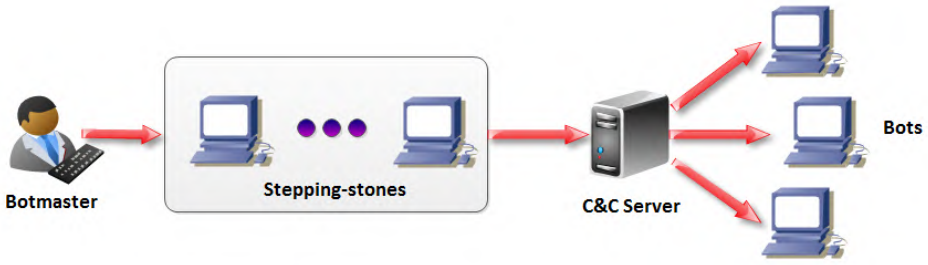


Figure 2.1: Structure of a typical botnet.

communication as regular Internet traffic has led botnets (Whitewell, Koobface) to (ab)use the user-generated content of Web 2.0. Blogs and Really Simple Syndication (RSS) feeds. In recent times, a number of highly sophisticated botnets have emerged, such as TDL-4, Duqu and Gauss. Flame [5] incorporated an MD5 collision attack to spoof Microsoft's digital certificates.

We defer more detailed discussion of botnets to chapter 3 and provide an overview of related work next.

2.1 Related Work

In this section, we provide a summary of notable work carried out in this area so far. Our thesis concerns itself with two tasks, i.e., systemization of knowledge and development of a botnet detection tool, to achieve the common goal of facilitating botnet research. We further break down related work with respect to the subgoals identified above.

2.1.1 Related Work–Botnet Taxonomy

A great deal of previous work has focused on different aspects of botnets, their detection and defense. A common thread to all these efforts is that they have touched a part of the entire botnet phenomenon. Moreover, a systematic taxonomy that organizes all this information is sorely missing. We give an overview of previous work related to botnet behavior, their detection and defense.

Cooke et al. [24] in their pioneer work explained C&C structures and proposed a novel structure which they called 'random'. A brief discussion of this structure can be found in section 3.7.2. A branch of research seeks to demystify certain aspects of botnet operation, such as evasion techniques [25, 26], C&C structures [13] and propagation [27]. Nazario [12] highlighted

the multifaceted nature of botnets and maintained that this should be reflected in related taxonomies. His taxonomy was based on factors such as network structure, language of bot binary, features (attacks,server,proxy) and propagation strategies. This relates to *Purpose*, *Topology* and *Propagation* in our taxonomy of botnet features. Their work does not take into account C&C characteristics, botnet evasion techniques and rallying mechanisms. Dagon [9] emphasized that the aim of taxonomy should be identification of detection opportunities. The taxonomy classified C&C on the basis of nature of C&C resources (public/private), RFC compliance, and activity level (how often the bots contact the botmaster). The concept of protocol agnostic detection is related to secondary data *Correlation* and *Network-based* semantic bot detection in section 4.2.2. The taxonomy only highlights C&C channels and the remaining work provides a general discussion of rallying, detection and response strategies. Trendmicro and SANS published detailed reports [10, 28] explaining various botnet components. Dagon et al. [11] classified botnet structures into three models and analyzed each model from the perspective of response mechanisms.

A subset of botnet research revolves around botnet detection. Bailey et al. [29] provided a survey of existing botnet research with emphasis on botnet technology and defense. The survey examined botnet detection methods on the basis of an interesting dimension, data sources. The data sources included DNS, Netflow, packet tap, address allocation, honeypot and host data. The authors posited that this information would be a useful metric for choosing the detection method that works on the data readily available to the interested party. In our botnet detection taxonomy, we have discussed the aspect of data source in all the subsections and made clear demarcation where absolutely necessary. Classification of detection techniques on the basis of cooperative behavior, signature matching and attack behavior map to our *Correlation* and *Syntactic* detection (sections 4.2.2, 4.3.2) and secondary data *Correlation* in section 4.2.2. A similar effort classified botnet detection techniques as signature-based, anomaly-based, DNS-based and mining-based [30]. Both of these lack the depth and organization that is characteristic of a taxonomy and is merely a high level categorization to help readers understand existing work in this area. Zeidanloo et al.[31] presented a taxonomy of botnet detection. Honeypots have been shown as a class of botnet detection. Honeypots are functional to understanding the botnet operation and assessing vulnerability of a network to the botnet threat, yet it is important to understand that they do not identify bots or bot families per se. The other class, Intrusion Detection System (IDS), is along the lines of established taxonomy for IDS [32] with the additional notion of activity in the context of network monitoring. The taxonomy provides a good overview of

the existing detection methods but falls short in distinguishing between the different targets of botnet detection methods. Researchers [33] established a broad framework for evaluating the evadability of automated bot/botnet detection mechanisms. This bears slight similarity with our overview of the effect of different botnet features on botnet detection in subsection 4.5.2. However, their work focuses on evasion mechanisms specific to six research papers related to automated botnet detection. Moreover, they do not discuss weaknesses in botnets that defenders can use to improve their detection approaches.

Finally, a section of research concerns itself with the area of botnet defense methods. MAAWG (Messaging Anti-Abuse Working Group) [34] published a comprehensive report about how large scale bot infections can be mitigated in residential networks. A number of reports provide a detailed discussion of policies [35], detection, measurement, disinfection and defense [14] against botnets. It discusses defense mechanisms against botnets in great detail, however there is no elaborate structure in the way this information is presented. A number of papers [36, 37] give useful insights into botnet defense but do not span the entire gamut of possible botnet countermeasures.

2.1.2 Related Work–Botnet Detection Tools

A number of tools have been written for botnet detection. As botnets represent a multifaceted and distributed threat, these tools detect from different aspects, for example, deployment location (host-based, network-based), depth of analysis (header-based, Deep Packet Inspection (DPI)), scope of analysis (single bot behavior, group behavior), detection methodology (signature-based, anomaly / behavior-based). We now briefly cover each of these areas related to our work.

Signature-based approaches identify botnets by comparison with known patterns of botnet C&C communication extracted from observed samples. Rishi [38] detects IRC based bots by using regular expressions as signatures to identify bot-like IRC nicknames. Snort [39] is an open source IDS that boasts of an extensive library of malware-related signatures. Some of these signatures are specific to botnets and are useful in detection of known threats.

Correlation approaches detect botnets by attributing different patterns in network activity to botnets. Correlation can be further classified as horizontal or vertical. Horizontal correlation is based on the idea that botnets are coordinated malware, and therefore exhibit similarity in behavior and/or communication. BotMiner [40] detects bots in a network by clustering similar communication and malicious activity patterns and then performs cross-cluster correlation. BotSniffer [41] detects bots by looking for group behavior

in network activities as bots are pre-programmed to respond to commands from the C&C server in a certain way. Other techniques [42, 43, 44, 45] detect botnets by clustering flows with similar characteristics such as bandwidth, duration, packet timing, external network, payload, platform used by internal hosts etc.

In vertical correlation, the behavior of individual hosts is mapped to an established model of bot behavior. Rajab et al [46] described a bot as a multifaceted phenomenon comprising of host exploit, malicious binary download and C&C communication. BotHunter [47] built on the previously mentioned model of the typical bot lifecycle and added two additional phases (inbound scanning and outward attack propagation) to it.

BotTracer [48] employs virtual machine techniques to detect botnets based on three invariants in the life of a botnet; automatic startup of bot with no user intervention (startup), C&C establishment (preparation) and attack.

A vein of research ties coordinated malicious behavior or its side effects observable at higher network elements to botnets. Zhuang et al [49] characterized botnets from spam email records collected at popular email service providers. Botnets can also be detected by correlating DNSBL queries that they perform as a precursor to launching a spam campaign [50]. Analysis and correlation of anomalous DDNS [51] and DNS traffic [52] can also lead to botnet detection.

Some tools rely on active traffic injection to elicit responses from bots that are likely to give them away. BotProbe [53] detects bots by injecting packets into chat-like network traffic. It leverages the fact that unlike humans, bots are intolerant to typographic mistakes in C&C commands and their response to 'correct' commands is deterministic.

All of the above mentioned tools, however, have one of two deficiencies. Either they are designed to focus on detecting botnets through a particular approach, and/or these tool are not released in open source to allow extending them to an evolving threat landscape. BotFlex is released as an open source, and extensible, platform where these two deficiencies can be removed simultaneously.

Chapter 3

Taxonomy of botnet behavior

Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?

— T. S. Eliot

This chapter presents the first taxonomy in a series of three taxonomies pertaining to different aspects of botnets: behavior, detection mechanisms and defense strategies. Our first taxonomy classifies botnet features based on their behavior. Different phases in the life cycle of a botnet (such as host infection, rallying, C&C communication) provide a high-level behavioral view. This behavioral landscape is complicated by evasion techniques and topological choices of botnet creators. We provide an extensive overview of all of these behavioral aspects, and posit that most of the past, present and future botnets can be entirely described and categorized with the help of this taxonomy.

3.1 Overview

Botnets are often described as complex malware. As such, a botnet can be characterized by various metrics. A number of previous works [10, 28, 9] attempt to explain botnets with respect to different behavioral features. However, these studies do not cover the entire breadth of possible botnet behavior. Thus, a comprehensive and systematic study of botnet behavior is sorely missing.

In this work, we propose a taxonomy of botnet behavior. We contend that any given botnet can be entirely described with the help of our taxonomy. To fully understand the botnet phenomenon, it is important to systematically

explain different features related to botnet behavior [1]. A systematic study of this kind entails the following benefits. First, it is easier to visualize a botnet using our taxonomy as a reference. Secondly, our taxonomy can be used as a benchmark to compare different botnets and understand their similarities and differences.

We explain the typical botnet life cycle with reference to our proposed taxonomy of botnet features (Fig. 3.1). From the point of view of a bot, the infection starts with execution of the bot binary on the victim machine. Bot binary is transported to the victim machine using a *Propagation* mechanism (section 3.2). The next step is to contact the C&C server and announce its presence. This is called call-home mechanism or *Rallying* (section 3.3). Rallying marks the establishment of a *C&C* channel through which the bot receives updates and commands (section 3.4). Based on how C&C communication takes place, the botnet forms an overlay *Topology* (section 3.7). The newly recruited bot then waits for commands to serve the actual *Purpose* of the botnet (section 3.5) and optionally spread the infection to other hosts using propagation mechanisms. An important consideration through all the botnet operations is *Evasion* (section 3.6). Different mechanisms are employed to ensure that the bot binary, C&C communication, C&C server(s) and botmaster may not be trivially detected.

3.2 Propagation

One of the primary goals of a botnet is to continuously increase its footprint in terms of number of bots. Most bot binaries have in-built mechanisms to facilitate its propagation to other hosts. Depending on the degree of required human intervention, propagation mechanisms can be broadly classified as active and passive.

3.2.1 Active

In this mode of propagation, the botnet is capable of locating and infecting other hosts without any (human) user intervention.

A predominant active propagation mechanism is *Scanning*. A scanning bot probes other hosts in the network looking for one or more vulnerabilities to exploit. The vulnerability exploit helps the botnet in gaining administrative privilege to the victim machine which is typically followed by installation of the bot binary and eventually C&C communication ensues. Some botnets

[1] For the rest of this document, the term botnet features refers to botnet behavioral features.

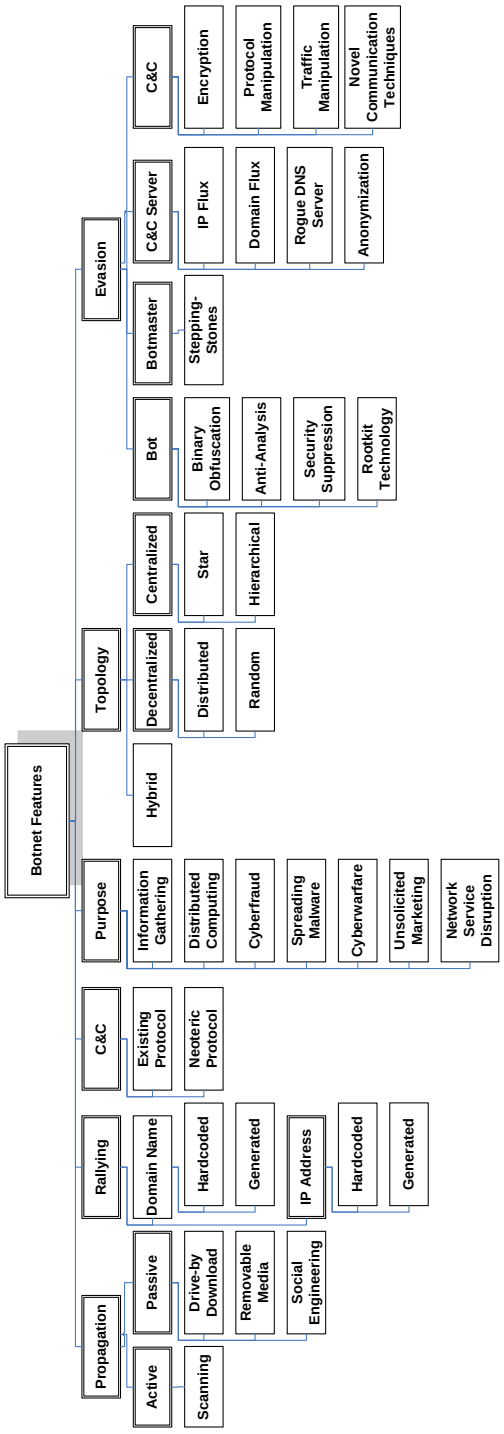


Figure 3.1: Taxonomy of botnet behavior.

borrow their propagation tactics from worms. They make copies of themselves and propagate automatically, aiming to infect as many hosts as possible. The worm may not necessarily include the main bot binary, however, it prepares ground for future bot binary installation. Both Storm [54] and Sinit [20] exhibited worm-like behavior for propagation. However, Sinit's use of random scanning for peer-discovery, instead of a well-defined bootstrap process, resulted in poor overall network connectivity.

3.2.2 Passive

Passive propagation requires some level of user intervention. Next we describe the three most widely-used passive propagation mechanisms.

Drive-by Download Some websites have been either compromised, or specially crafted for automated installation of bot malware on machines of visitors. These websites contain cleverly crafted 'active content' (such as JavaScript or ActiveX controls) which automatically initiate download of the malware to the visitor's machine.

Infected Media Botnets can also find new bots to recruit by sharing of infected media (e.g., USB hard drives). This is a powerful method of propagation as it can potentially spread the infection to private networks not connected directly to the Internet. Stuxnet is a highly targeted botnet which allegedly hurt Iran's nuclear program by causing sensitive equipment to malfunction. It initially infected Iran's uranium enrichment infrastructure through this mode of propagation [55].

Social Engineering An underestimated but powerful method of bot recruitment is through social engineering. Social engineering encompasses all methods that entice the user to willingly download the bot binary. Some botnets exploit the culture of trust prevalent in social networks by posting catchy messages from users' (hijacked) accounts. For example, Koobface [56] tricked users into clicking on a link that pointed to a fake YouTube website. The user was then asked to download specific executable file to watch the video which was actually malware that turned the machine into a bot. Another popular medium for social engineering is emails with interesting subjects and content, enticing users to download email attachments. Storm [54] sent spam emails with catchy subjects that contained malicious links to install the bot binary on victim machines.

3.3 Rallying Mechanism

Rallying is the process used by bots to discover their C&C servers. This marks the formal registration of a newly infected bot with the botnet. Some commonly-used rallying methods are described next.

3.3.1 IP address

In this method, the IP address or some means to get the IP address of the C&C server is provided along with the bot binary. IP addresses can be hardcoded or dynamically assigned.

Hardcoded The IP address of a C&C server can either be provided as part of the bot binary (binary hardcoding) or separately (seeding).

In binary hardcoding, the IP address of the C&C server is hardcoded into the bot binary. Botnets were tempted to gravitate towards binary hardcoding because it eliminates the use of DNS from the picture, making their activities stealthier. However, this is a rather primitive method of rallying. An obvious pitfall to this is that reverse engineering the bot binary may reveal the C&C server, potentially leading to C&C server hijack. Another disadvantage of this approach is that a network admininstrator can easily blacklist C&C IPs at a network gateway using an ACL, thereby severing call-back channels of all the bots.

Seeding is primarily used by p2p botnets. At the time of infection, the bot is provided with an initial list of peers. The list reflects a group of active peers in the botnet and is regularly updated. The peer list is separate than the bot binary and can be hidden anywhere on the infected machine with an elusive name. For example, Kelihos/Hlux, a p2p botnet, stored its peer list in the Windows registry under *HKEY_CURRENT_USER/Software/Google* together with other configuration details [57]. Reverse engineering the bot binary does not necessarily reveal the peer list.

Some botnets use methods that combine seeding with binary hardcoding. Nugache [23] provides a good representative of such a botnet. Initial seeding is done either by pre-seeding the victim machine's Windows Registry with a peer-list before actually running the malware, or by obtaining the list from a small set of default hosts hardcoded into the bot binary [22]. The former case falls under seeding while the latter is more typical of binary hardcoding.

Generated Although not observed in real life, the masterminds behind botnets may come up with ways to dynamically generate IP addresses of potential C&C servers. This particularly makes sense if the botnet controls blocks of IP addresses, any one of which can be used as a C&C server at a

given time. However, the IP address generation process is deterministic to a large extent as IP addresses of choice cannot be acquired with the same ease as domain names. For example, consider that the botmaster controls the address *10.1.1.1*. A simple IP address generation scheme would be to provide the bot with the IP address of a C&C server at the time of infection. Subsequently, the bot can calculate the IP address of the 'active' C&C server by incrementing the address every two days. When the full address space is exhausted, it can either start all over again or signal the current C&C server to provide it with a new address range. To evade detection, the address range can be provided through other channels (e.g., a Facebook account). In this case, a broad range of IP addresses is provided by the C&C server but the actual IP address to contact on a given date is generated by some algorithm.

3.3.2 Domain name

In this case, the bot is provided with domain names of potential C&C servers. In some cases, the domain name itself does not belong to the C&C server; rather it acts as a stepping-stone or a link to facilitate communication with the actual C&C server. A botnet may utilize additional services such as DDNS and rogue DNS server to maximize its lifetime and make its C&C structure more resilient.

Hardcoded Like IP addresses, domain names belonging to C&C servers can be hardcoded into the bot binary. This is a better approach than IP address hardcoding from the botnet point-of-view. If the IP address associated with the domain name is taken down, the bot master can still carry out its malicious activities by mapping the domain name to a new IP address while requiring no updation on the bot end.

Generated Botnets can dynamically generate domain names by using algorithm known to the bot and the botmaster. This makes the job of law enforcement agencies difficult. Taking down a domain is a complicated process involving several formalities. By the time the older domain is taken down, the botnet has typically already moved to a new domain. This is called bot-herding.

3.4 C&C

Without C&C communication, a botnet is just an incoherent, random collection of infected machines. In other words, C&C communication forms the

backbone of a botnet. Ideally, C&C communication should utilize a mechanism that involves minimum latency combined with simplicity, availability and stealth. C&C communication can either leverage existing communication protocols or use custom-made protocols for this purpose.

3.4.1 Existing Protocol

For a botnet, there are several incentives to use an existing protocols for carrying out its C&C communication. Existing protocols have been tried and tested and are less likely to have bugs compared to custom protocols. Also, using existing protocols enables C&C communication to mix with regular traffic making detection difficult.

In their infancy, IRC was the C&C medium of choice for botnets. IRC is widely deployed across the Internet and several public IRC networks are in existence. It has simple text based command syntax and provides almost real time communication between bots and C&C server. IRC remained dominant in botnet C&C for some time but is slowly being replaced by other protocols. The use of IRC is not common, particularly in enterprise networks. Also, the message format of the standard implementation of IRC is unique, making IRC traffic easily distinguishable from normal traffic. Agobot, Spybot, and Sdbot are some popular IRC based botnets [58].

After the relative success of law enforcement agencies and industry in tackling the issue of IRC botnets [59], the next step in botnet evolution was HTTP C&C communication. In HTTP-based botnets, bots contact C&C server periodically to fetch commands. Blocking of HTTP traffic is not a viable option for most organizations and corporate networks. Besides, HTTP is the most common protocol used on the internet making it ideal for C&C communication. Use of HTTP as the C&C protocol results in a centralized botnet structure. In the context of larger botnets, some strategy must be adopted to keep the C&C server from being overwhelmed if all the bots happen to contact it simultaneously.

Peer-to-peer (p2p) networks, originally developed to facilitate file sharing among peer nodes, have been utilized for botnet C&C communication. Commands can be dispersed using any node in the p2p network, making detection of C&C servers very difficult. In addition, p2p traffic classification is a daunting task, which makes it hard for gateway security devices to detect and filter p2p traffic. Several protocols are available for p2p-based C&C communication, such as WASTE, BitTorrent, Kademlia, Direct Connect, Gnutella, and Overnet. Slapper [19] and Sinit [20] are the forerunners of the current breed of p2p botnets, followed by Phatbot [21], Storm [22] and Nugache [23].

Researchers have shown how Skype (proprietary protocol) [60] and VOIP (e.g. Session Initiation Protocol) [61] may be used in future for C&C communication.

3.4.2 Neoteric Protocol

In the last few years, botnets have become more creative in the way they conduct C&C communication. The recent Web 2.0 explosion has resulted in a plethora of services focused on user generated content. In particular, social networks have generated enormous web following. Because of their huge size and dynamic nature, it is impractical to monitor or inspect all the user generated content. A botmaster can use any fake profile on social networks such as Facebook or Twitter as its C&C server, where commands are published as 'feed' or 'status'. The longevity of the botnet depends on its ability to generate fake profiles and convey this information to bots. Researchers have analyzed the feasibility and dynamics of social networks as C&C medium [62, 63].

Whitewell [64] used Facebook accounts as stepping-stones in establishing its C&C. The bot agent first accessed a Facebook account to retrieve configuration information including URLs pointing to C&C servers. Actual commands were received from the addresses pointed to by these URLs. Torpig used Twitter search trends in its domain generation algorithm while the actual C&C communication took place over http [65]. Sninfs is another botnet that made use of Twitter to download information-stealing malware on bots [66].

Botnets appearing in the wild that purely rely on social networks for C&C have been shrugged off by security researchers as proof-of-concept efforts. Social networks are deemed infeasible for C&C communication because they present a central point of control which can be taken down easily [66]–in contrast to HTTP-based botnets where taking down C&C server(s)is an arduous task involving third-party cooperation and possibly legal intervention. The task is further facilitated by the cooperative attitude of owners of these social networks who have a common interest in protecting their users.

3.5 Purpose

The main motive of a botmaster in recruiting and managing hundreds and thousands of bots is to use their combined power to carry out malicious activities on its behalf. We call this the purpose of the botnet. In contrast to other malware, motives behind botnet operation are of much graver na-

ture. The botmaster is able to derive large financial gains while remaining clandestine. The odds of being detected are also very low as the botmaster uses machines belonging to unwitting, innocent users for carrying out malicious activities. In addition to financial incentives, botnets are also driven by other goals such as intellectual property stealing, spying and cyberwarfare. We now discuss some of the prime purposes which are being served by contemporary botnets. Note that a botnet, or any subset of bots in it, can serve many different purposes at different times as long as the requisite C&C commands are understood by the bots.

3.5.1 Information Gathering

Information may be gathered for financial gains or reconnaissance purposes. Bots are equipped with a variety of tactics to steal sensitive data and credentials, such as credit card numbers and bank account numbers from infected machines. Botnets are also employed to gather information related to a rival party (a nation or a company), for the purpose of reconnaissance.

Aurora was a specialized botnet that came into limelight following Google China's public disclosure of being victimized by it. It was responsible for stealing intellectual property from several organizations located in different countries [67]. Ghostnet is another example of a botnet used for cyber espionage. It infiltrated high-value political, economic and media locations in several countries, including embassies, foreign ministries and other government offices [3].

Both Aurora and GhostNet are representatives of an emerging class of cyber threats called Advanced Persistent Threats (APT). APT is a category of cybercrime aimed at political and business assests. The attackers make use of all the latest developments in intrusion technologies. Overlooking immediate financial gains, they focus on a specific target. The attack is stealthy and continues over a long stretch of time, and may continue even after the key goals of the attacker have been achieved. The attacker usually has a greater objective than financial gain, is well funded, knowledgeable and well organized [67].

Information gathering can also be financially motivated. During the ten days when Torpig was hijacked, a goldmine of data was recovered including online bank accounts and credit and debit cards [68].

3.5.2 Distributed Computing

Desktop computers typically utilize only 5% of the available computing power [69]. Botnets utilize maximum potential of the bots by using their storage and pro-

cessing power to host and share files, perform distributed password cracking, or any other computational activity of distributed nature [70].

3.5.3 Cyberfraud

Cyberfraud refers to online activities related to deliberate deception for unfair gains. Botnets are used for carrying out cyberfrauds. Web-Phishing entices unsuspecting visitors into performing actions that they would not commit if informed about their consequences. Serving such content on authentic webservers runs the risk of being terminated by hosting authorities. Some botnets use bots to host pages of botmaster's choice by installing a stripped-down version of a webserver on the infected machine. Torpig, an information stealing botnet, used web-phishing to harvest senstive information from infected machines. When a user infected by Torpig visits a target website as specified in the bot configuration file, the original webpage to be displayed is replaced with a fake, but identical, page. The information entered by the user, such as passwords and credit card crdentials is conveyed to a drop-zone for the botmaster's benefit [68].

Botnets can be used to rig the results of online games and polls by ordering bots to act on the botmaster's behalf. Botnets also commit clickfrauds by directing bots to click on pay-per-click advertisements displayed on websites to yield biased click statistics.

3.5.4 Spreading Malware

Some botnets are known to be used for launching other malware. The malware may be part of the botmaster's scheme or the botnet's services may have been rented for this purpose. ZeuS and Pushdo got installed on victim machines by piggybacking on other malware [71, 72].

3.5.5 Cyberwarfare

Cyberwarfare involves measures taken by a state to disrupt or damage another state's assets by penetrating its computers and networks [73]. The last couple of years have seen botnets being increasingly used for cyberwarfare. Cyberspace is regarded as an important area for countries to gain strategic edge over each other. The concept gained momentum when a massive Distributed Denial of Service (DDoS) attack was launched against Estonian websites in 2007, allegedly by Russia [74]. Stuxnet, discovered in July 2010, took nation-sponsored cyberwarfare to the next level. It was a highly targeted and sophisticated piece of malware, which revolutionalized the cyber-

warfare landscape. Stuxnet affected Iran's nuclear sites by manipulating the Programmanble Logic Controller (PLC) used to control uranium enriching centrifuges [4].

3.5.6 Unsolicited Marketing

Online Marketing has proved to be more effective than traditional marketing methods, being instant yet cheap. However, this has been abused by some marketers by subjecting users to unsolicited advertisements in the form of spam emails, pop up ads etc. Several steps have been taken in the last few years to discourage this trend. Blacklisting was used effectively against mail servers responsible for sending spam. However, spammers have found an alternative in the form of botnets. Botmasters supply email templates to bots which send spam emails based on these templates. Spreading the task of spam sending over hundred of bots has made detection very difficult as each bot is responsible for sending only a small number of spam emails.

Rustock, Pushdo, Bagle, Bobax and MegaD are examples of botnets whose names have become synonymous with spam. Researchers tried to estimate spam economy by studying a number of recovered C&C servers. The largest email list was estimated to be worth 10,000-20,000 dollars and profit of the botnet's operators for offering spam services was valued at 1.7-4.2 million dollars [75].

3.5.7 Network Service Disruption

The combined power of the bots, often running into thousands, can be used to bring down legitimate internet services. Botnets were originally created in the context of IRC networks. A popular attack against IRC networks was the clone attack in which all the bots try to connect to an IRC network using clones. Clones are IRC clients controlled by programs/scripts. The resultant traffic overwhelms the IRC network by exhausting its resources. Resultantly, the network is brought down.

Botnets are also used to carry out DDoS attacks. Thousands of bots send requests to the victim service over short period of time, causing the service to crash and consequently become unavailable to legitimate clients. The ability to launch DDoS attacks and make websites and critical services unavailable is also used for cyber-extortion. Large business and enterprises are willing to pay extortion money to botmasters rather than losing sales and credibility. Network service disruption can potentially transform into cyberwarfare if the attack is motivated by a state trying to disrupt another

state's cyber infrastructure, as was evident in the case of cyber-attack against Estonia [74].

In a proof-of-concept experiment, researchers showed how DDoS can be trivially launched against a target by creating a specially crafted Facebook app. Using the app, users unwittingly generated traffic towards a victim [76].

3.6 Evasion

Botnets operate stealthily to evade detection and increase their probability and duration of survival. We can view the evasion strategies adopted by a botnet from the perspective of the bot, botmaster, C&C server and C&C communication.

3.6.1 Evasion tactics at Bots

For bots to remain available to the botmaster for an extended period of time, a number of mechanisms are employed to evade host-based detection. We discuss some of these mechanisms now.

Binary Obfuscation Bot family expansion occurs by exploitation of vulnerabilities on machines that are subsequently infected by the bot-binary. The bot-binary incorporates mechanisms to coordinate with the botmaster to receive commands. To avoid being detected by host-based security applications, several evasion techniques are employed to conceal the bot-binary. Pattern-based detection approaches are defeated by the use of polymorphism. Polymorphism refers to the ability of the bot-binary to exist in several forms. One of the ways to achieve this is by using encryption. The same effect can also be achieved by packing the bot-binary. Packing refers to file condensation. In the context of malware, packing helps obfuscate the malicious code. Some packers are able to produce new binaries every time the original malicious executable is packed.

While code polymorphism succeeds in concealing the bot-binary from pattern-based security applications, it can still be detected by memory-based detection approaches. When executed, the bot-binary needs to be decrypted or unpacked resulting in the same code. This problem is taken care of by code metamorphism. It allows for the bot-binary to be rewritten into different, but semantically equivalent code to defeat memory-based detection approaches

Anti-Analysis Researchers analyze botnet behavior by running bot binary on virtual machines or sandbox such as [77]. Another method for botnet analysis is to use honeypots that emulate known software and network

vulnerabilities to be infected by botnet(s). Honeypots are designed to be self-contained and prevent the spread of botnet beyond the honeypot. To evade such analysis, some bot-binaries perform checks to determine the environment in which they are being executed. If the binary detects a virtual machine or sandbox, it can either refuse to run or it modifies its functionality to evade analysis. After an initial surge of VM-aware botnets, such as Conficker, Rbot, SDbot/Reptile, Mechbot, SpyBot and AgoBot, the trend for such evasion technique is going down due to two reasons [78]. First, legitimate programs rarely perform tests for the execution environment, and thus unnecessarily flags the binary as suspicious. Secondly, virtual machines are now popular among ordinary users who are a legitimate (and ever growing) target, rather than being restricted for security analysis.

Security Suppression After successfully infecting a machine, a botnet may proceed and disable existing security software on the victim machine. If the host is already infected with other competing malware, those are also wiped out. For example, Conficker [79] disables several security related Windows services and registry keys upon installation. It includes a domain name blacklist which it uses to block access to certain security related websites and a process blacklist to terminate processes that may aid in its detection [68].

Rootkit Technology A rootkit is a program that maintains a persistent and undetectable presence on the infected machine by subverting normal operating system behavior. Botnets may install rootkits on compromised machines to gain privileged access to them. This enables them to carry out malicious activities while bypassing typical authentication and authorization mechanisms. As a result, traditional anti-virus software fails to detect intrusion.

3.6.2 Evasion tactics at C&C Servers

Botnet differs from other malware in the ability of the botmaster to remotely control and coordinate all the infected machines via C&C server(s). This essentially means that the 'brain' of the botnet lies in the C&C server. The same can turn into its Achilles' heel. Therefore, botnets invest considerable resources to conceal the C&C server.

IP Flux Botmasters utilize IP Flux to frequently change the IP address associated with the domain name of its C&C server. This helps in evading IP based blacklisting and blocking. This phenomenon is also known as Fast Flux and the resultant structure is called Fast Flux Service Network. IP Flux has been facilitated by Dynamic DNS (DDNS) service. Like DNS, DDNS

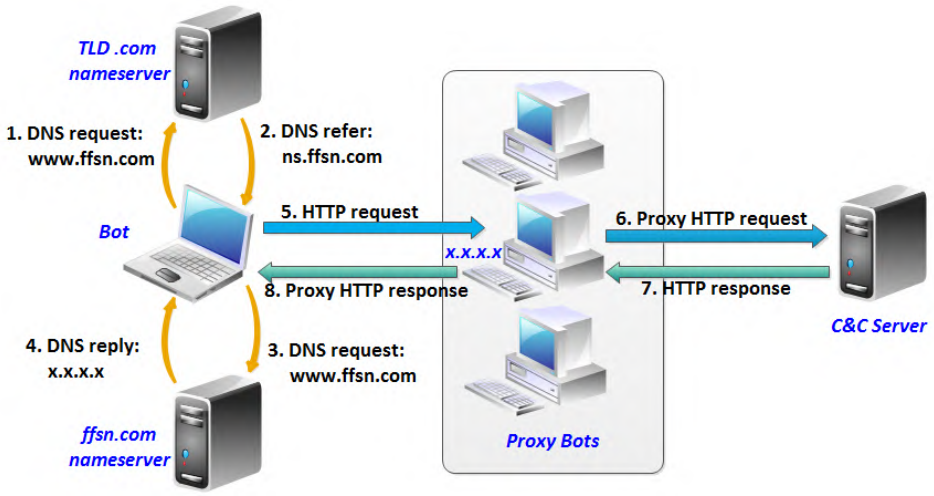


Figure 3.2: Fast Flux Service Network (FFSN) (based on [1])

performs domain name-to-IP mapping. In contrast with DNS which handles only static IP addresses, DDNS can operate with dynamic IP addresses too. Botnets make use of DDNS to keep the (C&C server) domain name to IP address mapping up to date in real-time. Fast flux comes in two flavors; single flux and double flux.

Considering that the bots are aware of the domain name associated with the C&C server, there are two basic steps involved in communicating with it:

i. Resolve the domain name to an IP address. This information will be provided by the nameserver responsible for the requested domain.

ii. Send request to the resolved IP address.

Single flux targets step (2) in the method explained above. Bots do not communicate directly with the C&C server. There is an intermediate layer of machines that act as proxies and relay communication between bots and C&C server. These machines themselves are compromised by the botnet, and hence we call them 'proxy bots'. The resolved IP addresses correspond to the proxy bots. A different subset of the entire pool of proxy bot IP addresses is associated with the domain name after short intervals of time. The botmaster uses its collection of proxy bot IP addresses in a round robin fashion, so the same IP addresses may reappear in the domain's A record [2] after some time. To ensure that the frequent change in domain's record is seen by all bots, the TTL in the name server's reply is set to be very short,

[2]DNS A (Address) Record performs domain name to IP address mapping.

usually a few seconds. Botnets use a domain name for a short period of time before disposing it off and associating a new domain name with the same set of proxy bots.

Double flux takes IP Flux to the next level by extending the concept of flux to the nameserver responsible for resolving the C&C domain name. A request for the C&C domain name will be resolved by a nameserver under the botnet control (step (1)). The IP address corresponding to the nameserver changes frequently. The nameserver's response will include IP address associated with the requested domain name. The resolved IP address also changes frequently and corresponds to the proxy bots, which relay messages back and forth between bots and the C&C server (step (2)).

Fig. 3.2 provides an example of IP flux. The bot knows the C&C server domain name, i.e., *www.ffsn.com*. The bot obtains information about the nameserver for *www.ffsn.com* from the Top-Level Domain (TLD) .com nameserver. Querying the authoritative nameserver *ns.ffsn.com* results in *www.ffsn.com* being mapped to the IP address *x.x.x.x*. The IP address *x.x.x.x* corresponds to one of the proxy bots. The relevant proxy bot then acts as a mediator between the bot and C&C server. In double flux, the nameserver *ns.ffsn.com* is also under botmaster's influence in addition to proxy bots.

Domain Flux Domain flux associates multiple domain names with the same IP address. It helps evade url-based filtering and detection techniques. Domain flux can be achieved by either leveraging on the existing service of DNS to provide domain wildcarding or by using a domain name generation algorithm. Domain wildcarding allows a higher level domain prefixed with any random string to be associated with the same IP address. For example, consider that *nust.edu.pk* maps to *10.1.1.1*. Through domain wildcarding, **.nust.edu.pk* will also be mapped to *10.1.1.1* where * can be any prefix. Alternately, bots can use an algorithm for domain name generation. This algorithm periodically generates a list of domain names. The list is usually large and not all the generated domain names are active at a given time. The bots identify the domain name currently being used as C&C server by trying to contact all these domain names one by one. The domain name that positively resolves indicates successful C&C server identification. The botmaster knows which domain names will be contacted by the bots at a given time. This is possible because the same algorithm is known to the botmaster too. He/she pre-registers the domain names which are expected to be contacted by the bots. Note that the botmaster can associate the same IP address with each domain name registered.

Conficker attracted worldwide attention by affecting millions of computers and reportedly infected key infrastructures and government and military

offices. It uses a domain generation algorithm that generates 50,000 domain names per day, out of which 500 randomly selected domains tried to contact with the C&C server [80].

Rogue DNS Server The attitude towards cyber crime differs greatly among different countries. In particular, local authorities in some countries are indifferent to requests for takedown of malicious servers. Botnets turn this to advantage by having their own DNS service (possibly distributed) hosted in such locations. The addresses of these DNS servers are provided to bots which use them to resolve the address of C&C servers. Rogue DNS servers not only provide a mechanism for carrying out C&C stealthily and effectively, but can also redirect a website's traffic to another bogus website (pharming) or a malicious website focused on stealing sensitive user information (phishing). For example, DNS Changer [81] changed the DNS settings of infected machines to point to rogue DNS servers hosted in Eastern Europe. Legitimate advertisements on web pages were replaced by ads that generated pay-per-click revenue for botnet owners and their clients.

Anonymization Anonymization hides the identity of the sender of a message such that a message cannot be traced back to its sender. Moreover, it cannot be confirmed if two messages were sent by the same sender. While this frustrates network surveillance and traffic analysis attempts, it is a boon for botnets which can carry out C&C communication using low latency anonymization networks, such as Tor [82], to conceal C&C server(s). Bots communicate with the C&C server without knowing its actual location.

3.6.3 Hiding C&C Communication

Detection of C&C communication can have various implications. On one hand, detection of C&C can enumerate bots which can be subsequently disinfected. On the other hand, it can expose C&C servers. The latter can have more serious consequences, for example C&C server can be hijacked crippling the entire botnet. At worst, detection of C&C between botmaster and C&C server can reveal the main culprit, the botmaster. In addition to this, discovery of C&C communication can help defenders understand how the botnet operates and exploit this information to damage the botnet. For these reasons, botnets employ several mechanisms to obfuscate C&C communication.

Encryption Botnets encrypt C&C communication to evade detection. Encryption causes content based analysis to fail, forcing researchers to rely on other traffic characteristics, such as packet arrival times and packet length.

Moving from simple obfuscation techniques to elaborate encryption schemes, botnets have made C&C communication virtually impenetrable. Nugache, touted to be one of the most advanced botnets [83], uses a sophisticated scheme for C&C encryption. A variable bit length RSA key exchange is followed by seeding symmetric Rijndael-256 session keys for each peer connection. Keystroke log files are also encrypted using Rijndael with the help of a key derived from some peer-specific information [23].

Protocol Manipulation Some botnets use protocol tunneling to disguise C&C communication. Typically, firewalls allow HTTP traffic. Leveraging this configuration, botnets have started to use HTTP tunnels for C&C communication. Another emerging trend is the use of IPv6 tunneling for C&C communication [84]. Most modern operating systems support IPv6 by default. However, many intermediate devices do not recognize IPv6 traffic. IPv6 tunneling allows transportation of IPv6 packets over incompatible intermediate devices that only support IPv4. Many firewalls and IDS either do not support IPv6 or are misconfigured, limiting their abilitiy to detect or filter IPv6 traffic. This can be exploited by botnets to carry out C&C communication while bypassing security measures.

Traffic Manipulation A very active C&C communication may tip off security applications about the botnet. Many modern botnets purposely create low volume C&C traffic spread over relatively large periods of time to defeat statistical and volume based detection techniques. For example, Rustock went into hibernation for five days before initiating C&C communication on a newly infected machine [85].

Novel Communication Techniques Botnets use novel communication techniques for C&C which cannot be trivially detected. Social networking websites, in particular Facebook and Twitter, are known to be used by botnets for C&C. Brazen [86], an information-stealing botnet, used Twitter to disseminate links that contained commands or executables to download. Bots subscribed to the malicious Twitter account using RSS to get status updates. Additionally, botnets may utilize any of the existing techniques for information hiding for C&C communication. Some possible candidates are the metadata in image files and least significant bit encoding in image files.

3.6.4 Evasion by the Botmaster

Botmaster is the most protected component of a botnet. Botmaster detection can lead to losing control of the botnet. Moreover, the botmaster can potentially face legal consequences in the form of a prison sentence and/or

hefty fines. Therefore, botmasters utilize elaborate mechanisms to evade detection.

Stepping-Stones Botmasters generally hide their true identity by setting up a number of intermediate hosts, called stepping-stones, between the C&C server and themselves. These stepping-stones can be network redirection services, for example proxies such as HTTP or SOCKS, and SSH servers. The stepping-stones themselves are hosts compromised by the bot master. Botnets prefer to set up stepping-stones in countries with lax cyber crime legislature. This complicates things for law enforcement agencies by necessitating the cooperation of organizations in other countries. Also, the trace back mechanisms are severely crippled because network redirection services operate at the application level and discard off all lower layer information before relaying messages to the next hop.

Botmaster can use an anonymization network as a stepping-stone. This offers the additional benefit of obscuring botmaster's IP address, making it very hard to traceback the botmaster. Defenders can leverage the inherent drawback of anonymity networks, i.e., traffic monitoring cannot be defeated at boundaries of the anonymity network. Detection measures deployed at Internet edges, before a botmaster's traffic enters the anonymity network, can yield promising results. However, it is extremely challenging because a botmaster generates very little traffic and this traffic is not easily distinguishable from legitimate traffic.

3.7 Topology

Another aspect in the taxonomy of botnet features is the topology of botnets based on how bots communicate with the C&C server(s). This topology is essentially an overlay network and is agnostic to the underlying physical topology. Our classification draws inspiration from a report by Damballa [13] on botnet communication topologies.

3.7.1 Centralized

In a centralized topology, all the bots report to and receive commands from a single C&C server. Thus its an easy implementation with minimum C&C overhead. However, it is also a single point-of-failure for the entire botnet. To deal with this problem, techniques such as IP flux and domain flux have been adopted by botnets over time. Botnets using HTTP and IRC as means of C&C communication are typical examples of centralized botnet topology.

Star The basic star topology was adopted by many initial IRC-based botnets. It is a simple model in which the bots directly communicate with the botmaster. An obvious advantage to this is increase in the speed of communication between bots and the botmaster. However, this topology also suffers from the problem of central point of failure–if the central C&C server is taken down, the entire botnet gets disbanded.

Hierarchical Driven by the desire to conceal the botmaster, botnets incorporate one or more layers of proxies between the bots and themselves. The proxies themselves are compromised machines serving the botmaster. This renders a hierarchical quality to the resultant topology, hence the name. There is little probability that analyzing activities of a bot will expose the C&C server. Even if one of the immediate bot proxies is taken down, the botnet will still be functional. The hierarchical nature also allows for portions of the botnet to be separately rented out to third parties. Because of the multiple layers of indirection, communication between the bots and the botmaster is bound to experience some latency.

3.7.2 Decentralized

In decentralized topology, no single entity is responsible for providing command and control to bots. Bot management is either distributed among multiple C&C servers or there is no obvious master-slave relationship between bots and C&C server.

Distributed In a distributed topology, multiple servers control a subset of the bot family. The servers are able to communicate directly with each other. Typically, the servers are spread over different geographical locations. This allows for fast communication by allocating similarly located bots to the nearest C&C server. This also entails benefits of load distribution, availability and resilience. If one server is taken down, its bots can be distributed among the remaining servers. Besides, the issue of central point of failure has also been tackled as it is unlikely that all the countries hosting the C&C servers will simultaneously and positively entertain legal take down requests. The resulting trade-off, for the botmaster, is increased complexity of the botnet implementation.

Random In a random topology, there is no clear master-slave relationship. Any bot can be used to issue commands to other bots in the botnet.

In peer-to-peer (p2p) networks, communication between bots and the botmaster forms unpredictable routes. The botmaster can use any peer node to float commands which will be broadcast to all the bots. The absence

of centralized C&C makes it extremely difficult to locate the botmaster or hijack the botnet. In hierarchical topology, shutdown of a proxy server may result in a portion of the botnet becoming dysfunctional. In contrast, in p2p topology, 'cleaning' of a single bot will have no effect as alternate routes are always available. A disadvantage to the p2p based random toplogy is that the C&C communication will experience unpredictable delays making it unsuitable for carrying out coordinated, large-scale attacks. Furthermore, capture of a single bot reveals several other bots because each bot maintains a peer-list.

Cooke et al. [24] proposed a topology that has not been observed in real world botnets but gives insight into possible future botnet trends. In this topology, each bot knows about only one other bot. The botmaster can use any bot to issue a command which is encrypted and passed on to the next bot discovered through random scanning of the internet. This demonstrates an extremely resilient model with high survivability.

3.7.3 Hybrid

An interesting topology for botnets would be a combination of centralized and decentralized topologies, for example, a botnet using centralized structure between C&C server and the front-end proxy bots but p2p as C&C for the bots under control of individual proxy bots. At the time of writing of this paper, we are not aware of any existing botnet that utilizes a hybrid topology. Further research is needed to investigate the pros and cons of such a structure.

3.8 Summary

As a first step to understand the botnet phenomenon, it is important to systematically explain botnet behavior. In this chapter, we present a taxonomy of botnet behavior. Our high-level categorization of botnet behavior comprises of *Propagation*,*Rallying*,*C&C*, *Purpose*, *Evasion* and *Topology*.

Propagation mechanisms refer to the methods used by botnets to infect other machines. Based on the degree of (human) user intervention required, we classify propagation mechanisms as *Active* and *Passive*. In *Active* mode of propagation, the botnet is capable of locating and infecting other machines without requiring assistance from a (human) user. *Scanning* is the only representative of this type of propagation. In *Passive* propagation, infection cannot spread to other machines without user assistance. Some examples of passive propagation are malware distribution through *drive-by download*, *infected media* and *social engineering*.

Rallying is the process used by bots to discover their C&C server(s). Bots can locate C&C server(s) by *IP address* or *domain name*. Both IP address and domain name of the C&C server can either be *hardcoded* into the bot binary or *generated* using some algorithm.

Once C&C server has been located, *command-and-control (C&C) communication* is used to issue instructions to the bots. These instructions could relate to updation / modification of botnet malware, spreading the infection to other machines or other malicious activities. C&C communication can take place either through *existing protocols* or *custom/neoteric protocols*.

Purpose refers to the main motive of the botmaster in recruiting the bots. The most notable motives include *information gathering*, *distributed computing*, *cyberfraud*, *spreading malware*, *cyber warfare*, *unsolicited marketing* and *network service disruption*.

A primary consideration in the operation of a botnet is stealth or *evasion*. Botnets employ a number of mechanisms to evade detection and thus increase their probability and duration of survival. These mechanims can be broadly categorized on the basis of the component of botnet infrastructure they are trying to obfuscate, i.e., *bots*, *C&C server(s)*, *C&C communication* or *botmaster*. *Evasion tactics at bots* strive to defeat host-based detection. These involve *binary obfuscation* of the bot binary and *rootkit technology*. Additionally, botnets may also include *security suppression* and *anti-analysis* techniques to evade detection by security software and research tools (such as virtual machines / honey pots) respectively. Being the control center of the botnet infrastructure, botnets set up a number of *evasion tactics at C&C servers* to enable them to function uninterrupted for the maximum possible duration. These tactics involve frequently changing the IP address (*IP flux*) or the domain name (*domain flux*) associated with the C&C server, *anonymization* of the C&C server and/or the use of DNS servers under the control of botmaster (*rogue DNS server*). Another important aspect of evasion is *hiding C&C communication*. This can be achieved through *encryption*, *protocol manipulation*, *traffic manipulation* and/or use of *creative/novel communication techniques*. Finally, *evasion tactics by the botmaster* typically entail the use of network redirection services or anonymization (*stepping-stones*).

Another important aspect in the taxonomy of botnet behavior is the *topology* of botnets based on how bots communicate with the C&C server(s). The high-level classification of botnet topology comprises *centralized*, *decentralized* and *hybrid* topologies. In a *centralized* topology, all the bots report to and receive commands from a single C&C server. *Star* and *hierarchical* toplogies are derivatives of the centralized botnet topology. In *decentralized* topology, no single entity is responsible for providing command and control

to bots. *Distributed* and *random* topologies derive from the decentralized botnet topology. Finally, the *hybrid* topology represents a combination of centralized and decentralized topologies.

Chapter 4

Taxonomy of Botnet Detection Mechanisms

'Mr. Holmes, they were the footprints of a gigantic hound!'

— Sir Arthur Conan Doyle in "The Hound of the Baskervilles"

This chapter presents the second taxonomy in a series of three taxonomies pertaining to different aspects of botnets: behavior, detection mechanisms and defense strategies. Our second taxonomy classifies botnet detection mechanisms.

Detection of botnet infection is an essential component of an effective security policy. The goals of a comprehensive security plan include minimization of threats, fixing the vulnerabilities and prevention of intrusion. If intrusion has been successful, the next important step is to be able to detect it. In our study, intrusion is synonymous to botnet infection. Detection of botnet infection helps in recovery from the compromise and taking steps to fix the vulnerabilities that helped the infection to succeed in the first place. We provide an extensive overview of different botnet detection mechanisms, based on which component is being targeted; bot, C&C or botmaster.

4.1 Overview

Several mechanisms for botnet detection have emerged over time. To the best of our knowledge, none of these techniques unveil all the botnet components at once. Existing detection approaches try to identify a part of the big jigsaw puzzle; i.e., the botnet. One part can lead to the other, and different parts can be placed together to reveal a greater portion of the puzzle, however, there is no panacea.

Based on which component is being targeted, we classify botnet detection into different facets; bot detection, C&C detection and botmaster detection. We have intentionally excluded C&C server detection from the aforementioned facets because detection of C&C communication typically reveals C&C servers too.

Our definition of botnet, a distributed malware with bots, C&C server(s), a botmaster, and the C&C communication between these components, is in harmony with the interpretation in existing literature. However, most previous literature refers to detection of single, bot-infected machines as bot detection, and detection of bot families as botnet detection. In view of the typical interpretation of the term 'botnet', this definition of botnet detection is paradoxical. We posit that what is commonly understood as botnet detection is still bot detection, with particular regard to bot families. Gu et al. [40] proposed a detection approach, BotMiner, that could detect bot families or sub-families within the monitored network. To elucidate the scope of their detection framework, they described a botnet as a coordinated group of malware infected machines with similar communication and activity patterns. This means that if the botmaster uses the bot family to carry out n non-overlapping malicious activities, BotMiner would detect n instances of the same bot family, hence the term bot sub-family detection. Another work [87] used the term bot family to refer to botnet infected machines with similar communication patterns. To provide a generalized frame of reference, we define the following terms:

Definition 4.1 *1. Botnet Detection: Detection of all components of a botnet, comprising the botmaster, C&C server(s), means of C&C, and (all or a subset of) bots.*

Definition 4.2 *2. Bot Detection: Detection of botnet infected machines, with or without regard to bot families.*

Definition 4.3 *3. Bot Family Detection: A class of* Bot Detection *focused on bot family detection.*

In this taxonomy, we broadly categorize botnet detection mechanisms as *Bot Detection*, *C&C detection* and *Botmaster Detection*. These facets of botnet detection can be used in combination. For example, *C&C Detection* can be followed by *Bot Detection* and vice versa. Botnet detection approaches can be broadly categorized as *Active* and *Passive* (We later discuss other alternatives in Section 4.5). In active detection, the strategy is to take part in the botnet operation by impersonating as a component of the botnet instead of passively observing its activities. For example, active C&C detection

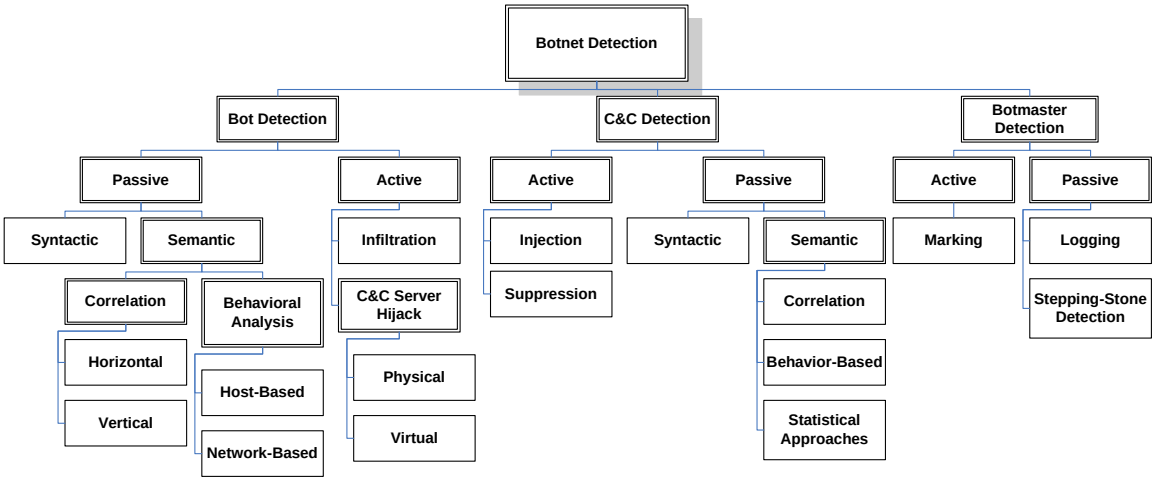


Figure 4.1: Taxonomy of botnet detection mechanisms.

involves online manipulation of network flows to deduce information about possible C&C communication. In contrast, passive detection approaches detect botnets by silently observing and analyzing botnet activities without making a conscious effort to participate in the proceedings.

Our high-level classification of botnet detection mechanisms into 'active' and 'passive' approaches is of practical importance for any researcher or network administrator. Active detection techniques raise several legal and ethical questions, and whether it can/cannot be performed will vary from organization to organization. For this reason, network policies typically prohibit active manipulation of network flows. Another issue that necessitates use of passive detection techniques is users' privacy concerns. Moreover, active detection methods can jeopardize the security of the very hosts they seek to protect. If active detection is discovered by the botmaster, there is a chance that he/she will direct attack traffic to the responsible host(s) as counter defense.

The rest of this section discusses further classification of these high level botnet detection facets; i.e., *Bot Detection*, *C&C detection* and *Botmaster Detection*.

4.2 Bot Detection

Bot detection can be performed with or without regard to bot families. Users and network administrators are usually indifferent to information about bot families. Their primary concern is to protect their systems and networks from infections, regardless of details about bot family. On the other hand,

security researchers are particularly interested in identifying bot families. The degree of prevalence of different botnets, their geographical distribution, and the common characteristics of botnets are some of the plausible reasons for their heightened interest. Detection of bots indicates vulnerability of a host or network to botnet infection. This can be followed by remedial strategies aimed at recovering from the infection and preventive measures to avoid getting infected in future.

4.2.1 Active Detection

Active bot detection involves participating in the botnet operation. This typically involves impersonating as a component of the botnet. Active detection approaches involve *Infiltration* and *C&C Server Hijack*.

Infiltration In infiltration, a defender-controlled machine masquerades as an actual bot and probes the C&C server or other peers in case of a p2p-based botnet to gain details about other bots. Nappa et al. [60] proposed a replay attack on a Skype-based botnet. The technique can be effective for other p2p botnets too. The defender progressively gains information about other bots by repeatedly issuing crafted, bogus messages to declare itself as a new bot and subsequently obtain new peer-list.

C&C Server Hijack Bots can be actively detected by C&C server hijack. Bots report to and receive commands from C&C server. Taking control of the C&C server will reveal all the bots that contact it. This can be achieved by exploiting botnet rallying mechanism. A defender can use this information to his/her advantage to hijack the server. This approach also leverages knowledge of botnet topology. Centralized botnet structures are more amenable to C&C server hijack. In decentralized botnets, the C&C server can be any peer and will, at most, reveal information about bots in its peer list. To gain futher information, some other techniques need to be employed, such as active crawling of the p2p botnet. The seizure of C&C servers can be *Physical* or *Virtual*.

In a *Physical* hijack, law enforcement agencies physically seize the C&C servers. However, it is possible to take over the C&C servers without involving legal authorities by mutual cooperation. This is possible if the C&C servers in question are not in geographically diverse locations. With the help of service providers, researchers gained access to several C&C servers used by Pushdo/Cutwail botnet [75]. In addition to other interesting information, 24 databases containing details about the bots and spam operations were disclosed.

In a *Virtual* take over, defenders hijack the C&C servers by redirecting C&C communication to a machine under their control. This technique has been used by researchers to hijack botnets that use domain names for rallying. By virtue of DNS sinkholing, traffic sent by bots to known botnet domains can be forwarded to defender-controlled machine. The domain names of C&C servers can be learnt by analyzing the botnet behavior on infected machines. Researchers [88, 89, 68] have recovered future rendezvous points by reverse engineering the domain generation algorithm used by botnets utilizing domain flux.

4.2.2 Passive Detection

Passive detection approaches detect botnets by silently observing and analyzing their activities without making a conscious effort to participate in the proceedings. Passive techniques can be *Syntactic* or *Semantic*.

Syntactic Syntactic or signature-based approaches identify botnets by comparison with pre-determined patterns of botnet infection obtained from observed samples. Rishi [38] is a purely syntactic approach for detection of IRC based bots. It formulated regular expressions as signatures to identify bot-like IRC nicknames. Snort [90] has a rich signature database and is used by BotHunter [47] along with other anomaly detection components to feed alerts to a correlation module.

Signature-based detection degrades if strong evasion mechanisms such as encryption and bot binary obfuscation are in place. New threats for which signatures have not been developed go completely undetected. Hence, this method should be complemented with behavior-based detection approaches.

Semantic Semantic detection methods use the context of events and protocol information to announce detection of malicious behavior. The process of bot detection entails careful analysis and is based on deviation from established benign behavior or similarity with bot behavior. Semantic detection techniques can be further classified as those based on *Correlation* and *Behavioral Analysis*.

Correlation techniques are used to identify bots as well as bot families. Botnet is a coordinated attack infrastructure. This idea has been used to cluster hosts that perform similar activities or communication. Correlation-based techniques have been shown to successfully detect bots utilizing centralized as well as decentralized topology.

Correlation can be performed on primary data as well as secondary data. Primary data encompasses all the activities vital to the sustenance of the botnet like egg download, propagation activities, and C&C communication.

Attacks and malicious activities fall under secondary data. Secondary data analysis is an effect-to-cause approach and relates malicious behavior to bots. It cannot be used as a generic method to detect all kinds of botnets because it is restricted to the attack characteristics that the analyst is trying to target. Researchers have tried to detect and study different bots by correlating similar spam emails collected from records of popular email service providers [49]. Ramachandran et al. [50] detected botnets by monitoring and correlating DNSBL queries which botnets perform as a way of reconnaissance before launching a spam campaign. Researchers have also detected botnets by analyzing and correlating anomalous DDNS [51] and DNS traffic [52].

Complex Event Processing (CEP) is an area that closely relates to correlation. It correlates events in real-time to detect a target complex event comprising of multiple simple or complex events. The concept has recently been applied to the area of information security [91, 92, 93]. Given its complex nature, botnet detection seems to be a suitable candidate to be mapped as a CEP problem. However, we are not aware of any research effort that attempts this mapping.

Correlation can be further divided into two main branches; *Vertical* and *Horizontal* correlation.

Horizontal correlation detects bots by observing similarities in host behavior and/or communication. BotMiner [40] clusters similar communication and malicious activity patterns and then performs cross-cluster correlation to list bots in the monitored network. BotSniffer [41] uses spatial-temporal similarities between bot families to detect bots. Several methods have been proposed that detects botnets by clustering flows with similar characteristics [42, 44, 43, 45]. Horizontal correlation techniques suffer from a setback; there must be more than one bot infected by the same botnet in the monitored network for successful detection.

Vertical correlation correlates activities of a single machine and compares it with a model of bot behavior. BotHunter [47] analyzes the sequence of communication exchanges between a host and the internet. It models the infection as a loosely coupled sequence of five stages: inbound scanning, exploit usage, egg downloading, outbound bot coordination dialog, and outbound attack propagation. Suspicious outbound activity coupled with intrusion detection activity indicates a successful bot infection. BotTracer [48] uses virtual machine techniques to detect botnets based on three phases of botnet life cycle; automatic startup of bot with no user intervention, C&C establishment and attack.

Behavioral Analysis is another class of semantic detection techniques that analyzes botnets by observing deviations of machine/traffic behavior from an established normal pattern or its similarity with known botnet behavior.

Behavioral Analysis can be *Host-based* or *Network-based*.

Host-based detection methods look for signs of bot-like behavior on a host. BotSwat [94] detects botnet infection by identifying command-response behavior. It tracks programs that use data received from unreliable network sources (tainted data) and looks for possibility of remote bot initiation. It has related a set of system calls with bot activity, which are called 'gate functions'. Botnet infection is suspected if tainted data is passed as argument(s) to gate functions.

Network-based methods use information derived from network traffic and services to detect bots. It typically entails classification of traffic into network applications and looking for bot-like behavior in individual application traffic. Classification of traffic into network applications is not trivial. Applications using dynamic and random ports have rendered port-based application classification futile. Developing payload based application signatures is defeated by the use of encrypted traffic and privacy issues. Lu et al. [95] used payload signature examination method to classify traffic and found that 40% network flows could not be classified into specific applications. A number of methods [96, 97, 98, 38] focus on detection of IRC-based botnets and hence require some means to divide traffic into IRC and non-IRC parts. P2p bots can be separated from p2p file sharing applications on the basis of flow characteristics, p2p churn, and difference between machine and human behavior [45]. Researchers have detected bots comprising FFSN's by observing DNS resource records [99, 100]. Similarly, bots can be detected by monitoring requests to rogue DNS servers known to support botnet activities. A vein of research focuses on identifying C&C communication in network traffic which can be used to reveal bots. More discussion can be found under the facet about C&C Detection.

4.3 C&C Detection

Detection of C&C channel is an important aspect of botnet detection. Identification of C&C and its subsequent analysis can help in understanding botnet behavior. This information can be leveraged to identify bots and possibly C&C servers.

4.3.1 Active Detection

Active C&C detection involves taking part in the botnet operation, for example, online manipulation of network flows to deduce information about possible C&C communication. Active C&C detection involves *Injection* or

Suppression.

Injection Injection entails injecting packets into suspicious network flows. The similarity of the reply to the injected packets with typical bot response indicates that the flow might be part of C&C communication. Injection can be performed either by infering the botnet C&C protocol, or by blindly replaying incoming packets in the suspicious flow with or without minor changes. The latter route can be taken for botnets using stateless C&C protocol. It will fail for botnets that are secured against replay attacks of this form by using timestamps or sequence numbers. C&C detection based on protocol inference requires reverse engineering the botnet C&C protocol. This allows initiation of an informed C&C dialog for forensic purposes. C&C detection based on protocol inference can be automated by feeding the detector with information about protocols of known botnets. Using this knowledge, packets can be injected into suspicious flows to compare the response with known botnet response [101]. BotProbe is a tool based on active injection techniques to identify chat-like botnet C&C communication [53] . BotProbe leverages two basic differences between human and bot responses. Unlike humans, bots respond deterministically to the same command and are intolerant to typographic mistakes.

Suppression In suppression, incoming/outgoing packets in suspicious network flows are suppressed to elicit known response from any of the ends of the C&C communication. For example, consider a suspected bot that requests C&C server for some update and the corresponding response is dropped. After some retries, the bot will activate its back-up mechanism. If the events that are fired up as part of the back-up mechanism are already known, botnet infection can be confirmed .

We are not aware of any existing detection mechanism that uses suppression and introduce it as a new venue for research.

4.3.2 Passive Detection

Active C&C detection is complemented by passive techniques. Active C&C detection receives more scrutiny and criticism because of policy restraints and greater penalty in case of false positives. This explains the prevalence of passive C&C detection methods. It involves silently observing network traffic, looking for cues of C&C communication. These mechanisms can be broadly categorized as *Syntactic* and *Semantic*.

Syntactic Syntactic C&C detection works by developing signature-based models of C&C traffic. The signatures are obtained by observing frequently

occurring strings or token sequences in malicious traffic. Manual signature development is less reliable and time-consuming. Methods for automated model generation have been proposed recently [102, 87].

Semantic Semantic C&C detection approaches use some heuristic to associate certain behavior with C&C traffic. These can be further divided into *Statistical*, *Correlation* and *Behavior-based* approaches.

Statistical Approaches can be used to detect botnet C&C communication. Machine learning has been extensively used for network traffic classification [103]. However, its efficacy in detecting C&C has not been explored much. Machine learning, particularly supervised learning, has been used for C&C detection. It involves identification of features, such as range of packet lengths, inter-packet arrival times and flow duration. Using these features, a classifier is trained on relevant datasets. Subsequently, it develops rules which are fed to the Machine Learning algorithm for classifying network traffic as benign and C&C. Machine learning algorithms have been used to first classify network traffic into IRC and non-IRC traffic and then identified botnet and non-botnet traffic within the IRC traffic [104]. In another approach, graph-based models are used to represent malicious C&C connections [105]. The graph model for each network connection is based on the system calls that lead to this connection and the system calls that operate on data that is returned. Machine learning techniques are then used to automatically generate graph templates for C&C activity by training the classifier on a set of graphs that are associated with known C&C.

The main idea behind *Correlation* based methods is that similar communication patterns in network traffic can point to C&C traffic. Strayer et al. [43] used a method to first identify botnet-like traffic and then find C&C traffic by clustering flows with similar characteristics such as bandwidth, packet timing, and burst duration. Gu et al. [40] combined this approach with activity-based correlation of hosts and performed cross-cluster correlation to detect botnets.

In *Behavior-based* detection methods, C&C traffic is identified by observing its deviation from normal traffic or its similarity with established behavioral model of C&C traffic. Divergence of a flow from typical network usage for a user can qualify it as possible C&C flow. For example, a connection made to Russia at midnight from a user's machine who does not use the network after evening is cause for concern. Regardless of which C&C protocol is being followed, C&C communication has some behavioral characteristics that can give it away. Wurzinger et al. [87] presented a system to automatically generate C&C models from botnet samples run in a controlled environment. These models were generated by attributing response behavior,

such as sending spam emails and carrying out DoS attacks, to previously-issued commands. The command portion of the model was signature-based. However, response was detected by observing anomalies in behavior. These command-response models were then used to detect C&C and subsequently bots.

4.4 Botmaster Detection

Not many botnet detection techniques target the botmaster. Botmaster detection can have serious ramifications for a botnet. On the one hand, it can lead to legal prosecution and hefty fines for the botmaster. On the other hand, it can lead to disbandment of the entire botnet based on information provided by the botmaster. For the reasons cited above, botmaster is the most protected part of a botnet and hence difficult to detect. The botmaster issues only a few commands to the C&C server to be relayed to the bots, thus generating little traffic that might also be encrypted. Detection of the botmaster is further complicated by the presence of stepping-stones between the botmaster and the C&C server.

4.4.1 Active Detection

Active botmaster detection involves manipulation of botnet activity. There are very few mechanisms that actively detect the botmaster. Active botmaster detection techniqes revolve around *Marking*.

Marking techniques have been used extensively to traceback culprits responsible for malicious activities over the Internet. There are various flavors of packet marking such as probabilistic packet marking [106, 107, 108], ICMP traceback [108, 109] and deterministic packet marking [110, 111]. In marking schemes, some information is written into packets by either the victim machine or intermediate routers to help locate the attacker. Ramsbrock et al. [112] presented a mechanism for live traceback of botmaster by injecting watermark in response packets to the botmaster from a rogue bot under the defender's control.

4.4.2 Passive Detection

Passive detection of botmaster involves analysis of network traffic and other data without manipulating or modifying botnet operation. Passive mechanisms for botmaster detection involve *Logging* and *Stepping-stone Detection*.

In logging mechanisms, routers log information about packets passing through them. This information is used to verify whether or not suspected packets were forwarded by specific routers. Logging mechanisms incur heavy computational complexity in addition to scalability issues. Source Path Isolation Engine (SPIE) is a hash-based IP traceback mechanism [113]. SPIE used deterministic logging mechanism to reconstruct path to the attacker. Logging mechanisms have not been used for botmaster detection so far, mainly because routers on a packet's path cannot be dictated to support and maintain additional logs.

Another way to detect botmaster is through stepping-stone detection. The botmaster hides its identity behind one or more stepping-stones. While stepping-stone detection does not directly detect the botmaster, in certain cases it can be used recursively to identify the botmaster. There are two main difficulties in the detection of stepping-stone connections. Packets from the botmaster may arrive to the C&C server with random delays between them. The delay can be caused by network factors or the botmaster can intentionally introduce them to evade detection. The botmaster may also add additional packets, called chaff, to further confuse the detection process.

All existing stepping-stone detection methods work on the basis of correlation between connection content, host activity or packet timing. The methods based on host activity correlate user login information from different hosts part of the stepping-stone chain. As the stepping-stones are under the botmaster's control, detection can be easily evaded by manipulating and forging information on these hosts. Content-based detection methods such as *thumbprinting* [114] detect connections belonging to the same chain by observing similarity in their contents. Because these methods are based on content inspection, they are effective for unencrypted traffic only. The predominance of encryption for obscuring C&C traffic has rendered content-based detection futile.

Another branch of approaches for stepping-stone detection leverages information regarding packet arrival time. Timing and chaff perturbation are great challenges for the effectiveness of these methods. Both timing and chaff perturbation are traffic mainpulation techniques. Timing perturbation refers to random delays between packets while chaff refers to insertion of meaningless packets to frustrate analysis. Characteristics of interactive traffic such as packet size and timing have been used to detect encrypted stepping-stone connection [115]. Wang et al. [116] correlated inter-packet timing characteristics of both encrypted and unencrypted connections to detect stepping-stone connections. Donoho et al. [117] used wavelets and multi-scale methods to separate short-term behavior of stepping-stone connections (delay and chaff perturbations) from their long-term behavior (which can be

correlated). Blum [118] used techniques from computational learning theory and the analysis of random walks to detect and identify encrypted stepping-stone connections with polynomial upper bounds on the number of packets required for the analysis. Zhang et al. [119] proposed techniques to detect encrypted stepping-stone connections. Their method was agnostic to delay and chaff perturbations. Other timing based methods include [120, 121, 122].

The previously discussed techniques passively detect stepping-stones, while very few techniques actively detect stepping-stones [123, 120]. Wang et al. [123] coined the sleepy watermarking method to actively detect unencrypted stepping-stone connections. Sleepy Watermark Tracing (SWT) activates or 'wakes up' when an intrusion is detected. In such an event, it injects a watermark into backward connection of the intrusion and collaborates with intermediate routers to reveal all the hosts in the stepping-stone chain.

4.5 Discussion on the Taxonomy

In the preceding discussion, we broadly classified botnet detection mechanisms as 'active' and 'passive'. This classification was along the dimension *Level of Activity*. In this section, we identify a number of other dimensions that could be used to classify botnet detection mechanisms. We highlight classification based on the remaining dimensions as new venues for research. Furthermore, we investigate the effect of different botnet features on the accuracy of botnet detection approaches.

4.5.1 Dimensions of Botnet Detection

Depending on the interest of the reader, each facet of botnet detection can be further explored in the light of any of the applicable botnet detection dimensions as described in Fig. 4.2. For example, a security researcher interested in estimating footprint of different botnets would want to explore detection techniques with the dimension *Discernment*. On the other hand, researchers developing a botnet detection tool to be deployed at a large ISP would be interested in the dimension *Analysis Depth*. We call these different classification criteria 'dimensions'.

Some dimensions may not be applicable to certain facets. For example, *Analysis Direction* and *Discernment* are more relevant to *Bot Detection*. *Specificity* is applicable to *Bot Detection* and *C&C Detection*. The remaining dimensions can be used with any of the three facets of botnet detection; i.e., *Bot Detection*, *C&C Detection* and *Botmaster Detection*.

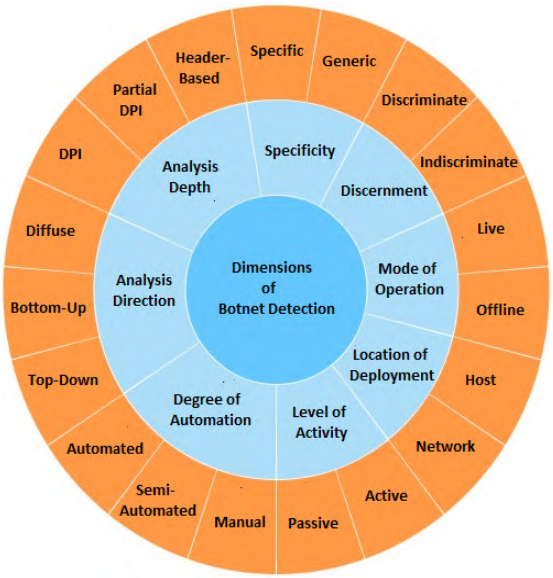


Figure 4.2: Dimensions of botnet detection.

Degree of Automation Depending on the degree of human participation in the detection process, we can classify botnet detection methods as manual, semi-automated and automated.

Manual approaches that require significant human effort to detect botnets fall under this category. Typically, such techniques require manual acquisition and reverse engineering of bot binary for developing signatures that are fed into custom-made botnet detection software. Considerable human effort is required to reflect even the slightest change in botnet functionality. Many methods [124, 22] for p2p-based botnet detection are manual.

Semi-Automated methods for botnet detection require very little human intervention and most of the detection is performed in automated fashion. Typically, human involvement is required only when something changes in an existing botnet or new kinds of botnet have to be catered for in the detection engine. For example, [38, 47] heavily rely on manually developed signatures for botnet detection. If the botnet C&C style changes, or a new botnet appears, corresponding signatures have to be manually developed and incorporated in the detection framework. This is in contrast with recent work that automatically infers botnet C&C protocol [101] which may be used for signature development.

An *Automated* system should require no human intervention after initial development. Ideally, any detection method should be as generic and automated as possible. Methods that rely on behavioral characteristics of botnets such as [43] or automatic generation of detection models [87] have

the potential to operate in a fully automated manner.

Analysis Direction Botnet detection can be carried out in several directions, where direction means the sequence of analysis. Some detection methods start at the bots and perform correlation at an upstream network component, others take the opposite route and observe anomalies at upstream network components and trace the effect back to the cause.

Top-Down approaches observe phenomena at upstream network components such as spam [49, 46, 125] and anomalies in DDNS [51] and DNS traffic [52]. Clustering and correlation techniques are then applied to identify bots belonging to specific botnets.

In *Bottom-Up* approaches, behavior exhibited by individual bots is analyzed at an upstream network component to identify bot families based on similarity of behavior. BotMiner [40] detects bot families by clustering machines that display similar communication and activity patterns.

Diffuse methods of detection are more relevant to p2p botnets where there is no hierarchy in the botnet components. The detection process can start at any bot and other bots can be discovered by analyzing communication patterns or peer-lists.

Analysis Depth Some detection methods base their analysis on compact and easily accessible data, thus involving minimum overhead. Other techniques perform more in-depth analysis of data. This factor plays a significant role in the effectiveness of network-based, real-time detection algorithms.

Deep Packet Inspection (DPI) based detection methods that perform fine-grained analysis of data. Deep Packet Inspection (DPI) has been used for signature-based detection of malicious payloads [47] or communication [41, 38] and to compute content-similarity for packet payloads [45]. DPI-based methods involve significant computational and operational complexity, particularly if large amount of network traffic is involved.

Partial DPI based methods perform DPI only for suspicious data instead of analyzing all data indiscriminately. Zhang et al. [126] proposed a sampling-based approach that identified bot flows which were then forwarded to fine-grained botnet detectors.

Header-Based methods operate on header/flow level data and incur minimum complexity, making them ideal for being deployed over large networks. Karasaridis et al. [97] compared flow records of suspicious hosts with IRC traffic models to identify botnet controller activity.

Specificity Some detection methods are tailor-made to detect certain kinds of botnets. Other methods perform botnet detection on the basis of general

botnet characteristics which do not vary among different instances. Specificity is the measure of a detection method's dependence on instance-specific features.

Specific detection methods are custom-made to detect certain kind of botnets. For example, Rishi [38] can only detect IRC-based botnets. These approaches fail to detect botnets using a different structure than the one targeted.

Generic approaches consider general characteristics of botnets instead of targeting specific botnet instances [43, 40]. Resultantly, the results are more promising and the scope of detection is broader.

Discernment Discernment is the ability of a detection method to differentiate between different bot families. While a home user is less likely to be concerned about details of the other members of the bot family that infected his/her machine, this information is valuable to researchers who try to estimate the footprint of different botnets.

Discriminate methods for botnet detection not only identify the infected machine, but can also provide information about bot family. This typically requires an elevated view of the network. Bot families are determined by clustering hosts with similar communication or behavioral characteristics [40]. Host-based detection techniques can also identify bot family which the bot in question serves. Signature-based detection can possibly name the bot family to which the bot belongs. Use of system calls and library routines associated with known botnets can also reveal which bot family the bot is associated with.

Indiscriminate methods can detect botnet-infected machines but cannot make distinction between different bot families. A good example of this is BotHunter [47] which can discover infected machines in the monitored network but gives no information about bot families.

Mode of Operation Mode of operation refers to the suitability of the detection method to operate in a live or offline environment.

Live methods can detect botnets in real-time while the monitored host or network carries out its normal operations. For example, Ramsbrock et al. [112] use watermarking for live detection of botmaster. The ability of a detection method to operate in live mode is closely related to how much it relies on DPI. A network based detection method performing full DPI is bound to falter in live environment with potentially huge and unpredictable traffic.

Offline methods for botnet detection provide promising results when ran on log files or network traffic dumps. These methods are particularly helpful for performing network forensics or research purposes. BotHunter [47] can

operate in both live and offline setting. In offline mode, it can operate on pcap file as well as Snort [90] logs. Some security companies [15, 16] offer services to investigate botnet-related incidents by analyzing logs and other evidence.

Location of Deployment Based on location of deployment, botnet detection methods can be classified as host-based and network-based. These are not mutually exclusive and a distributed method can make use of both of them.

Host-Based methods that analyze host behavior or data to detect potential botnet infection. Obviously, these methods can only declare botnet infection on individual machines and give no information about other bots belonging to the same bot family. Both BotSwat [94] and BotTracer [48] are examples of botnet detection systems deployable on hosts only.

Network-Based methods analyze network traffic and depending on the scope of analysis, can be deployed anywhere in the network hierarchy. Possible deployment location can be proxy server, ISP and so forth. Karasaridis et al. [97] proposed a method to detect botnets on a large Tier-1 ISP network.

Degree of Activity By degree of activity, we mean the extent to which the detection method interferes in the botnet operation. Some detection methods quietly observe ongoing botnet activity and base their decision on this information alone. Other detection techniques involve active participation in botnet C&C or infiltration.

In *Active* detection, the strategy is to take part in the botnet operation by impersonating as a component of the botnet instead of passively observing its activities. Active C&C detection involves online manipulation of network flows to deduce information about possible C&C communication.

Passive detection approaches detect botnets by silently observing and analyzing botnet activities without making a conscious effort to participate in the proceedings.

4.5.2 Effect of Botnet Features on Botnet Detection

Different botnet features positively or negatively affect the accuracy of botnet detection mechanisms. This is very important information from the point of view of a security researcher. Armed with knowledge of pros and cons of individual detection mechanisms, complementary approaches can be paired to achieve a synergistic effect. Additionally, informed decisions can be made in the choice of detection mechanisms for designing tailored strategies targeted at detection of specific threats. To this end, we present three tables,Table 4.1, 4.2 and 4.3, that investigate the effect of botnet features on

Table 4.1: The effect of botnet behavioral features on bot detection mechanisms.

Bot Detection Mechanism	**Related Botnet Features**	
	Improve	**Degrade**
Infiltration		Encryption, Binary Obfuscation.
C&C Server Hijack	Centralized Topology, Rogue DNS Server.	Decentralized Topology, Bot Binary Obfuscation, Anonymization, IP Flux, Information Hiding.
Syntactic		Bot Binary Obfuscation, Information Hiding, Encryption.
Horizontal Correlation	Centralized topology, Purpose.	Traffic Manipulation, Information Hiding, Decentralized topology.
Vertical Correlation	Purpose, Propagation, Bot Evasion, Rogue DNS Server.	C&C Evasion, Binary Obfuscation.
Host-based		Traffic Manipulation.
Network-based	C&C, IP Flux, Rogue DNS Server.	Protocol Manipulation, Traffic Manipulation, Information Hiding.

the three facets, *Bot Detection*, *C&C detection* and *Botmaster Detection* of botnet detection mechanisms, respectively. We do not aim to explain each row in the tables. Instead, we highlight some interesting observations.

Binary Obfuscation and *Encryption* degrade the accuracy of detection

Table 4.2: The effect of botnet behavioral features on C&C detection mechanisms.

C&C Detection Mechanism	**Related Botnet Features**	
	Improve	**Degrade**
Injection		Encryption.
Suppression		Traffic Manipulation.
Syntactic		Encryption.
Correlation	Centralized Topology.	Traffic Manipulation, Information Hiding, Protocol Manipulation.
Behavior-based	Rogue DNS.	Traffic Manipulation, Information Hiding.
Statistical Approaches	Neoteric Protocol.	Existing Protocol, Traffic Manipulation, Information Hiding, Protocol Manipulation, Encryption.

Table 4.3: The effect of botnet behavioral features on botmaster detection mechanisms.

Botmaster Detection Mechanism	**Related Botnet Features**	
	Improve	**Degrade**
Marking		Traffic Manipulation.
Logging		Anonymization.
Stepping-stone Detection	Stepping-stone.	C&C Evasion.

approaches that rely on reverse engineering of bot binary or C&C. *Syntactic* detection approaches leverage on known malicious patterns in host or network data. *Infiltration* and *Injection* require understanding of the bot binary and/or C&C protocol. For *C&C Server Hijack*, the IP addresses of C&C servers for subsequent hijack can be retrieved by analyzing bot binary in which these can be possibly hardcoded. Another way for discovering such IP addresses is to monitor C&C communication. *Binary Obfuscation* complicates inference of the bot binary while *Encryption* renders C&C indecipherable.

The choice of botnet topology makes some detection approaches more effective. *C&C Server Hijack* is more effective for botnets utilizing *Centralized* structure. Taking down the central C&C server(s) incapacitates the entire botnet. *Horizontal Correlation* associates group activity and communication patterns with botnet behavior. Similarity in communication patterns is particularly evident in the case of *Centralized* botnets. *C&C Server Hijack* is not an effective detection approach for *Decentralized* botnets as any node can be used to float commands. At best, hijacking a supernode can reveal a small portion of the entire botnet.

All bots will ultimately carry out the *Purpose* for which they were employed by the botmaster. The same, particularly if carried out in an aggressive manner, can hint at botnet infection. *Correlation* based on host activity is more effective for botnets with aggressive *Purpose*. For example, botnets targeted at *Unsolicited Marketing* and *Network Service Disruption* are more likely to generate traffic that would be feasible to be detected by network traffic analysis than botnets focused on *Information Gathering*.

Detection approaches that rely on botnet command-response behavior or observe similarity in communication patterns and flow characteristics (e.g. packet size, inter-packet arrival times and flow duration) degrade if a botnet makes use of *Traffic Manipulation*. This includes some *Host-based* detection approaches, *Network-based* methods, in particular those that rely on traffic classification, and *Suppression* in case of C&C detection. Use of *Marking* for botmaster detection experiences loss in accuracy if *Traffic Manipulation*, especially timing perturbations and traffic padding, is employed for C&C evasion.

Protocol Manipulation affects detection techniques that are based on observation of traffic anomalies or possible C&C communication. *Information Hiding* techniques make it difficult to detect botnets by analyzing network traffic for the presence of C&C communication. *Behavioral Analysis*, in particular *Network-based* methods that rely on traffic classification, degrade considerably because of *Protocol Manipulation*.

IP addresses of C&C servers can be discovered by monitoring C&C com-

munication. *Anonymization* and *IP Flux* make it difficult to trace C&C communication to C&C servers. *Logging* mechanisms use information stored by routers to trace attack packets back to the source. Logging mechanisms do not help much in the presence of *Anonymization* which hides the source of a network flow by bouncing packets through a network of volunteer servers before delivering them to the destination.

The choice of protocol for C&C communication is closely related to the effectiveness of some detection approaches. C&C based on *Neoteric* and unpopular protocols can be more conveniently detected than C&C that utilizes popular, *Existing* network protocols such as HTTP. Statistical approaches, particularly Machine Learning, have been used to classify network traffic and subsequently detect C&C by observing anomalies in individual application traffic. Their accuracy degrades if C&C communication employs popular *Existing* protocols. Additionally, the task of machine learning algorithms is made significantly difficult by the use of *Traffic Manipulation*, *Information Hiding*, *Protocol Manipulation* and *Encryption*. Use of *Existing Protocol* for C&C further complicates telling benign traffic and C&C apart. The aforementioned features have a similar effect on *Stepping-stone detection*.

Botnets employ *Rogue DNS Servers* to serve phishing content and to avoid blacklisting services. *Behavior-based* detection techniques reinforce confidence about botnet C&C by looking for suspicious behavior, such as the use of *Rogue DNS Server* for resolving domain names instead of the host's default DNS server. In the context of *IP Flux*, detection of FFSN by monitoring DNS responses to suspicious domain names offers two benefits. It can help in detecting the compromised machines being used as proxies to host malicious botnet services. Secondly, DNS queries to web services hosted by FFSN can indicate that the querying machine is also compromised. Moreover, DNS queries directed to known *Rogue DNS Servers* can reveal bots.

Some detection methods such as *Vertical Correlation* correlate events in a temporal fashion. Detection of malware *Propagation* and unusual behavior such as *Security Suppression*, *Anti-analysis*, *Rootkit* installation and use of *Rogue DNS Server* hint at botnet infection. Observing *Purpose* of a botnet, in particular its attack behavior, adds to confidence of the detection approach. *C&C Evasion* techniques degrade vertical correlation if potential C&C traffic is one of the inputs to the correlation engine.

These tables can be best utilized by identifying complementary detection approaches that compensate each other's shortcomings and highlight their respective strengths. Development of a comprehensive strategy along these lines will help in generic and more effective botnet detection.

4.6 Summary

In the event that an intrusion attempt succeeds, the very least a robust security plan should address is its detection. With this objective in mind, in this chapter, we provide an extensive overview of botnet detection mechanisms. To highlight the component of botnet infrastructure that a detection technique is targeting, we classify botnet detection mechanisms as those concerning *bots*, *C&C communication* and *botmaster*.

Bot detection refers to detection of botnet compromised machines without regard to the larger superset of botnet population of which the compromised host is a member. Based on the degree of participation of the defender in botnet operation, bot detection strategies can be further classified as *active* and *passive*. *Active* bot detection requires the defender to impersonate as a component of the botnet. Examples include botnet *infiltration* and *C&C server hijack*. *Passive* detection approaches detect botnets by silently observing and analyzing their activities without making a conscious effort to participate in the proceedings. This analysis can leverage comparison with pre-determined patterns of botnet infection derived from observed samples (*syntactic* analysis) or the context of events and protocol information (*semantic* analysis). The latter can further entail *correlation* and *behavioral analysis*.

A number of botnet detection mechanisms focus on *detection of C&C channel*. *Active detection* involve taking part in the botnet operation, for example, online manipulation of network flows to deduce information about possible C&C communication. *Suppression* and *Injection* represent subclasses of active mode of C&C detection. *Passive detection* involves silently observing network traffic, looking for cues of C&C communication. This may entail comparison with signature-based *syntactic* analysis) or behavioral (*semantic* analysis) characteristics of C&C traffic. The latter can be further broken down into *statistical* and *behavior-based* approaches.

Some detection mechanisms target the botmaster. *Active detection* involves manipulation of botnet activity, for example, insertion of information into packets either by victim machine or intermediate routers to help locate the attacker (*marking* techniques). *Passive detection* involves analysis of network traffic without manipulation / modification of botnet operation. *Logging* and *stepping-stone detection* represent subclasses of passive botmaster detection.

We argue that the high-level classification of botnet detection mechanisms, i.e., bot, C&C and botmaster detection (referred to as facets of botnet detection) can be further categorized with different dimensions depending on interest of the reader. In our taxonomy, we classified all the facets of botnet

detection with the dimension *level of activity*, i.e., *active* and *passive* detection. We identify a number of other dimensions for classification of botnet detection techniques; *degree of automation*, *analysis direction*, *analysis depth*, *specificity*, *discernment*, *mode of operation* and *location of deployment*. We leave detailed discussion as an avenue for future research.

We also analyze how the absence or presence of different botnet behavioral features affect the accuracy of botnet detection mechanisms. In this regard, we present three tables that investigate the effect of botnet behavioral features on the three facets (bot detection, C&C detection and botmaster detection) of botnet detection mechanisms. We assert that this information can be utilized to devise a comprehensive detection strategy that combines complementary detection approaches.

Chapter 5

Taxonomy of Botnet Defense Mechanisms

Strategy without tactics is the slowest route to victory.
Tactics without strategy is the noise before defeat.

— Sun Tzu, "The Art of War"

This chapter presents the last taxonomy in a series of three taxonomies pertaining to different aspects of botnets: behavior, detection mechanisms and defense strategies. Our third taxonomy classifies botnet defense mechanisms.

Defense against botnet infection is a two-fold concept. First, it refers to proactive measures to prevent the intrusion. Secondly, it also encompasses the mechanisms to recover from botnet infection and / or impair the botnet. In this chapter, we provide an extensive overview of various botnet defense mechanisms.

5.1 Overview

Botnet defense outlines the preventive measures that can be taken to keep infections at bay. Preventive mechanisms may not always succeed as botnets represent a moving target with constantly evolving behavior. Botnet defense mechanisms also encompass methods to recover from an infection either by cleaning the infected machines or impairing the botnet itself.

A number of studies give useful insights into botnet defense but do not span the entire gamut of possible botnet countermeasures. Furthermore, there is no elaborate structure in the way this information is presented. Resultantly, there is a chance that the reader will be overwhelmed and confused as the discussion becomes more involved.

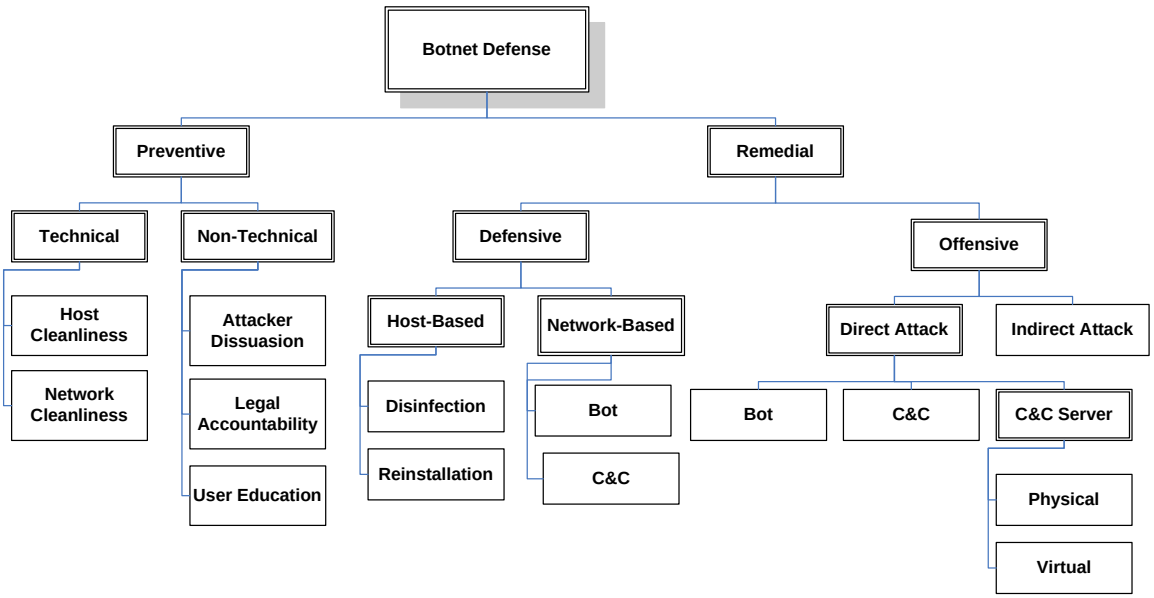


Figure 5.1: Taxonomy of botnet defense mechanisms.

We present a systematic study of botnet defense mechanisms. We broadly categorize botnet defense mechanisms as *Preventive* and *Remedial*. *Preventive* mechanisms strive to proactively defend against possible botnet infection by hardening existing security. *Remedial* defense mechanisms are reactive in nature. Such techniques address the problem by removing the infection from infected machines or networks or by damaging the botnet infrastructure. The rest of this chapter further classifies these high level classes of botnet defense mechanisms.

5.2 Preventive

Hosts and networks can adopt preventive measures against botnets to raise the bar for possible botnet infection. These methods are effective before the botnet infection has taken place.

5.2.1 Technical

Technical approaches include botnet defense activities and measures that are related to computers and networks.

Host Cleanliness Botnets spread by exploiting operating system or application vulnerabilities on victim machine or by social engineering. Measures

for host cleanliness should be adopted to proactively defend against botnet infection. A reasonable patch management system should be in place to install the latest security patches for operating systems and applications. Operating systems and most applications now come with the auto-update feature to relieve users of the responsibility of manual updation. Infection can be warded off by avoiding opening emails from unknown sources, particularly if the email contains attachments with executable files or scripts. Similarly, one should exercise restraint in clicking links on untrusted websites. Security settings of web browsers should be adjusted optimally and automatic execution of scripts such as JavaScript, ActiveX and VBScript should be turned off. Programs and users should be given only basic permissions in accordance with the principle of least privilege and administrative rights should be exercised with discretion. Installation of security software such as anti-virus , anti-spyware, anti-trojan, rootkit detection packages and firewall can help fend off some infections. The former can be complemented by Host Intrustion Prevention System to protect from previously unknown threats by identifying anomalous behavior. Closing ports used by applications favored by botnets for C&C communication such as IRC and FTP can reduce the risk of botnet infection to some extent.

Network Cleanliness While network administrators can benefit from the previous section to defend individual hosts in the network, there are some network level preventive steps that can further reduce the risk of botnet infection. The principle of separation of privilege dictates that multiple levels of security make it non-trivial for the attacker to circumvent it. Same is true of networks. In addition to host-based protection, servers, external connections, email gateways should all have optimum security in place. Network Anomaly Detection System and firewall are good options for proactive defense against the botnet threat. Access to known malicious domains should be blocked. A list of such domains (malicious/C&C/RBN domains) can be obtained from the internet. Network traffic should be made to pass through a web proxy where it is possible to scan incoming content for malware presence and outbound content can be scrutinized for possible data leaks. Placing honeypot and/or darknet in the network can indicate vulnerability of the network to certain threats and can point to the presence of infected machines in the monitored network. Finally, a comprehensive network security policy should be developed and enforced.

5.2.2 Non-Technical

These approaches fall under domains which are not directly related to the digital world of computers.

Attacker Dissuasion It has been proved time and again that there is increasing financial motivation behind botnet operation. Several studies [127, 128] suggest the lucrativeness of the business of spam, clickfrauds and other malicious activities carried out by botnets. Targeting the business model employed by botnets can reduce incentives for the bot master to run the botnet. MARK (the Multihost Adware Revenue Killer) was a distributed network of machines capable of controlling advertising impression numbers, click through rates and software package installs, to carry out effective attacks against malicious-code generated revenue streams [129]. The motivation was to make the botmaster give up maintenance of the botnet by reducing advertising revenue generated by adware and botnets. A number of anti-spam approaches have been tendered [130, 131, 132]. Detecting and subsequently filtering spam email before it reaches the intended recipients can hurt the profitability of this model of marketing to advertisers. In turn, the botmasters will lose business and give up the botnet. Current defense mechanisms against DDoS are not very effective [133]. Better preventive strategies against DDoS that stop the attack before it affects the targeted service will reduce the utility of the botnet to the botmaster.

Legal Accountability It is important to complement existing defense mechanisms against botnets with a solid legal framework. As long as the botmaster enjoys relative impunity over the internet, technical detection and mitigation efforts alone will not suffice. Microsoft's takedown of Rustock botnet [85] reinforces the frequently voiced concern of security organizations for more comprehensive legislature to tackle the botnet phenomenon [134]. USA, EU countries, China and Japan have strict penalties for cyber crimes [35]. Despite this, USA, Germany and France were among the top three countries hosting C&C servers in a survey conducted by security firm Damballa [135]. C&C servers tend to be scattered around the globe in countries with different legal attitude towards cyber crime, therefore taking down a server in one country will not decapitate the whole botnet. Botnets are a distributed phenomenon and therefore there is a strong need for collaborative legislation and cross-border cooperation to fight botnets. The legal consequences of cybercrime can potentially deter botmasters from operating botnets.

User Education All the technical defenses will ultimately falter if the user has no motivation to follow them. Generally, users are not concerned about the security of their computers as long as they can perform their

routine activities without hindrance. People are often the weakest link in an otherwise secure network. Botnets exploit this inherent weakness to infect systems by use of social engineering tactics. Some countries legally bind users to act responsibly by imposing privacy standards and fines for negligent or willful non-compliances [136]. Users should be educated about the gravity of the botnet threat and the measures they can take to avoid becoming bots.

5.3 Remedial

Remedial defense strategies help in partial or complete recovery from botnet infection. These mechanisms try to solve the problem either by removing the infection from infected machines and networks (defensive) or damaging the botnet infrastructure in a way that it either stops or significantly complicates perpetration of malicious activities (offensive).

5.3.1 Defensive Strategies

This class of methods is aimed at self-recovery in the event of botnet infection. Defensive strategies can be further classified as *Host-based* and *Network-based*.

Host-based Host-based techniques help individual machines recover from botnet infection. Mitigation techniques that target C&C servers do nothing for the infected bots. Bots will continue to experience negative effects of the infection and can even unwittingly participate in a never-ending cycle of malicious activities if they fail to receive the stop command from the exanimate C&C server for a previously issued start command. Host-based defense seeks to restore individual bots to their clean state. Host-based defensive strategies can involve *Disinfection* or *Reinstallation*.

One of the defensive methods for recovering machines from botnet infection is *Disinfection*. There are off-the-shelf programs dedicated to disinfecting compromised systems, however their effectiveness in completing eliminating the botnet infection is uncertain. Determining all the services, files and registry keys that have been changed or installed does not guarantee total disinfection particularly if rootkit installation was involved. Master Boot Record (MBR) infections are notoriously difficult to get rid of and disinfection efforts may prove futile. In most cases, OS reinstallation is required.

Another option to tackle the problem of botnet infection is *Reinstallation*. In majority of cases, botnet infection cannot be entirely removed and users need to restore the operating system to an uncorrupted previous state. If a clean image is not available, complete reinstallation might be required.

Network-based Network-based techniques aim to secure networks once it is known that one or more machines in the network are infected with botnet. These techniques can be categorized into those that *Block Bots* or *Block C&C Communication*.

Botnet-infected networks can be secured by blocking *Bots*. To contain the infection, the administrator can quarantine the infected machines in the administered network and fix them before bringing them back on the network. This is similar to the notion of walled-garden. Walled-garden is a state of isolation in which an ISP can place machines showing symptoms of botnet infection to reduce further damage to other machines inside or outside the network. The infected machine is denied all network communication except with a white-list of domain names helpful for remediation of the infection. The bot machine is not brought back on until its cleanliness is confirmed as per the ISP policy. The Messaging Anti-Abuse Working Group (MAAWG) described the concept of walled-garden as a three step process comprised of detection, notification and remediation [34].

Network-based defense strategies also involve blocking *C&C* communication. A network can be protected by identifying C&C communication and then blocking it. The task can be done in a pro-active fashion by making use of up-to-date URL and IP blacklists. However, identification of C&C traffic within the network traffic is not a trivial task. In the simplest form, traffic to and from known malicious ports should be blocked. For finer results, traffic analysis might be required. Alternately, all network communication can be blocked except a whitelist of known 'good' domains till the infected machines are cleaned.

5.3.2 Offensive

This class encompasses aggressive defense mechanisms that intimidate or completely paralyze a botnet. This can be done by carrying out an *Indirect* or a *Direct Attack* on botnets.

Indirect Attack Indirect attack reduces usability of the botnet to the botmaster. Unlike malware of past whose purpose was to create mayhem on internet, there are greater financial incentives for botnets in carrying out their malicious activities. Attacking the underlying business model to significantly reduce profits has the potential to daunt the bot master to an extent it gives up maintenance of the botnet. Some botnets steal sensitive information, such as passwords and banking credentials. The stolen information is transferred from the victim's computer to a dropzone and ultimately sold to third parties. The botmaster's credibility can be hurt if fake information is

injected into the dropzone. This can be done by reverse engineering the botnet's C&C protocol. This will damage the botmaster's reputation and affect his business prospectives. Ormerod et al. [137] suggested injecting fake but tracable credentials into the botnet dropzone so that the misfeasors could be identified and prosecuted.

Direct Attack Direct attack entails mechanisms to directly hurt one or more components of the botnet. On the basis of target of the attack, this class of remedial mechanisms can involve attack on *Bots*, *C&C Communication* and *C&C Servers*.

A direct attack can be staged on *Bots*. Like other software, the bot binary and C&C protocol are likely to have bugs [138, 139]. These can be exploited by the defender to take control of bots and remotely disinfect them or trigger the C&C command related to removal of the bot from the botnet. There are legal and ethical issues involved in remotely disinfecting a user's computer without his/her consent and may be seen as privacy invasion.

Another option is to coordinate a direct attack on *C&C Communication*. The C&C communication can be poisoned by introducing bogus commands in the channel. In p2p botnets, bots search new commands by using bot command keys. Holz et al. [22] presented a way to disrupt botnet communication by injecting benign content with command keys same as those used by botnets so that the actual commands never reach the bots. Sybil attack [140] is an attack against p2p botnets that introduces into it peers under control of the defender. The sybils can be strategically positioned to gain control over a part of the p2p network. They can reroute queries to any non-bot node or a peer sybil. In this way, the sybils can disrupt a part of the overall C&C communication. Eclipse attack [141] is similar to Sybil attack except that it gains control over a much smaller portion of the p2p network. Inference of the C&C protocol enables the defender to alter and inject commands into C&C that help minimize the damage caused by botnet. The specific commands for carrying out attack such as spam and DDoS can be stopped from reaching bots and replaced by commands to remove bots from the botnet.

Direct attack can also be launched on *C&C Servers*. This involves *Physical* or *Virtual* removal of C&C servers.

Physical removal of C&C servers can potentially render the botnet dysfunctional. However, the task is easier said than done. It is applicable to centralized botnets only and requires prior knowledge of the location of the C&C servers, which itself is an arduous task. There are ethical and legal hurdles involved in taking down servers and requires cooperation from hosting providers. Most C&C servers are hosted by providers claiming bullet-proof hosting that are immune to the nature of content being hosted. This neces-

sitates legal intervention.

It is also possible to perpetrate *Virtual* take over of C&C servers. Physical seizure is not always necessary and the desired effect can be achieved by other means. Border Gateway Protocol (BGP) blackholing is a technique to null-route traffic to and from known malicious servers. The same can be done through DNS Sinkholing. For DNS queries for known C&C servers, the sinkhole returns a non-routable or any address except the actual address of the malicious server. Alternately, defender has the option to carry out DoS attack against known, centralized C&C server to degrade its service.

5.4 Summary

Botnet defense comprises mechanisms that help in prevention of botnet infection. This is not always a dependable option as botnets represent advanced threat and the intrusion may succeed despite proactive security measures. Botnet defense also covers reactive techniques such as recovery from botnet infection or offensive measures to disrupt / impair the botnet. This chapter presents a comprehensive taxonomy of botnet defense mechanisms. Botnet defense mechanisms can be broadly classifies as *Preventive* and *Remedial*.

Preventive mechanisms strive to discourage botnet infection proactively. These can be further classified as *technical* and *non-technical*. *Technical* mechanisms advocate hardening of computer and network security (*host cleanliness* and *network cleanliness* respectively) to ward off possible botnet infection. *Non-technical* mechanisms address botnet infection through non-computer related means, such as *attacker dissuasion*, *legal accountability* and *user education*.

Remedial defense mechanisms are reactive in nature. These can be further classified as *defensive* and *offensive*. *Defensive* strategies are concerned with recovery from botnet infection. This could involve *host-based defense measures*, such as host *disinfection* and software *reinstallation* or *network-based defensive measures*, such as *isolating bots* or *blocking C&C communication*. *Offensive* strategies comprise of aggressive defense mechanisms that aim to disrupt a botnet. This can be achieved by carrying out a *direct* or *indirect* attack on botnets. *Direct attack* entails mechanisms to directly hurt one or more components of a botnet. *Indirect attack* reduces the usability of the botnet to the botmaster so that there is little incentive to carry on its maintenance.

At present, defense against botnets is mostly preventive or defensive. Preventive defense includes proactive measures to avoid botnet infection. Defensive methods are reactive in nature and concern themselves with cleaning

systems once they have been infected. There is an aching need for developing defense strategies that solve the problem at its root. While bot disinfection is important from user point-of-view, it does not hurt the botnet which finds other machines to infect and serve its purpose. Ideally, a defense method should incapacitate the whole botnet or significantly damage it.

Chapter 6

BotFlex–A botnet detection tool

A weapon which you don't have in your hand won't kill a snake.

— African Proverb

Botnets pose the most potent threat to the security and integrity of networked systems. Realizing the gravity of this threat, the academic community has produced scores of research papers and reports to explain botnet behaviors, topologies, sizes, detection parameters, defense strategies, and future trends. However, to date *no* open source tool exists that can be used to easily implement and test new botnet detection mechanisms. Consequently, a novice researcher has no option but to implement existing literature from scratch, even if she means to test a single parameter that can help in botnet detection. With this significant barrier to entry, the most impactful botnet research remains largely confined to security labs and companies.

In this chapter, we articulate the need to develop an open source botnet detection tool to accelerate research in this area. We implement and release this tool[1], with an extensible architecture that we hope allows other researchers to experiment with, and compare, different detection techniques. Finally, we validate the performance of BotFlex over a groundtruth traffic dataset and provide initial insights that point to the benefit of using such a tunable and open source botnet detection tool.

[1] `https://github.com/sheharbano/BotFlex`

6.1 Overview

Despite remaining in the headlines for more than a decade, botnets continue to threaten the security of our digital infrastructure. While hundreds of research papers and reports related to botnets are published on a regular basis, it is surprising that to date *no* open source botnet detection tool exists. The options available to a researcher are either to purchase services of commercial security companies [15, 8, 16], select from a very small number of closed source tools [17], or implement existing algorithms from scratch. These difficult to realize options have inadvertently resulted in a culture where botnet research is carried out in isolation, without much comparison with other tools or techniques. Thus, botnet research fails to experience the uplift that results from fusion of ideas on an open source platform. A universally accessible platform of this kind offers the additional advantage of channeling time and resources towards improvement of existing technology rather than reproducing what already exists in literature.

In this chapter, we present BotFlex — an open-source, extensible tool for botnet detection — to bridge the above identified gap. In order to evolve with the rapidly-changing botnet threat landscape, we design BotFlex to achieve the goals of *flexibility* in addition and removal of detection features; *extensibility* in adding new decision and correlation elements; support for *forensics and analysis* with the capability to *react to events in real-time*.

We build BotFlex over the Bro intrusion detection system (IDS) [142]. Our choice of Bro was dictated largely by its modular design that allows easy implementation of our goals. We further provide an architecture where both the detection mechanisms (or Sensor) and their correlation policy can be modified. This architecture allows significant flexibility to a researcher to select and compare the performance of a botnet detection system with different combination of techniques and policies.

We use BotFlex to analyze hundreds of GBs of a home ISP's traffic, with ground truth provided by a commercial security company. We observe true and false positive rates of 89.6% and 28.6%, respectively. These are very positive results for a system currently using a handful of features and decisions elements from existing literature. We ran BotHunter, a closed-source bot detection tool, over the same data set and observed similar detection rates and performance, validating our implementation against the only freely available tool [17]. More importantly, we reveal interesting insights that result from parameter tuning and decision policy variations. For instance, we observe a true and false positive rate of 87.3% and 26.6%, respectively, when using just C&C blacklist matching. These rates change to 40% and 3.4% when we activate all detection sensors and declare a bot if a C&C blacklist match is

substantiated by another evidence. We also observe, through manual analysis, that 5% of hosts declared false positives based on our groundtruth list were actually malicious hosts. These observations, possible due to the tunability of BotFlex, point to interesting aspects of relying solely on C&C blacklists. Such insights substantiate our thesis that the security research community can benefit significantly from an open source platform for botnet research.

In section 6.2, we provide system overview of BotFlex while highlighting its design goals, system architecture and the design choices we made to meet the design goals. Section 6.3 provides details about how we implemented the core architecture of BotFlex. In section 6.4, we present evaluation of BotFlex.

6.2 BotFlex System Overview

In this section, we highlight design goals of BotFlex followed by a discussion on the system architecture. Finally, we describe the design choices we made to meet the design goals.

6.2.1 Design Goals

We begin by identifying the design goals of BotFlex.

- **Real-time handling of network events:** The tool should support real-time handling of network events to detect malicious behavior as and when it occurs. This provides the defender the opportunity to respond to the threat in a timely fashion.
- **Extensibility:** The botnet threat is rapidly evolving. Consequently, detection approaches need to be updated frequently. Therefore, the tool should be extensible so as to effortlessly accommodate new detection parameters.
- **Flexibility:** The tool should be flexible so as to be tweaked and tuned according to different specifications, such as deployment location and research analysis.
- **Forensics and analysis:** The tool should support extensive logging. This information is invaluable for network forensics. Moreover, it provides an opportunity to correlate historic information that might be missed during live analysis.

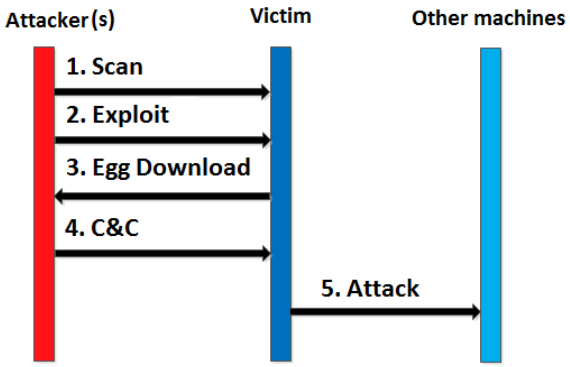


Figure 6.1: Typical botnet life cycle.

6.2.2 System Architecture

BotFlex is a network-based tool for detection of botnet infection. Conceptually, it has been built along the established model [46, 47] of bot lifecycle comprising scanning, host exploit, malicious binary download, C&C communication and attack phases (Figure 6.1). From the point-of-view of architecture, BotFlex has been built on top of Bro [143]. BotFlex intercepts input about various network activities from Bro and correlates/improvises on the received information to declare botnet infection.

In this section, we first discuss the architecture of BotFlex with respect to Bro, i.e., its placement within Bro and how it harnesses Bro's capabilities to converge to a botnet-specific detection platform. Next we discuss the core architecture of BotFlex, i.e., how BotFlex internally implements its logic for detecting botnet infection.

6.2.2.1 BotFlex Architecture with respect to Bro

BotFlex has been built on top of Bro. Bro represents a network activity as an event and generates hundreds of such events during its normal operation. BotFlex captures events of interest and ties them together into a botnet-centric detection tool. In particular, Bro's event model [144] perfectly maps to the design goals we laid out earlier in section 6.2.1 We briefly discuss Bro's architecture and then explain how BotFlex fits into it.

Bro's basic architecture comprises of three layers from bottom to top; Network, Event engine and Policy Script Interpreter (Figure 6.2). The network layer uses *libpcap* to capture network traffic. Packet filtering specification applied at upper layers flows down to this layer and thus only packets of interest are captured. The event engine comprises of various C++ scripts that pack network information into suitable data structures (e.g. each unique tuple of

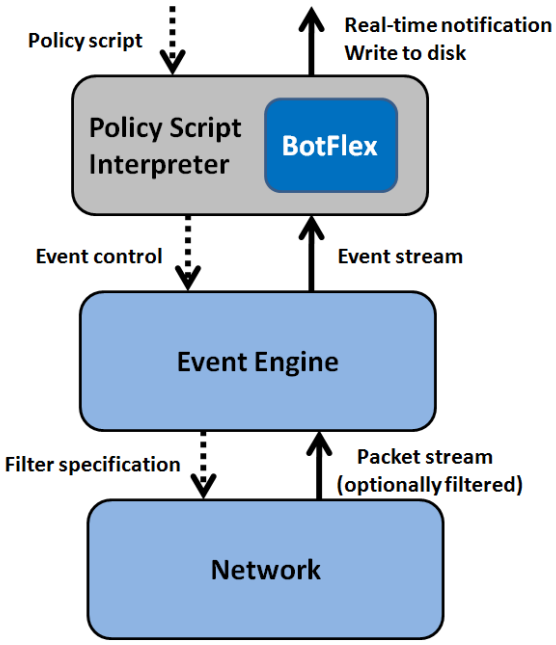


Figure 6.2: BotFlex architecture with respect to Bro

source IP, source port, destination IP, destination port and transport layer protocol represents an object of the data type *connection*). Similarly, network activities are represented by events (e.g. observing the RST packet for a connection results in the event *connection_reset*). Finally, the top layer presents a clean interface for handling network information and events pumped by the event engine. A high point of this layer is that it supports a domain-specific scripting language tailor-made to handle network-related data and activities. BotFlex resides at this layer and has been written in Bro scripting language.

Alternative approaches to Bro Botnet detection requires a fairly high-level semantic analysis of network activities. We briefly discuss the the different approaches we took before deciding to base BotFlex on an IDS and their pros and cons.

We developed a reduced prototype of the tool with a number of technologies. The list includes Python (with *Pypcap*, *Scapy*, *Impacket/Pcapy*), Java (with *Jpcap*) and C++ (with *libpcap*). We tested each prototype on two data traces (260 GB and 438 GB) collected at an ISP. The reason for testing on these data sets was to reveal potential issues that become apparent only on big, real world data traces with diverse traffic.

The primary drawback of using high-level languages (Java, Python) was their high processing time. This issue became all the more prominent with

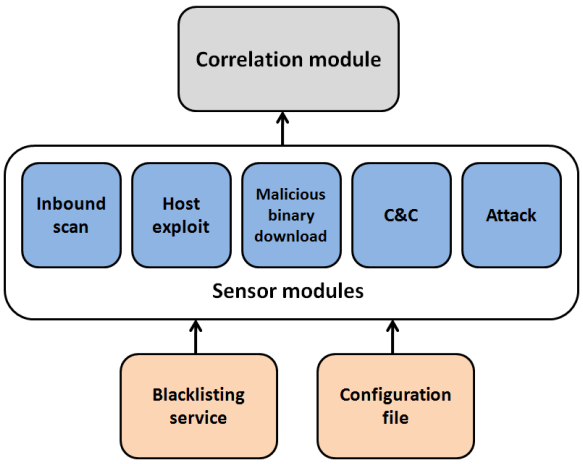


Figure 6.3: BotFlex core architecture

increase in the size of data.

Our other alternative was to use a low level language so we went with C++ with libpcap. While we experienced great improvements in processing times, building BotFlex in C++ had the same connotation as developing an intrusion detection system in C++ and then tying it to our botnet detection requirements. Finally, with both the high-level and low-level approaches, we could not achieve the *flexibility* aspect that we had visualized for BotFlex.

6.2.2.2 BotFlex Core Architecture

BotFlex derives from the established model of the lifecycle of a typical botnet infection. A brief overview of this model is as follows. A botnet discovers hosts to infect through scanning. Alternately, the infection might spread through other means, such as drive-by download websites, social engineering and exchange of infected media. Hosts are then exploited, typically by buffer-overflow attack on vulnerable services to initiate the next phase, malicious binary download. The malicious binary usually contains information about how to contact the C&C server, evasion mechanisms, outward infection and attack propagation. The C&C phase sees establishment of communication between the newly infected host and the C&C server. In the final phase, the bot spreads the infection to other hosts, and carries out malicious activities on behalf of the botmaster.

BotFlex implements bot lifecycle as an event chain, wherein a series of events can, in different combinations, trigger the final event of botnet infection declaration. It naturally follows that events can be either simple or complex. A simple event is an event that can take place in one and only one

way. For example, observing a local host communicating with an IP that has been listed on a blacklist is a simple event. On the other hand, a complex event comprises of multiple simple or complex events. The constituent events can combine in a logical AND, OR, or any other combination thereof. For example, the complex event 'malicious binary download' could be triggered if a host is seen to have downloaded an executable file with misleading extensions and the size of the file is less than 1 MB.

Keeping in line with our description of key phases in the life of a typical bot, we have split up the modules in our system into two basic types; Sensor modules and a Correlation module 6.3. Additionally, the tool has been equipped with a dedicated service that pulls blacklists from carefully selected blacklisting sources of high reputation.

Sensor Module A *sensor* module represents a phase in the life of a bot. We have sensor modules for malicious scanning, host exploit, egg download, C&C communication and outbound infection/attack. Output events from the sensor modules drive our correlation module.

Correlation Module This is the module that implements the core logic of when to declare botnet infection. Because of its significance, we added the notion of cardinality to the five backbone events previously discussed. This means that the evaluation logic can not only view whether a backbone event was present or absent, but also have information about how many times the event was reported for each host. This allows defining flexible criteria for declaring botnet infection.

6.2.3 Revisiting Design Goals

Having described the architectural details of BotFlex, we now explain how BotFlex satisfies the design goals laid out earlier in section 6.2.1.

Real-time handling of network events: BotFlex utilizes an *event-driven model*. We define an event as any interesting network activity that can (i) directly increase the evidence of botnet infection (e.g. observing DNS MX queries indicating spambot activity), or (ii) contribute to another event that hints at botnet infection (e.g. observing a large number of failed connections which is an indicator of malicious scan that in turn hints at botnet pre-infection or post-infection behavior, depending on whether the scan was inbound or outbound). The underlying platform of BotFlex, i.e., Bro inherently supports an event-driven model, thus facilitating BotFlex in meeting one of its primary design goals.

BotFlex incorporates the notion of *temporal awareness* by making use of data structures with expirable items. This is a very important consideration in the context of network traffic where events are churned at a high rate. As such information about outdated events needs to be discarded to make room for new ones.

Extensibility: BotFlex is based on the concept of *event chains*. This model allows for easy addition and removal of events.

Furthermore, BotFlex incorporates *logic decoupling*, i.e., a clear separation between event generation and handling logic. The event generation logic does not influence how it is handled and vice versa. This offers the benefit of adding and removing events as threats evolve without requiring modification of the core decision logic.

Flexibility: BotFlex uses *subscription model* for events, i.e., events at a higher level can subscribe to or unsubscribe from an event at a lower level. This implies that information flows in an upstream fashion from events at lower level to those at upper level. Bro's event engine is also based on a subscription model and thus integrates seamlessly with BotFlex's design goals.

BotFlex provides the feature of *tunable attributes*. The threshold values used to trigger various events and observation intervals (or windows) are configurable.

Forensics and Analysis: BotFlex records all events of interest in such a way that a parent event can be tracked right up to the lowest child event that contributed to its occurrence. The task is further facilitated by Bro's own extensive logging of information pertaining to different layers in the OSI model.

6.3 BotFlex Implementation Details

We provide details about how we implemented the sensor and correlation modules that comprise the core architecture of BotFlex (section 6.2.2.2).

6.3.1 Implementation of Sensor Modules

In this section, we discuss the detection parameters that we used corresponding to different phases of the bot lifecycle in the sensor modules. We have written our own policy scripts to implement required functionality. In some cases, we have modified or extrapolated policy scripts shipped with Bro. Any additional detection parameters that can increase the evidence of botnet infection should be added here.

6.3.1.1 Scan

Scanning is a precursor to host exploit in a large number of cases. The advent of p2p-based applications that legitimately scan a large number of IP addresses for peer discovery has blurred the definition of what constitutes malicious scanning. However, it is unusual for normal applications to carry out large- scale scans with the same destination port. The phenomenon is commonly know as portsweep. The probability of the scan being malicious further increases if the target port is not only uniform, but is also a historically high vulnerability port.

The above insights helped us develop a generalized and simple algorithm for scan detection. We maintain a hash table that is indexed by the tuple *source ip, destination port*, the yield/value of which is a set of addresses. Each entry in the table expires after an interval t has elapsed since the time it was created. We declare *event scan* when the number of unique hosts scanned by a host at a specific port exceeds a threshold. This check is performed each time a connection initiated by a host fails. We define a failed connection as one for which the traditional three-way tcp handshake was not completed. To give more weightage to portsweeps involving highly vulnerable ports, we define separate thresholds for medium severity portsweep and high severity (critical) portsweep. Information about high-vulnerability ports is obtained from DShield [145]. To keep up with the evolving threat landscape, this information is regularly pulled and fed into BotFlex through our blacklisting service. Note that we are not taking UDP scans into account because of processing overhead in addition to increased number of false positives.

6.3.1.2 Host Exploit

Host exploit involves activities that allow a remote entity to escalate its privileges and thus run arbitrary commands on the compromised machine. In most cases, the command executed relates to downloading and installing a malicious binary. BotFlex uses blacklists of hosts known to be involved in actively exploiting other hosts (push-style exploit) [146, 145, 147] and URLs serving drive-by-download exploits (pull-style exploit) [148]. BotFlex's blacklisting service regularly updates these blacklists.

6.3.1.3 Egg Download

An egg is a malicious executable (exe) program that typically contains instructions about (i) how to locate and communicate with the C&C server, (ii) how to spread the infection to other machines and (iii) how to evade detection. For now, we have used two parameters to detect egg-download.

Whenever an exe is seen to be downloaded from the Internet, its md5 hash is checked against TeamCymru's Malware Hash Registry (MHR) [149]. The MHR project is a look-up service that allows real-time querying for a computed MD5 or SHA-1 hash of a file. An appropriate reply is generated, depending on whether the hash matches one in the repository of known malware.

In some cases, an exe might not be listed in the list of known malware. This necessitates the use of some additional heuristics to distinguish between benign and malicious exe files. Our second parameter flags a file as suspicious if its content matches that of an exe file but it goes with a different extension. In the data trace we used for evaluation, we observed eggs being downloaded with the extensions *.txt*, *.php* and *.zip* among others.

Some other parameters we might include in future as indicators of maliciousness are the size of an exe file and whether the file was packed with packers known to be favored by malware authors.

6.3.1.4 C&C Communication

C&C communication refers to exchange of information between the C&C server and bots about updates, attack parameters and other behavioral aspects of the bot. The predominant method for network-based detection of C&C communication is matching external IPs with blacklists of known C&C servers or the notorious Russian Business Network (RBN). A number of services [150] on the Internet distribute these blacklists for no cost whatsoever. We maintain an up-to-date list of C&C servers and RBN address spaces through our blacklisting service.

In addition to C&C blacklists, signature-based methods are also employed to look for patterns in communication that are known to be associated with botnet C&C. Snort boasts of an extensive library of malware signatures. Unfortunately, Snort's signatures occur in a custom syntax (referred to as Rules) that cannot be trivially translated to other platforms. We advocate the need for a 'platform-independent' syntax for signature definition so that interested applications have the choice to consume them in accordance with their platforms.

Both blacklist matching and signature-based detection point to intelligence-based security which is an approach to curtail known threats. However, zero day or unknown threats go under the radar of these mechanisms. Therefore, it is important to augment blacklist and signature matching with identification of behavioral characteristics of C&C that remain independent of specific botnet instances. We identified one such parameter that is a good indicator of detecting botnets that use domain generation algorithm (DGA). Some

botnets such as Conficker [88] and Torpig [151] use DGA to spew a large number of domain names of potential C&C servers. The bot makes DNS queries about these domain names until one is found that resolves into an IP address. The resolved domain name corresponds to the C&C server. A side effect of the use of DGA for C&C identification is a large number of DNS failures, indicated by RCODE being set to NXDomain in DNS responses. We set a threshold on the number of DNS failed queries in an observation window. If the threshold is crossed, we suspect botnet C&C activity.

6.3.1.5 Attack

Attack phase represents the actual purpose for which the botnet was created (or rented). It includes a number of diverse malicious motives, such as information stealing, spam, DDoS attacks, infection spreading among others. BotFlex covers the spam and outbound scan aspects of the attack phase.

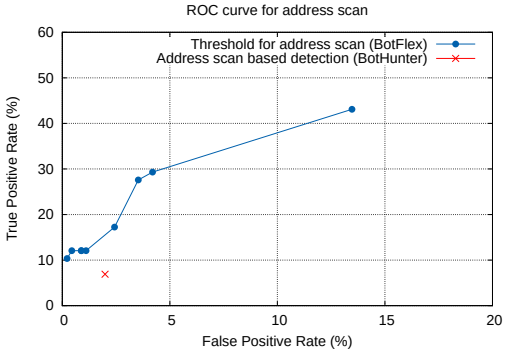
The mechanism for outbound scan is similar to the one used for inbound scan 6.3.1.1, except that here we are looking at scan initiated by an internal host targeted at external hosts.

For spam detection, we use the number of SMTP connections and DNS MX queries issued by a host as possible indicators of spam. MX queries are used to identify the mail server responsible for accepting emails on behalf of a recipient's domain. In case of internet-based email services, emails are transported from the client machine to one of the service provider's machine over HTTP. Most host-based email clients (also called Mail User Agent) are configured to hand off emails to mail servers. Thus, it is unusual for a machine that is not a mail server to generate MX queries. To avoid spam-filtering mechanisms employed by most mail servers, spambots attempt to send emails themselves. This results in generation of MX queries. Additionally, the number of SMTP connections exceed that generated by normal human usage. We tag a host as a spambot if the number of SMTP connections and MX queries generated by it exceed a threshold within an observation period.

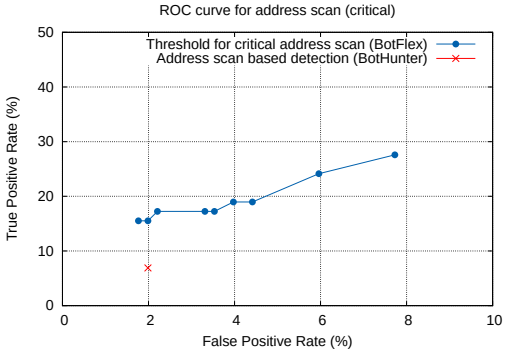
6.3.2 Thresholding detection parameters

Once BotFlex had been instantiated with some detection parameters, the next task was to find suitable thresholds for them. We decided to plot ROC curves for this purpose (Figure 6.2). The idea is that the best threshold for a parameter is one that yields the best balance between true positive rate (TPR) and false positive rate (FPR) among the available choices.

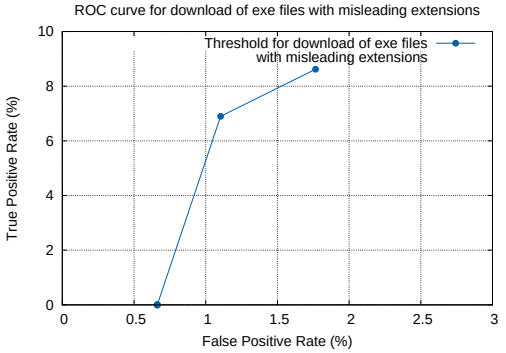
This, admittedly, is not the ideal approach to solve the problem. Botnets represent a multifaceted malware and do not exhibit uniformity in behavior



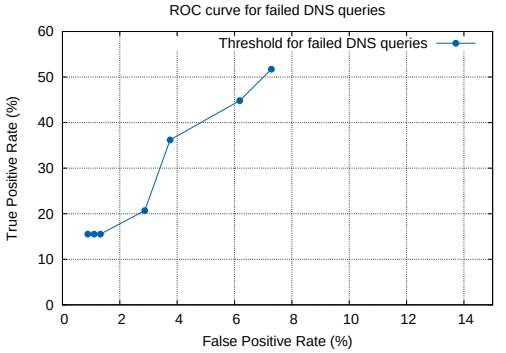
(a) Medium severity scan



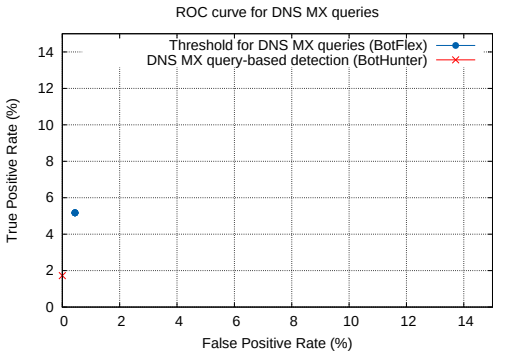
(b) High severity (critical) scan



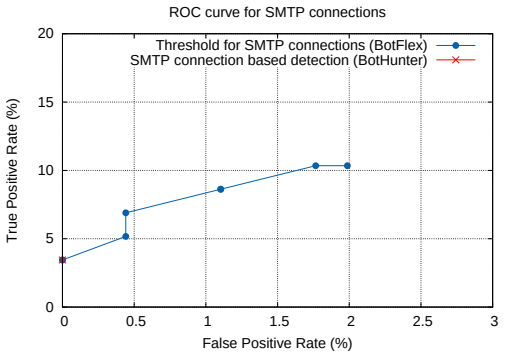
(c) Disguised .exe files downloaded



(d) Failed DNS queries



(e) MX queries



(f) SMTP connections

Figure 6.2: ROC curves for BotFlex detection parameters

like other malware such as worms and virus. For example,the absence of spam behavior does not reduce the probability of a host to be a bot. It is possible for a bot to engage in malicious activities other than spam. This means that the observed TPR for a parameter corresponds to a subset of the actual bot population. Similarly, false positives for a detection parameter

do not necessarily mean that our mechanism for its detection is flawed. For example, a host contributing to our false positives on the ROC curve for scanning could hint at worm infection (but not necessarily botnet infection). We can say that the FPR for a parameter corresponds to a superset of the entire phenomenon. That is so because the false positives correspond to both the cases when (i) a benign host has been mistakenly tagged with a detection parameter d (ii) a host was correctly identified by detection parameter d, but it was not part of a botnet.

We summarize our interpretation of ROC curves for botnet detection parameters as:

- The TPR could improve when aggregated with other trails of evidence in the correlation module.
- The FPR could reduce for lack of other trails of evidence in the correlation module.

Next we share some interesting observations related to the ROC curves. We plotted ROC curve for the number of executable files (Figure 6.3c) downloaded over an observation period of 30 minutes with threshold values ranging from 1 up to 6. We observed an improvement in FPR for threshold value of 2. However, for higher threshold values, TPR and FPR remained constant. We deduce that the number of disguised exe files downloaded is irrelevant from botnet infection point of view. A single malicious binary download can suffice as evidence of a successful infection. Therefore, we selected the threshold 1 for this parameter. We investigated the true and false positives at threshold values 1 and 2. It turned out that some legitimate files such as update patches for anti-virus software comprise of executable content but have different extensions such as *.bundle*,*.patch*, *.patchmanifest*,*.solidpkg* and *.kdl*. We whitelisted these extensions and then the ROC curve reduced to a single point, with TPR of 0% and FPR of 0.6%. Further investigation revealed that the 0.6% FPR comprised of actual malicious binary downloads later confirmed by TeamCymru MHR.

We observed TPR and FPR for DNS failed queries (Figure 6.3d) for threshold values between 3 and 50. The TPR for this parameter was good with a reasonably low FPR. We investigated the false positives and found out that some p2p clients make DNS queries for torrent trackers such as *denis.stalker, nemesis and opentracker* which frequently fail. However, the queried domain name remains the same for p2p clients, unlike the case of domain name generation in botnets. In future, we intend to correlate dns failed queries with the distribution of queried domain names to reduce false positives.

The ROC curve for MX queries (Figure 6.2e) turned out to be a single

point. We used threshold values ranging from 5 up to 35, but the TPR and FPR remained constant. It reinforces the existing notion that normal machines do not generate MX queries and it is a very distinct indication of spambots. We investigated the few false positives and those turned out to be for spambots that were not declared as bots as per our ground truth. More discussion can be found in section 6.4.4.

6.3.3 Implementation of Correlation Module

Correlation module incorporates the actual decision logic that triggers the event botnet infection based on input from Sensor modules. We did not assign any weightage to inbound scan in the final condition that decides botnet infection. From experiments, we felt that inbound scan is always there because of worms randomly scanning the Internet, or legitimate applications making peer discovery. Moreover, inbound scan itself does not guarantee whether a host will be exploited in future or not.

It is possible for some of the phases in bot lifecycle to be missing. For example, if malicious binary was installed on a host via infected media, the exploit phase would not be observed by BotFlex. We try different possible combinations of bot lifecycle phases to devise a suitable filter to detect bots. We define the condition *c_botnet_infection* for declaring botnet infection as:

```
c_botnet_infection = condition1 OR condition2 OR condition3
```

where

```
condition1 = Direct C&C blacklist match

condition2 = (Exploit OR Egg download) AND (C&C OR Attack)

condition3 = (Egg download AND C&C) OR (C&C AND Attack)

    OR (Egg Download AND Attack)
```

6.4 Evaluation and Results

In this section we discuss the approach we took for evaluating BotFlex. We begin by explaining our methodology for obtaining data and ground truth. We provide a macro evaluation of BotFlex, focusing on its overall performance and comparing the results with another botnet detection tool, BotHunter, for reference. Next we perform micro evaluation of BotFlex and observe the distribution of true positives and false positives with respect to

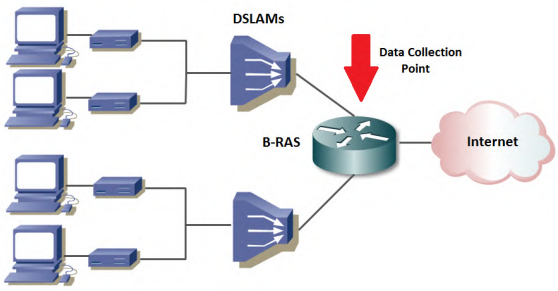


Figure 6.3: Data collection point (B-RAS)

various detection parameters used in its sensor modules and evaluation conditions in the correlation module. Finally, we share insights gained from our evaluation.

6.4.1 Evaluation Methodology

We evaluated BotFlex on 200 GB data obtained at one of the country's leading ISPs. The data was collected at the ISP's B-RAS corresponding to network traffic from 511 homes (Figure 6.3). We obtained ground truth for this data from a commercial company that specializes in security products and research. The company also maintains a proprietary threat repository of known command and control servers identified with the help of a chain of globally deployed sensors. We synchronized our data collection with the company's real-time C&C server monitors. The company provided us with a list of C&C servers that were being actively contacted by hosts from the monitored region for six hours starting from 18:48 GMT on 28th February 2012. For exactly the same interval, we collected data at the ISP's B-RAS. We labeled our data on the principle that the hosts in our data set that communicate with the ground truth C&C servers are bots. The experiment revealed that 58 (11.3%) of total 511 homes have been compromised by botnets.

6.4.2 Macro evaluation of BotFlex

We ran BotFlex on the labeled data set to observe its performance and pinpoint areas where further work is needed. For reference, we also ran another closed source, and to the best of our knowledge, the only free, publicly available botnet detection tool BotHunter on the same dataset. Note that we did not provide the ground truth blacklist to either of BotHunter or BotFlex.

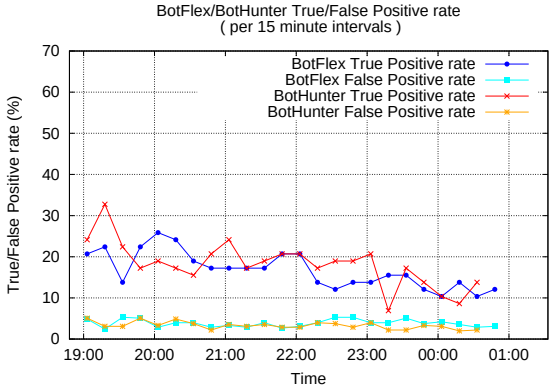
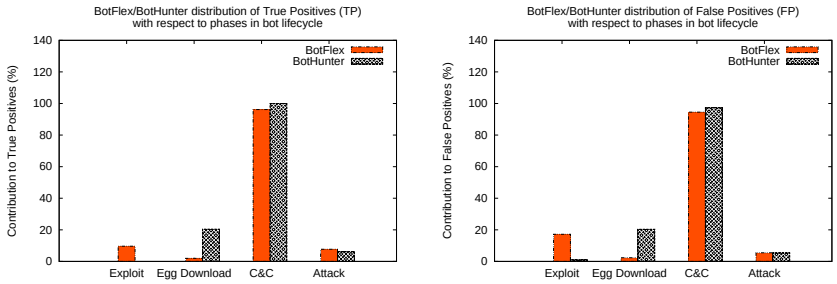


Figure 6.4: Detection rate of BotFlex and BotHunter over 15 minute intervals



(a) BotFlex/BotHunter True Positives

(b) BotFlex/BotHunter False Positives

Figure 6.5: Contribution of sensor modules to detection rate

The results of our experiment were surprising to say the least. Even with a bare-minimum set of detection parameters, BotFlex detected 52 of the ground truth 58 bots, with a true positive rate (TPR) of 89.6% and false positive rate (FPR) of 28.3%. On the same dataset, BotHunter gave a TPR of 84.4% and FPR of 24.9%. The TPR and FPR of BotFlex and BotHunter calculated over 15 minute intervals shows very similar results (Figure 6.4).

Next, we wanted to know how much individual sensor modules (corresponding to different bot lifecycle phases) contribute to botnet detection. We did this by analyzing the generated bot profiles for the number of times alert from a particular sensor module was present. To avoid redundancy caused by a bot being reported multiple times with different evaluation conditions coming out to be true, we considered alerts from a specific module for a bot only once. We did a similar analysis for BotHunter as it is also based on the model of typical bot lifecycle.

Our analysis revealed that the role of a sensor module towards true positives is proportional to that in false positives (Figure 6.5). The above observa-

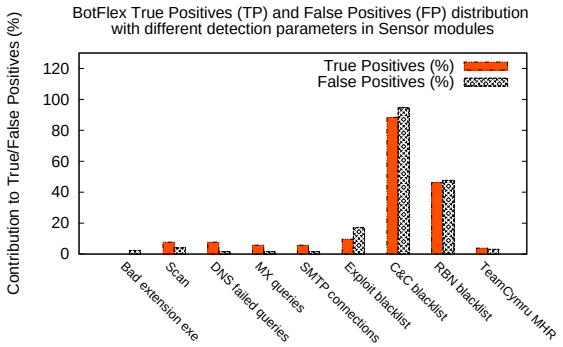


Figure 6.6: Contribution of detection parameters to detection rate

tion holds for both BotFlex and BotHunter. BotHunter and BotFlex behave proportionately in other departments except for exploit and egg download phase. BotHunter's handling of egg download is slightly more liberal than ours. In addition to other parameters, it triggers alert for for any executable file which has a size less than 1MB. This has the effect that in contrast to BotFlex, BotHunter's egg download phase has a significant contribution towards its true positives. On a downside, a significant number of BotHunter's false positives are generated by the same module.

As for host exploit, BotHunter uses anomaly-based engine to detect payload exploits. On the contrary, BotFlex uses a blacklist of hosts actively attacking other hosts on the Internet or serving drive-by-download exploits. The approach used by BotFlex yields more true positives than BotHunter's. However, the same effect is true for false positives too which is undesirable.

6.4.3 Micro Evaluation of BotFlex

We evaluate the accuracy of BotFlex with respect to detection parameters implemented within sensor modules. Furthermore, we analyze the role of different evaluation conditions in correlation module in the overall performance of BotFlex.

6.4.3.1 Evaluation with respect to detection parameters in sensor modules

We wish to gauge the efficacy of different detection parameters implemented within BotFlex sensor modules in detecting bots. This information will give us an idea about which parameters have the potential to improve accuracy of BotFlex and which should be removed or downplayed for lack of useful

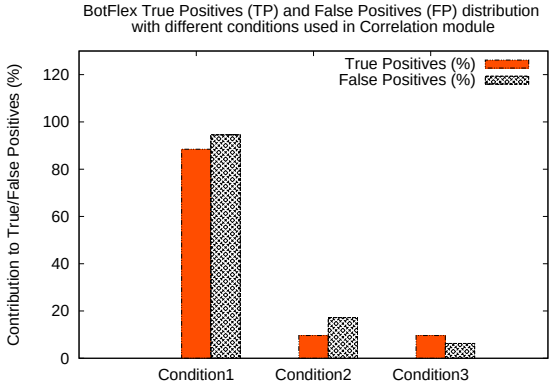


Figure 6.7: Contribution of correlation module to detection rate

contribution. For each detection parameter, we analyze the bot profiles in which it was present. Then we calculate how many bot profiles per detection parameter were false positives or true positives (Figure 6.6).

It turns out that C&C and RBN blacklists are major contributors towards true positives. Unfortunately, they have exactly the same degree of contribution towards false positives. Our detection parameters for spam detection, i.e., number of DNS MX queries and SMTP connections show promising results. The number of false positives generated by outbound malicious scan (for attack) and exe files with misleading extensions and TeamCymru's MHR lookup was larger than expected. Analysis revealed that these false positives could possibly correspond to actual bots not tagged as such by our ground truth (section 6.4.4.2).

6.4.3.2 Evaluation with respect to correlation module

We wish to identify the evaluation condition(s) that helps BotFlex achieve its maximum accuracy. For this purpose, we investigate the contribution of each evaluation condition to true and false positives generated by BotFlex (Figure 6.7). Condition 2 and 3 are somewhat eclipsed by condition 1. This is expected because our final evaluation performs an *OR* between the three conditions. As such, if the first condition of C&C blacklist match succeeds, the remaining two conditions never get a chance to trigger. Condition 1 incurs a lot of false positives. We felt that our false positives could be reduced by modifying condition 1 such that

```
condition1 = known C&C IP/URL match AND

<any other heuristic from a sensor module>
```

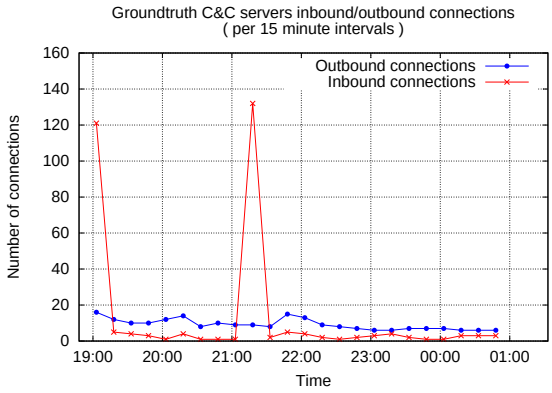


Figure 6.8: Traffic to and from C&C servers per 15 minute intervals

Running BotFlex with modified condition 1 on the data trace revealed a true positive rate of 40% and false positive rate of 3.4%. The false positives were significantly reduced, but there was a reduction in true positives too.

6.4.4 Insights

We share some insights gathered from running BotFlex on the data trace (described in section 6.4.1). We first discuss the trends in botnet C&C according to our data that represents traffic from 511 homes. We conclude the section by sharing some general insights about botnet detection.

6.4.4.1 Trends in Botnet C&C

Botnets are often described as coordinated malware [41, 40]. They exhibit group behavior in communication and/or activities. The similarity in communication is particularly obvious in the case of botnets with centralized C&C, such as HTTP. This can be observed as spikes in the number of outbound connections to external hosts as several bots connect with the C&C server at the same time. To check this behavior, we plotted the number of inbound and outbound connections from and to each one of the ground truth C&C servers per 15 minute intervals.

Instead of observing the expected spike in outbound connections to C&C server(s), we observed it in inbound connections from C&C server(s) (Figure 6.8). Further investigation revealed that the two very significant spikes were the result of inbound scan on port 1433/tcp (used for remote connections to Microsoft SQL Server database) by a single C&C server (Figure 6.9). We suspect that the C&C server in question was a 'do-it-all' server, where the same machine was used for finding out potential victims and later providing

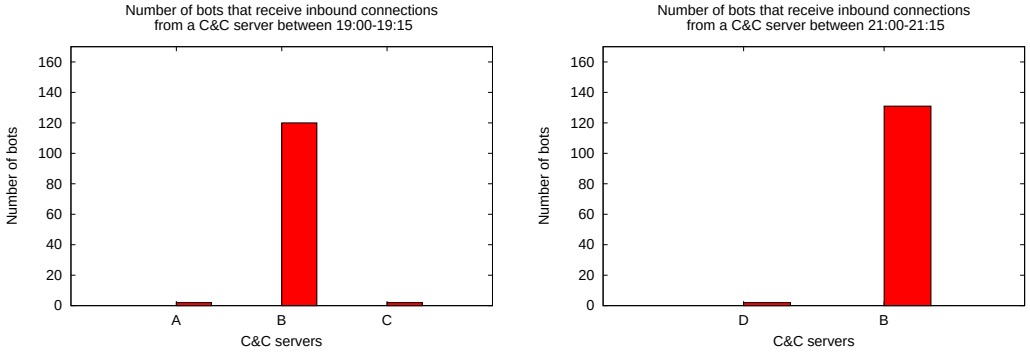


Figure 6.9: Spikes in inbound connections from C&C servers

commands to successfully infected machines. Moreover, it is typical of botnets to use mechanisms to conceal C&C servers because of their importance. The C&C server we observed was naive in exposing itself by performing aggressive scanning which can be easily detected.

We also analyzed IRC traffic for possible C&C communication. We used Bro's dynamic protocol detection (DPD) framework to classify IRC traffic. Of 511 hosts, only 3 were found to be using the IRC protocol. The extremely small amount of IRC traffic made manual analysis viable. We identified only one host that seemed like an IRC bot on account of its unusual nick *USA—X-123—0—XP—672955935*. The host in question was identified as bot by both BotFlex and BotHunter with evidence of outbound scanning on 445/tcp, malicious exe download and large number of outbound SMTP connections.

From our analysis of botnet C&C, we gather that the trend in botnet C&C has shifted to p2p or other hard-to-detect means of communication. In particular, IRC has been replaced by other protocols for botnet C&C.

6.4.4.2 Botnet Detection with C&C Blacklists

In our analysis we found some interesting insights into the widely prevalent mechanism of using C&C blacklists to detect botnet infections.

We first present the case of hosts flagged as false positives by our C&C blacklist, which on manual analysis (using the forensics support of BotFlex), actually appeared to be bots. For this purpose, we sliced all the alerts generated per false positive host from our log file. This resulted in 128 files where each file represented evidence trail that led the host to be declared as a bot. We wrote a script to parse all the files and analyze those wherein any alert other than direct C&C blacklist match existed. As a result of our analysis, we discovered 6 hosts that exhibited malicious behavior but were not treated

Figure 6.10: The profile of a 'false positive' bot

as such by the ground truth. The profile of one such host can be seen in Figure 6.10.

Separately, when we changed our declaration policy in Section 6.4.3.2 we observed that almost half of the bots communicate with a C&C server but do not engage in any (detectable by BotFlex) malicious activity. There can be two possible explanations for such a result. The first one is that some botnets intelligently spread out their malicious activities over a large time duration, with our six hour observation period insufficient to capture the entire gamut of botnet behavior. Alternately, it is possible for these bots to be just proxy bots [152]. Proxy bots may not directly take part in the malicious activities associated with a botnet. Rather, they act as relays on behalf of the C&C server. It seems that the only way to detect such tacit bots is to have external intelligence about them in the form of blacklists. The more aggressive bots can be detected with the help of detection approaches on a host or network perimeter.

Thus, these results highlight a need to augment C&C blacklists with behavioral techniques as together they provide a much broader scope of detection than in isolation.

6.5 Future Directions

BotFlex will be incorporated in *Bottleneck* [153]– a generic, extensible and flexible framework for botnet defense. Bottleneck has been designed to meet the following goals: (1) General and flexible in detecting different classes of botnets while adapting to deployment requirements; (2) Extensible to

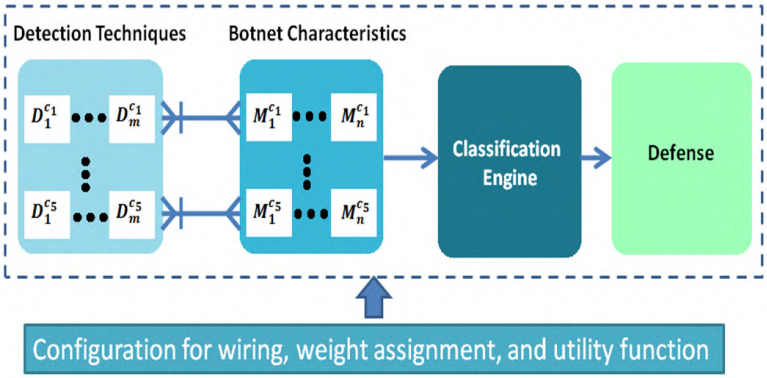


Figure 6.11: Bottleneck: An extensible framework for botnet detection

the evolving threat model by providing seamless integration of new botnet detection techniques; (3) Integrated with defense strategies which can be triggered after detection and classification of the threat.

We now discuss the architecture of Bottleneck and explain how it benefits from BotFlex. Bottleneck uses evidence from various botnet detection techniques to arrive at a decision about botnet infection, and triggers a suitable defense mechanism in line with the nature of the detected threat. As shown in Figure 6.11, Bottleneck leverages the observation that a set of five botnet characteristics — propagation, rallying, C&C, attack, and evasion — provide a complete and time-invariant high-level characterization of the botnet phenomena. Let $\{c_1, \cdots, c_5\}$ denote these characteristics. Botnet creators employ a number of mechanisms to implement each characteristic. For a given characteristic c_k, let $\{M_1^{c_k}, \cdots, M_n^{c_k}\}$ represent the set of existing and detectable mechanisms. For example, {scanning, client exploit, social engineering} represent a subset of all possible botnet propagation mechanisms, M^{c_1}. These mechanism implementations, for any given characteristic, can be detected by one or more of m detection techniques represented by, $\{D_1^{c_k}, \cdots, D_m^{c_k}\}$. Once a botnet detection technique declares botnet infection, the next task is for the classification engine to identify the type of the threat (e.g. based on IRC, HTTP, p2p). The output of classification triggers a suitable defense strategy. BotFlex can assume the role of the first component *Detection Techniques* (Figure 6.11) in Bottleneck. In particular, the extensible nature of BotFlex coincides with Bottleneck's design goal for general and flexible detection of different botnet classes while adapting to deployment requirements.

6.6 Summary

Botnets have been the cause of some of the gravest cyberthreats in recent times. As such, they have been the subject of a vast body of research involving their behavior, detection, defense, size, future trends. In this chapter, we presented BotFlex which, to the best our knowledge, is the first open-source tool for botnet detection. BotFlex is a flexible and extensible network-based tool that adheres to the established model of a botnet lifecycle. We build BotFlex over Bro intrusion detection system (IDS) [142]. Our choice of Bro was dictated largely by its modular design that allows easy implementation of our goals. Furthermore, it utilizes an event-based model for real-time detection of emerging threats and performs extensive logging to facilitate forensics and analysis. BotFlex currently uses a set of existing parameters to detect bots based on the established model of bot life cycle, involving the phases of scanning, host exploit, binary download, C&C communication and attack behavior. *Sensor modules* are responsible for detection of the different phases in botnet lifecycle. *Correlation module* implements the core logic of when to declare botnet infection.

We evaluate BotFlex on six hours of traffic captured at an ISP with ground truth provided by a leading security company. Even with a modest number of detection parameters and decision elements from existing literature, BotFlex demonstrates a TP rate of 89.6% with a FPR of 28.3%. For reference, these rates are at par with the closed source botnet detection tool BotHunter, which we ran on the same dataset.

We observed that the trend in botnet C&C has shifted to p2p or other hard-to-detect means of communication. In particular, IRC has been replaced by other protocols for botnet C&C. Moreover, our results highlight the need to augment intelligence-based security (C&C blacklists) with behavioral techniques as together they provide a much broader scope of detection than either one in isolation.

Chapter 7

Conclusion

**'They say that genius is an infinite capacity for taking pains,'
he remarked with a smile. 'It's a very bad definition,
but it does apply to detective work.'**

— Sir Arthur Conan Doyle, "A Study in Scarlet"

This chapter presents a summary of the contributions of this thesis (section 7.1). We condense key findings of this thesis in section 7.2.

7.1 Summary of Contributions

We recapitulate our problem statement and then provide a summary of contributions of this thesis. This thesis aimed to systemize existing knowledge of botnet behavior, detection and defense, and subsequently use this knowledge to develop a flexible, open-source botnet detection tool.

The first contribution of this thesis is the systematic analysis of the botnet threat from three aspects: botnet behaviors/architectures, detection mechanisms and defense strategies. In chapter 3, we presented our first taxonomy of botnet behavior. We explored the botnet phenomenon from different angles, such as propagation, rallying, C&C and purpose. We also enumerated evasion mechanisms employed by botnets to circumvent detection. Chapter 4 presents our second taxonomy of botnet detection approaches. Botnets represent a distributed threat. In our taxonomy, we classified different detection approaches with respect to the component of botnet that they target. In chapter 5, we proposed our third taxonomy of botnet defense mechanisms. The taxonomy was driven by a number of useful considerations, such as, whether a defense strategy is preventive or remedial, defensive or offensive and so forth.

The second contribution of this thesis is the development of BotFlex – an extensible open source tool for botnet detection to facilitate research in this area (chapter 6). To evolve with the rapidly-changing botnet threat landscape, BotFlex was designed to achieve the goals of *flexibility* in addition and removal of detection features; *extensibility* in adding new decision and correlation elements; support for *forensics and analysis* with the capability to *react to events in real-time*. BotFlex was used to analyze hundreds of GBs of a home ISP's traffic, with ground truth provided by a commercial security company. We observed true and false positive rates of 89.6% and 28.6%, respectively. We ran BotHunter, a closed-source bot detection tool, over the same data set and observed similar detection rates and performance, validating our implementation against the only freely available tool [17].

7.2 Conclusions

We conclude by condensing key findings of this thesis.

How botnet behavioral features affect botnet detection approaches We structured existing botnet literature into three comprehensive taxonomies of botnet behavior, detection and defense. The proposed taxonomies provide a comprehensive framework that could be utilized to understand the botnet problem and its solution space. Our taxonomies brought to attention some previously unknown, interesting facts. We noticed an inherent connection between botnet behavioral features and detection approaches. The selection of botnet behavioral features (from our first taxonomy) have a direct impact on the accuracy of the detection approaches (from the second taxonomy). In this context, we presented three tables that show how the presence of a botnet behavioral feature improves or degrades the accuracy of detection approaches (section 4.5.2). This is very important information from the point of view of a security researcher. Armed with knowledge of pros and cons of individual detection mechanisms, complementary approaches can be paired to achieve a synergistic effect. Additionally, informed decisions can be made in the choice of detection mechanisms for designing tailored strategies targeted at detection of specific threats.

Dimensions for classifying botnet detection approaches We proposed a number of classification criteria (dimensions) for structuring botnet detection mechanisms (sec 4.5.1). In our proposed taxonomy, we chose the dimension *level of activity*. Level of activity refers to the participation of a detection approach in botnet operation. Our choice of dimension is dictated by prevalence of policies that advocate passive detection and restrict active

methods, such as flow manipulation or impersonation. We leave classification by other dimensions as future avenue of research. To summarize, the notion of 'dimension' reveals various metrics of interest in evaluating botnet detection mechanisms.

A note on botnet defense Defense measures against botnets can be classified into three categories based on the component of botnet being targeted, i.e., (i) cleaning bots, (ii) disruption of C&C communication and (iii) shutdown of C&C servers. People are often the weakest link in an otherwise secure network. Botnets exploit this inherent weakness to infect systems by use of social engineering tactics. Host-based security software typically fail to defend against botnet infection. The second option of C&C disruption is losing relevance as botnets resort to more resilient technologies such as p2p and strong encryption. The last option of C&C server shutdown has yielded some promising results in recent times [85, 57, 81, 154]. However, the long-term efficacy of these methods is debatable [155]. Reactive defense approaches like removal of C&C servers yields promising results in the short-term. Botnets find a way to resurrect and resume their malicious activities. There is a pressing need to simultaneously address the botnet problem closer to the source (cleaning bots). Some countries legally bind users to act responsibly by imposing privacy standards and fines for negligent or willful non-compliances [136]. ISPs can play a significant role in this context. This notion is reinforced by China's success in handling its spam problem by extending liability to Chinese ISPs [156]. Users should be educated about the gravity of the botnet threat and the measures they can take to avoid becoming bots.

Openness in botnet research In this thesis, we articulated the need to develop a platform to accelerate research in the area of botnet detection and defense. It is a well-known fact that botnet creators exchange attack code, information about vulnerable hosts, rent resources and piggyback malware. Security community needs a similar degree of cooperation to counter the botnet threat. We note that to date *no* open source botnet detection tool exists. Thus, botnet research fails to experience the uplift that results from fusion of ideas on an open source platform. With this background, we developed BotFlex–an open source tool for botnet detection. Even with a modest number of detection parameters and decision elements from existing literature, BotFlex demonstrated a TP rate of 89.6% with a FPR of 28.3%. For reference, these rates are at par with the closed source botnet detection tool BotHunter, which we ran on the same dataset.

We used BotFlex to analyze a comprehensive ISP dataset with ground truth provided by a commercial security company. We found some interesting

insights into the widely prevalent mechanism of using C&C blacklists to detect botnet infections. Apart from contacting known C&C server(s), half of the ground truth bots did not engage in malicious behavior (detectable by BotFlex). Moreover, BotFlex detected 8 bots that were missed by ground truth. We confirmed through manual analysis that the flagged machines were indeed bots and not false positives. Thus, these results highlight a need to augment C&C blacklists with behavioral techniques as together they provide a much broader scope of detection than in isolation.

Bibliography

[1] L. T. Borup, "Peer-to-peer botnets: A case study on waledac," Master's thesis, Technical University of Denmark, Kongens Lyngby, Denmark, 2009.

[2] J. R. Vacca, *Network and System Security*. Syngress, March 2010.

[3] R. Deibert, A. Manchanda, R. Rohozinski, N. Villeneuve, and G. Walton, "Tracking ghostnet: Investigating a cyber espionage network," *Network*, vol. JR02-2009, no. JR02-2009, p. 53, 2009. [Online]. Available: http://tinyurl.com/d5q3cj

[4] T. Chen, "Stuxnet, the real start of cyber warfare?" *IEEE Network*, vol. 24, no. 6, 2010.

[5] Micorsoft, "Flame malware collision attack explained," http://tinyurl.com/dxxlb5j, June 2012, [Online; accessed 1-Aug-2012].

[6] J. Fielding, "25% of all computers on botnets," http://tinyurl.com/9e7bdkr, January 2007, [Online; accessed 12-December-2011].

[7] CIOinsight, "Botnets still a major threat," http://tinyurl.com/cw5bypo, February 2011, [Online; accessed 12-December-2011].

[8] T. S. Foundation, "http://www.shadowserver.org/wiki/," Online, may,2012.

[9] D. Dagon, "Botnet Detection and Response-The network is the infection," *Cooperative Association for Internet Data Analysis DNS-OARC Workshop, July*, vol. 25, 2005.

[10] T. Micro, "Taxonomy of botnet threats," *Micro*, pp. 1–15, November 2006. [Online]. Available: http://tinyurl.com/c7mlsjo

[11] D. Dagon, G. Gu, C. P. Lee, and W. Lee, "A taxonomy of botnet structures," *Twenty-Third Annual Computer Security Applications Conference ACSAC 2007*, vol. 36, pp. 325–339, 2007. [Online]. Available: http://tinyurl.com/8kxuknw

[12] J. Nazario, "Bot and botnet taxonomy," http://tinyurl.com/6bcthj, 2008, [Online; accessed 15-December-2011].

[13] G. Ollmann, "Botnet Communication Topologies," 2009. [Online]. Available: http://tinyurl.com/8kosjv3

[14] D. Plohmann, E. Gerhards-Padilla, and F. Leder, "Botnets: Detection, measurement, disinfection & defence," European Network and Information Security Agency, Tech. Rep., 2011.

[15] FireEye, "Next generation threat protection- fireeye inc." http://www.fireeye.com/, [Online; accessed 12-December-2011].

[16] Damballa, "Damballa::homepage," http://www.damballa.com/, [Online; accessed 12-December-2011].

[17] BotHunter, "Bothunter: A network-based botnet diagnostic system," http://www.bothunter.net/, [Online; accessed 12-December-2011].

[18] D. Dittrich, "The dos project's trinoo distributed denial of service attack tool," http://staff.washington.edu/dittrich/misc/trinoo.analysis, October 1999, [Online; accessed 25-August-2012].

[19] I. Arce, E. Levy, and E. Levy, "An analysis of the slapper worm," *IEEE Security & Privacy*, vol. 1, pp. 82–87, 2003.

[20] P. Wang, 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*. Berkeley, CA, USA: USENIX Association, 2007, pp. 2–2. [Online]. Available: http://dl.acm.org/citation.cfm?id=1323128.1323130

[21] J. Stewart, "Phatbot trojan analysis," http://tinyurl.com/9srw4gh, 2004, [Online; accessed 15-December-2011].

[22] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freilling, "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 Exploitsand Emergent Threats (LEET*, Berkeley, CA, USA, 2008.

[23] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, "Analysis of the storm and nugache trojans: P2P is here," in *USENIX ;login*, vol. 32, no. 6, 2007.

[24] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," *ACM USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet SRUTI*, vol. 7, pp. 39–44, 2005. [Online]. Available: http://tinyurl.com/8h94o9u

[25] E. Karamatli, "Modern botnets: A survey and future directions," http://docs.karamatli.com/paper/karamatli-modern_botnets.pdf, Bogazici University,Turkey, 2011.

[26] G. Ollmann, "Serial variant evasion tactics," 2009. [Online]. Available: http://tinyurl.com/93jnurm

[27] A. Cole, M. Mellor, and D. Noyes, "Botnets: The rise of the machines," in *Proceedings on the 6th Annual Security Conference*, 2007.

[28] K. Bong and J. Brozyck, "Managing large botnets," http://www.sans.edu/student-files/projects/200704_001.doc, 2007, [Online; accessed 15-December-2011].

[29] M. Bailey, E. Cooke, F. Jahanian, Y. Xu, and M. Karir, "A survey of botnet technology and defenses," in *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 299–304. [Online]. Available: http://dl.acm.org/citation.cfm?id=1524292.1524347

[30] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," *2009 Third International Conference on Emerging Security Information Systems and Technologies*, pp. 268–273, 2009. [Online]. Available: http://tinyurl.com/9njpehq

[31] H. R. Zeidanloo, M. J. Z. shooshtari, M. . Safari, P. V. Amoli, and M. Zamani, "A taxonomy of botnet detection techniques," *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, pp. 158–162, 2010. [Online]. Available: http://tinyurl.com/9ttpjwm

[32] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," Chalmers Univ., Tech. Rep. 99-15, Mar. 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.6603

[33] E. Stinson and 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*. Berkeley, CA, USA: USENIX Association, 2008, pp. 5:1–5:9. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496702.1496707

[34] N. Mody, M. O'Reirdan, S. Masiello, and J. Zebek, "Common best practices for mitigating large scale bot infections in residential networks," *MAAWG*, july 2009.

[35] A.-P. E. C. AEC, "Guide on Policy and Technical Approaches against Botnet," http://tinyurl.com/9b68qmj, Dec 2008, [Online; accessed 10-December-2011].

[36] F. Leder, T. Werner, and P. Martini, "Proactive botnet countermeasures an offensive approach," in *Cooperative Cyber Defence Centre of Excellence*, Tallinn, Estonia, March 2009.

[37] S. Stankovic and D. Simic, "Defense strategies against modern botnets," *International Journal of Computer Science and Information Security*, vol. 2, no. 1, 2009.

[38] J. Goebel and T. Holz, "Rishi: Identify bot contaminated hostsby irc nickname evaluation," in *USENIX Workshop on HotTopics in Understanding Botnets (HotBots'07)*, 2007.

[39] M. Roesch, "Snort - lightweight intrusion detection fornetworks," in *Proceedings of USENIX LISA'99*, 1999.

[40] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network trafficfor protocol- and structure-independent botnet detection," in *Usenix Security Symposium*, 2008.

[41] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," in *Network and Distributed System Security Symposium (NDSS)*, 2008.

[42] T. Wang and S.-Z. Yu, "Centralized botnet detection by traffic aggregation," in *IEEE International Symposium on Parallel and Distributed Processing with Applications*, 2009.

[43] T. Strayer, D. Lapsley, R. Walsh, and C. Livadas, *Botnet detection based on network behavior*, ser. Advances in Information Security. Springer, 2008, vol. 36, pp. 1–24.

[44] T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting botnets with tight command and control," in *Proceedings 2006 31st IEEE Conference on Local Computer Network*, 2006.

[45] T.-F. Yen and M. K. Reiter, "Traffic aggregation for malware detection," in *Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2008.

[46] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *ACM Internet Measurement Conference(IMC)*, 2006.

[47] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in *Usenix Security Symposium*, 2007.

[48] L. Liu, S. Chen, G. Yan, and Z. Zhang, "Bottracer:execution-based bot-like malware detection," in *11th Information Security Conference*, 2008.

[49] L. Zhuang, J. Dunagan, D. Simon, H. Wang, I. Osipkov, G. Hulten, and J. Tygar, "Characterizing botnets from email spam records," in *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[50] A. Ramachandran, N. Feamster, and D. Dagon, "Revealing botnet membership using dnsbl counter-intelligence," in *Conference on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006.

[51] R.Villamarin-Salomon and J. Brustoloni, "Identifying botnets using anomaly detection techniques applied to dns traffic," in *Proc. 5th IEEE Consumer Communications and Networking Conference (CCNC 2008)*, 2008.

[52] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in dns traffic," in *Proc. 7th IEEE International Conference on Computer and Information Technology (CIT 2007)*, 2007.

[53] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee, "Active botnet probing to identify obscure command and control channels," in

Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC), 2010.

[54] P. Porras, H. Sadi, V. Yegneswaran, P. Porras, H. Sadi, and V. Yegneswaran, "A multi-perspective analysis of the storm (peacomm) worm. available at: http://www.cyber-ta.org/pubs/stormworm/report," 2007.

[55] T. Economist, "A worm in the centrifuge," http://www.economist.com/node/17147818, 2010, [Online; accessed 20-December-2011].

[56] J. Baltazar, J. Costoya, and R. Flores, "The real face of koobface : The largest web 2 . 0 botnet explained," *Trend Micro Threat Research*, 2009. [Online]. Available: http://www.furcosystems.com/downloads/koobface.pdf

[57] T. Werner, "The inside story of the kelihos botnet takedown," http://tinyurl.com/3gzmtzd, threatpost:The Kaspersky Lab Security News Service, 2011, [Online; accessed 2-December-2011].

[58] P. Barford and V. Yegneswaran, "An Inside Look at Botnets," in *Malware Detection*, ser. Advances in Information Security, M. Christodorescu, S. Jha, D. Maughan, D. Song, and C. Wang, Eds. Boston, MA: Springer US, 2007, vol. 27, ch. 8, pp. 171–191. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-44599-1_8

[59] T. Cymru, "A taste of http botnets," http://tinyurl.com/9o5chx2, 2008, [Online; accessed 20-December-2011].

[60] A. Nappa, A. Fattori, M. Balduzzi, M. Dell'Amico, and L. Cavallaro, *Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, ch. Take a Deep Breath: A Stealthy, Resilient and Cost-Effective Botnet Using Skype.

[61] A. Berger and M. Hefeeda, "Exploiting sip for botnet communication," *2009 5th IEEE Workshop on Secure Network Protocols*, pp. 31–36, 2009. [Online]. Available: http://tinyurl.com/8n9rkex

[62] J. Nazario, "Twitter based botnet command and control," http://tinyurl.com/8ojo5wl, 2009, [Online; accessed 15-December-2011].

[63] E. J. Kartaltepe, J. A. Morales, S. Xu, and R. Sandhu, "Social network-based botnet command-and-control: emerging threats and countermeasures," in *Proceedings of the 8th international conference on Applied cryptography and network security*, ser. ACNS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 511–528. [Online]. Available: http://dl.acm.org/citation.cfm?id=1894302.1894342

[64] A. Lelli, "Trojan.whitewell: Whats your (bot) facebook status today?" http://tinyurl.com/yeb5rvb, 2009, [Online; accessed 15-December-2011].

[65] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Taking over the torpig botnet: My botnet is your botnet," http://www.cs.ucsb.edu/~seclab/projects/torpig/, 2009, [Online; accessed 15-December-2011].

[66] R. Westervelt, "Botnet masters turn to google, social networks to avoid detection," http://tinyurl.com/8ta9nly, 2009, [Online; accessed 15-December-2011].

[67] Damballa, "The command structure of the operation aurora botnet: History, patterns, and findings," Tech. Rep., 2010.

[68] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szyd-lowski, R. Kemmerer, C. Kruegel, and 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)*, Nov 2009.

[69] J. Pinto, "Distributed & Grid Computing," http://www.jimpinto.com/writings/grid.html, Automation.com, May 2003, [Online; accessed 15-December-2011].

[70] A. Cole, M. Mellor, and D. Noyes, "Botnets: The rise of the machines," in *Proceedings on the 6th Annual Security Conference*, 2007.

[71] N. Lewis, "Zeus botnet analysis: Past, present and future threats," http://tinyurl.com/d3vwcll, 2010, [Online; accessed 15-December-2011].

[72] A. Decker, D. Sancho, L. Kharouni, M. Goncharov, and R. McArdle, "Pushdo/cutwail botnet: A study of the pushdo/ cutwail botnet," TrendMicro Labs, Tech. Rep., 2009.

[73] R. Clarke and R. Knake, *Cyber War: The Next Threat to National Security and What to Do About It.* HarperCollins, 2010. [Online]. Available: http://books.google.com.pk/books?id=_oASQgAACAAJ

[74] I. Traynor, "Russia accused of unleashing cyberwar to disable estonia," http://www.guardian.co.uk/world/2007/may/17/topstories3.russia, May 2007, [Online; accessed 12-December-2011].

[75] B. Stone-Gross, G. S. T. Holz, and G. Vigna., "The underground economy of spam: A botmasters perspective of coordinating large-scale spam campaigns," in *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2011.

[76] E. Athanasopoulos, A. Makridakis, S. Antonatos, D. Antoniades, S. Ioannidis, K. Anagnostakis, and E. Markatos, "Antisocial networks: turning a social network into a botnet," in *Proceedings of the 11th Information Security Conference*, Taipei, Taiwan, 2008.

[77] CWSandBox, "Cwsandbox," http://mwanalysis.org/, [Online; accessed 15-December-2011].

[78] A. Mushtaq, "The dead giveaways of vm-aware malware," http://tinyurl.com/4ejusr2, Jan 2011, [Online; accessed 15-December-2011].

[79] F. Leder and T. Werner, "Know Your Enemy: Containing Conficker, To Tame a Malware," The Honeynet Project, http://honeynet.org, Tech. Rep., Apr. 2009.

[80] V. Tiu, "Information about worm:win32/conficker.d," http://tinyurl.com/8c84vhl, March 2009, [Online; accessed 15-December-2011].

[81] M. S. Mimoso, "Fbi takes down dns changer botnet," http://tinyurl.com/8nltjzf, Nov 2011, [Online; accessed 12-December-2011].

[82] Tor, "Tor: Anonymity online," https://www.torproject.org/, [Online; accessed 20-December-2011].

[83] D. Fisher, "Storm, nugache lead dangerous new botnet barrage," http://tinyurl.com/8folbmw, 2007, [Online; accessed 15-December-2011].

[84] US-Cert, "Malware tunneling in ipv6." http://www.us-cert.gov/reading_room/IPv6Malware-Tunneling.pdf, 2005, [Online; accessed 15-December-2011].

[85] P. Bright, "How Operation b107 decapitated the Rustock botnet," http://tinyurl.com/4bajdjx, 2011, [Online; accessed 10-December-2011].

[86] J. Nazario, "Twitter-based botnet command channel," http://tinyurl.com/78je7ej, 2009, [Online; accessed 12-December-2011].

[87] P. Wurzinger, L. Bilge, T. Holz, J. Gobel, C. Kruegel, and E. Kirda, "Automatically generating models for botnet detection," in *European Symposium on Research in Computer Security (ESORICS)*, 2009.

[88] S. Shin and G. Gu, "Conficker and beyond: A large-scale empirical study," in *Proceedings of Annual Computer Security Applications Conference (ACSAC)*, 2010.

[89] J. Kristoff, "Experiences with conficker c sinkhole operation and analysis," in *Proceedings of Australian Computer Emergency Response Team Conference)*, May 2009.

[90] M. Roesch, "Snort - lightweight intrusion detection fornetworks," in *Proceedings of USENIX LISA'99)*, 1999.

[91] L. Aniello, G. Lodi, and R. Baldoni, "Inter-domain stealthy port scan detection through complex event processing," in *Proceedings of the 13th European Workshop on Dependable Computing*, ser. EWDC '11. New York, NY, USA: ACM, 2011, pp. 67–72. [Online]. Available: http://doi.acm.org/10.1145/1978582.1978597

[92] L. Aniello, G. A. Di Luna, G. Lodi, and R. Baldoni, "A collaborative event processing system for protection of critical infrastructures from cyber attacks," in *Proceedings of the 30th international conference on Computer safety, reliability, and security*, ser. SAFECOMP'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 310–323. [Online]. Available: http://dl.acm.org/citation.cfm?id=2041619.2041651

[93] F. Doelitzscher, C. Reich, M. Knahl, and N. Clarke, "Incident detection for cloud environments," in *Proceedings of the Third International Conference on Emerging Network Intelligence (EMERGING 2011)*, Nov 2011.

[94] E. Stinson and J. C. Mitchell, "Characterizing bots' remote control behavior," in *International Conference on Detectionof Intrusions & Malware, and Vulnerability Assessment (DIMVA)*, 2007.

[95] W. Lu, M. Tavallaee, and 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*, ser. ASIACCS '09. New York, NY, USA: ACM, 2009, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/1533057.1533062

[96] J. Binkley and S. Singh, "An algorithm for anomaly-based botnet detection," in *Usenix Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, 2006.

[97] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, 2007.

[98] T.-F. Yen and M. K. Reiter, "Are your hosts trading or plotting? telling p2p file-sharing and bots apart," in *International Conference on Distributed Computing Systems*, 2010.

[99] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi, "Fluxor : Detecting and monitoring fast-flux service networks," *Detection of Intrusions and Malware and Vulnerability Assessment*, pp. 186–206, 2008. [Online]. Available: http://www.springerlink.com/index/R710H553172801X6.pdf

[100] T. Holz, C. Gorecki, K. Rieck, and F. Freiling, "Detection and Mitigation of Fast-Flux Service Networks," in *Proceedings of NDSS 2008*, San Diego, CA, USA, Feb. 2008. [Online]. Available: http://tinyurl.com/9q64vk5

[101] J. Caballero, P. Poosankam, C. Kreibich, and D. Song, "Dispatcher: Enabling active botnet infiltration using automatic protocol reverse-engineering," in *ACM Conference on Computer and Communications Security*, Nov 2009.

[102] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clusteringof http-based malware and signature generation using malicious network traces," in *USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, 2010.

[103] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 2008. [Online]. Available: http://tinyurl.com/9hpoa3d

[104] C. Livadas, R. Walsh, D. Lapsley, and W. Strayer, "Using machine learning technliques to identifybotnet traffic," in *Proceedings of the 2nd IEEELCN Workshop on Network Security*, Nov 2006.

[105] G. Jacob, R. Hund, T. Holz, and C. Kruegel, "Jackstraws: Picking command and control connections from bot traffic," in *USENIX Security Symposium*, 2011.

[106] S. Savage, D. Wetherall, A. R. Karlin, and T. E. Anderson, "Practical network support for ip traceback," *Computer Communication Review*, vol. 30, pp. 295–306, 2000.

[107] D. X. Song and A. Perrig, "Advanced and authenticated marking schemes for ip traceback," in *IEEE INFOCOM*, 2001, pp. 878–886.

[108] S. M. Bellovin, M. Leech, and T. Taylor, "ICMP traceback messages," Obsolete Internet draft, February 2003. [Online]. Available: https://www.cs.columbia.edu/~smb/papers/draft-ietf-itrace-04.txt

[109] A. Mankin, D. Massey, C. long Wu, S. F. Wu, and L. Zhang, "On design and evaluation of "intention-driven" icmp traceback," in *International Conference on Computer Communications and Networks*, 2001.

[110] A. Belenky and N. Ansari, "Ip traceback with deterministic packet marking," 2003.

[111] A. Yaar, A. Perrig, and D. Song, "Pi: A path identification mechanism to defend against ddos attacks," in *IEEE Symposium on Security and Privacy*, 2003, pp. 93–107.

[112] D. Ramsbrock, X. Wang, and X. Jiang, "A first step towards live botmaster traceback," in *Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, ser. RAID '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 59–77. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-87403-4_4

[113] A. C. Snoeren, "Hash-based ip traceback," *Computer Communication Review*, vol. 31, pp. 3–14, 2001.

[114] S. Staniford-Chen and L. T. Heberlein, "Holding intruders accountable on the internet," in *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, ser. SP '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 39–. [Online]. Available: http://dl.acm.org/citation.cfm?id=882491.884246

[115] Y. Zhang and V. Paxson, "Detecting stepping stones," in *USENIX Security Symposium*, 2000.

[116] X. Wang, D. S. Reeves, and S. F. Wu, "Inter-packet delay based correlation for tracing encrypted connections through stepping stones," in *European Symposium on Research in Computer Security (ESORICS)*, 2002, pp. 244–263.

[117] D. L. Donoho, A. G. Flesia, U. Shankar, V. P. J. Coit, , S. Staniford, J. Coit, and S. Staniford, "Multiscale stepping-stone detection: Detecting pairs of jittered interactive streams by exploiting maximum tolerable delay," in *Proc. of The 5th International Symposium on Recent Advances in Intrusion Detection (RAID)*. Springer, 2002, pp. 17–35.

[118] A. Blum, D. Song, and S. Venkataraman, "Detection of interactive stepping stones: Algorithms and confidence bounds," in *Conference of Recent Advance in Intrusion Detection (RAID), (Sophia Antipolis, French Riviera*. Springer, 2004, pp. 258–277.

[119] L. Zhang, A. G. Persaud, A. Johnson, and Y. Guan, "Detection of stepping stone attack under delay and chaff perturbations," in *International Performance, Computing, and Communications Conference*, 2006.

[120] X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays," in *Proceedings of the 10th ACM conference on Computer and communications security*, ser. CCS '03. New York, NY, USA: ACM, 2003, pp. 20–29. [Online]. Available: http://doi.acm.org/10.1145/948109.948115

[121] W. T. Strayer, C. E. Jones, I. Castineyra, J. B. Levin, and R. R. Hain, "An integrated architecture for attack attribution," *Tech. Rep. BBN REPORT-8384*, Dec 2003.

[122] K. Yoda and H. Etoh, "Finding a connection chain for tracing intruders," in *Proceedings of the 6th European Symposium on Research in Computer Security (ESORICS)*, 2000, pp. 191–205.

[123] X. Wang, X. Wang, D. S. Reeves, D. S. Reeves, S. F. Wu, S. F. Wu, J. Yuill, and J. Yuill, "Sleepy watermark tracing: An active network-based intrusion response framework," in *Proc. of the 16th International Information Security Conference*, 2001, pp. 369–384.

[124] J. B. Grizzard and T. Johns, "Peer-to-peer botnets: Overview and case study," in *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots07)*, 2007.

[125] M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "My botnetis bigger than yours (maybe, better than yours) : Why size estimates remain challenging," in *USENIX Workshop on Hot Topics in Understanding Botnet*, 2007.

[126] J. Zhang, X. Luo, R. Perdisci, G. Gu, W. Lee, and N. Feamster, "Boosting the scalability of botnet detection using adaptive traffic sampling," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '11. New York, NY, USA: ACM, 2011, pp. 124–134. [Online]. Available: http://doi.acm.org/10.1145/1966913.1966930

[127] T. Holz, M. Engelberth, and F. Freiling, "Learning more about the underground economy: A case-study of keylog-gers and dropzones," in *European Symposium on Research in Computer Security (ESORICS)*, 2009.

[128] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage, "Spamalytics: An empirical analysis of spam marketing conversion," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, Alexandria, Virginia, USA, Oct 2008, pp. 3–14.

[129] R. Ford and S. Gordon, "Cent, five cent, ten cent, dollar: hitting botnets where it really hurts," in *Proceedings of the 2006 Workshop on New Security Paradigms (NSPW'06)*. New York, NY, USA: ACM, 2007, p. 310.

[130] URIBL, "uribl-website," http://www.uribl.com/, [Online; accessed 12-December-2011].

[131] T. A. Meyer and B. Whateley, "SpamBayes: Effective open-source, Bayesian based, email classification system," in *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.

[132] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G. Voelker, V. Paxson, N. Weaver, and S. Savage, " Botnet Judo: Fighting Spam with Itself ," in *Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, March 2010.

[133] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the dos and ddos problems," *ACM Comput. Surv.*, vol. 39, April 2007.

[134] B. Furfie, "Laws must change to combat botnets Kaspersky," http://tinyurl.com/9tpdyj5, Feb 2011, [Online; accessed 10-December-2011].

[135] J. Leyden, "Botnet-harbouring survey fails to accounts for sinkholes," http://www.theregister.co.uk/2010/10/26/botnet_hosting_isps/, Oct 2010, [Online; accessed 10-December-2011].

[136] G. L. Orgill, G. W. Romney, M. G. Bailey, and P. M. Orgill, "The urgency for effective user privacy-education to counter social engineering attacks on secure computer systems," in *Proceedings of the 5th conference on Information technology education*, ser. CITC5 '04. New York, NY, USA: ACM, 2004, pp. 177–181.

[137] T. Ormerod, L. Wang, M. Debbabi, A. Youssef, H. Binsalleeh, A. Boukhtouta, and P. Sinha, "Defaming botnet toolkits: A bottom-up approach to mitigating the threat," in *Proceedings of the 4th International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, 2010.

[138] SecuriTeam, "Sasser worm remote ftpd buffer overflow exploit code (port 5554)," http://www.securiteam.com/exploits/5AP0J0ACUM.html, 2004, [Online; accessed 15-December-2011].

[139] C. Y. Cho and J. Caballero, "Botnet infiltration: Finding bugs in botnet command and control," http://www.eecs.berkeley.edu/~chiayuan/cs261/cs261_cho.pdf, [Online; accessed 15-December-2011].

[140] J. R. Douceur, "The sybil attack," in *Proceedings of the International workshop on Peer-To-Peer Systems (IPTPS)*, March 2002.

[141] A. Singh, T.-W. J. Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *IEEE International Conference on Computer Communications (Infocom)*, 2006.

[142] V. Paxson, "Bro: a system for detecting network intruders in real-time," in *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*, ser. SSYM'98. Berkeley, CA, USA: USENIX Association, 1998, pp. 3–3. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267549.1267552

[143] Bro, "http://bro-ids.org/," Online, may,2012.

[144] C. Kreibich and R. Sommer, "Policy-controlled event management for distributed intrusion detection," in *Proceedings of the Fourth International Workshop on Distributed Event-Based Systems (DEBS) (ICDCSW'05) - Volume 04*, ser. ICDCSW '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 385–391. [Online]. Available: http://dx.doi.org/10.1109/ICDCSW.2005.112

[145] DShield, "http://www.dshield.org/," Online, may,2012.

[146] TCATS, "http://tcats.stop-spam.org/tcats/sibl/," Online, may,2012.

[147] C.I.Army, "http://ciarmy.com/," Online, may,2012.

[148] BLADE, "http://www.blade-defender.org/eval-lab/," Online, may,2012.

[149] TeamCymru, "http://www.team-cymru.org/services/mhr/," Online, may,2012.

[150] EmergingThreats, "http://www.emergingthreats.net/," Online, may,2012.

[151] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szyd-lowski, R. Kemmerer, C. Kruegel, and 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)*, Nov 2009.

[152] F. Leder and T. Werner, "Know Your Enemy: Containing Conficker, To Tame a Malware," The Honeynet Project, http://honeynet.org, Tech. Rep., Apr. 2009.

[153] N. R. Ramay, S. Khattak, A. A. Syed, and S. A. Khayam, "Bottleneck: A generalized, flexible and extensible framework for botnet defense," in *IEEE Symposium on Security and Privacy, 2012*, May 2012, poster Paper.

[154] D. Goodin, "Waledac botnet 'decimated' by takedown," http://tinyurl.com/7apnn9b, March 2010, [Online; accessed 12-December-2011].

[155] ITN-News, "'slain' kelihos botnet still spams from beyond the grave," http://tinyurl.com/9esmb3e, Feb 2012, [Online; accessed 1-February-2012].

[156] R. McMillan, "China cleans up spam problem," http://tinyurl.com/clnedlt, February 2011, [Online; accessed 12-December-2011].