Chapter 8
Analysis of Peer-to-Peer Botnet Attacks and Defenses

Ping Wang, Lei Wu, Baber Aslam and Cliff C. Zou

Abstract A “botnet” is a network of computers that are compromised and controlled by an attacker (botmaster). Botnets are one of the most serious threats to today’s Internet. Most current botnets have centralized command and control (C&C) architecture. However, peer-to-peer (P2P) structured botnets have gradually emerged as a new advanced form of botnets. Due to the distributive nature of P2P networks, P2P botnets are more resilient to defense countermeasures. In this chapter, first we systematically study P2P botnets along multiple dimensions: bot candidate selection, network construction, C&C communication mechanisms/protocols, and mitigation approaches. Then we provide mathematical analysis of two P2P botnet *elimination* approaches – index poisoning defense and Sybil defense, and one P2P botnet monitoring technique – passive monitoring based on infiltrated honeypots or captured bots. Simulation experiments show that our mathematical analysis is accurate.

Ping Wang
Symantec Corporation, Lake Mary, Florida 32746, USA e-mail: `jenpwang@gmail.com`

Lei Wu
Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA e-mail: `lwu4@ncsu.edu`

Baber Aslam
National University of Sciences & Technology, Islamabad, Pakistan e-mail: `baber-mcs@nust.edu.pk`

Cliff C. Zou
Department of Electrical Engineering & Computer Science, University of Central Florida, Orlando, Fl 32816, USA e-mail: `czou@cs.ucf.edu`

8.1 Introduction

8.1.1 Botnets

A “botnet” is a network of compromised computers (bots) running malicious software, usually installed via all kinds of attacking techniques such as Trojan horses, worms and viruses. These zombie computers are remotely controlled by an attacker (botmaster). Botnets with a large number of computers have enormous cumulative bandwidth and computing capability. They are exploited by botmasters for initiating various malicious activities, such as email spam, distributed denial-of-service attacks, password cracking and key logging. Botnets have become one of the most significant threats to the Internet.

Today, centralized botnets are still widely used. In a centralized botnet, bots are connected to several servers (called command and control servers) to obtain commands. This architecture is easy to construct and efficient in distributing a botmaster’s commands; however, it has a weak link - the command and control (C&C) servers. Shutting down those servers would cause all bots in a botnet to lose contact with their botmaster. In addition, defenders can easily monitor the botnet by creating a decoy to join a specified C&C channel.

In the last few years, peer-to-peer (P2P) botnets, such as Trojan.Peacomm botnet, Storm botnet and its newly improved version Waledac botnet, have emerged as attackers gradually realize the limitation of traditional centralized botnets. “Peer-to-peer botnets” are defined as botnets that rely on peer-to-peer communication mechanisms to facilitate the command and control by their botmasters. There are different ways for a P2P botnet to utilize P2P communication for its command and control. For example, a P2P botnet could use a P2P network (either an existing P2P network, or a unique P2P network formed by its bot members) to directly disseminate its botmaster’s commands to all bot members, or it could use a P2P network to disseminate the IP addresses of C&C servers to bot members (like what Storm botnet [53] does, which utilizes an existing P2P protocol to form a hierarchical multi-tier command and control architecture). Due to the fundamental distributive nature of P2P networks, P2P botnets are robust against removal of bots and C&C servers, and have shown great advantages over traditional centralized botnets. As the next generation of botnets, they are more robust and difficult for security community to defend against.

Researchers have started to pay attention to P2P botnet threat in recent years. Trojan.Peacomm botnet, Stormnet and Waledac botnet have been dissected in details in literature [18, 19, 27, 41, 43, 54]. Andriesse et al. [5] reverse engineered P2P botnet Zeus. However, in order to effectively fight against this new form of botnets, it is not enough to simply enumerate and analyze every individual P2P botnet we have encountered in the wild. Instead, we need to study P2P botnets in a more systematic way. Therefore in this book chapter we try to explore the nature of various kinds of P2P botnets, analyzing their similarities and differences, and discussing their weaknesses and possible defenses.

8.1.2 Botnet Countermeasures

From our understanding, botnet countermeasures can be classified into three categories: detection, monitoring, and mitigation.

Detection refers to detecting and identifying a botnet in a network. It includes identifying bot members by various ways, such as signature-based malware detection, network flow monitoring, honeypot infiltration, etc; it also includes discovering C&C channels, such as locating the Internet Relay Chat (IRC) servers of an IRC-based botnet.

Monitoring refers to infiltrating a discovered botnet and monitoring its activities. It can help people better understand a botmaster's motivation, a botnet's behavior and design, etc. There are two types of monitoring: active – actively contact bot members to explore their behaviors, and passive – set up traps and wait for bots to contact, such as dark address space monitoring.

Mitigation refers to eliminating a discovered botnet, by either curing all/most bots in a botnet or disabling its botmaster's capability in command and control. Upon botnet detection and monitoring, mitigation is the ultimate goal for botnet defense. Because most botnets are large and contain bots located in areas that are beyond our control, in most cases curing all/most bots in a detected botnet is not feasible. Therefore, botnet mitigation usually means isolating bots by disrupting a botnet's C&C channels. This idea can be easily applied to centralized botnets, because in a centralized botnet, the C&C traffic will go through one or a few clearly-defined central servers. As long as we are able to identify the centralized servers of a botnet and stop botnet-related network activities to/from these servers, we can stop the communication between a botmaster and his/her bots, resulting in disabling the botnet. On the other hand, as a P2P botnet utilizes a P2P network to pass important messages across the entire botnet, it is generally much harder to disrupt the information distribution.

There are many research works focusing on botnet detection and monitoring (discussed in Section 8.4), but few research works studying botnet mitigation. For P2P botnets, researchers have presented two mitigation techniques – index poisoning defense and Sybil defense [24, 27]. The original ideas behind these two techniques were first introduced to "sabotage"[1] legitimate P2P networks, and now defenders leverage the same ideas to fight against P2P botnets. Empirical studies on index poisoning defense and Sybil defense have been presented in [24, 27], which have shown that they can successfully disrupt the communication of P2P botnets.

In our preliminary study [65], we presented the systematic study of P2P botnets, and provided the mathematical analysis of index poisoning and Sybil defense, but without much discussion and with no simulation evaluation. In this chapter, we study index poisoning defense and Sybil defense techniques further by providing new simulation study and detailed discussions. In addition, we present our new investigation on passive monitoring technique, providing both the analytical study of the capability of a monitoring node in a P2P botnet, and the simulation evaluation. To the best of our knowledge, we are the first to provide mathematical analysis on the performance of index poisoning defense, Sybil defense and passive monitoring, not only for P2P botnets, but also for their corresponding attacks targeting general P2P systems. We also confirm the accuracy of our analysis with simulation experiments. We hope to shed light on P2P botnets, and help researchers and security professionals be well prepared and develop effective defenses against them.

8.1.3 Contributions

The major contributions of this chapter are summarized as follows.

- We systematically study P2P botnets along multiple dimensions: infection vectors, bot candidate selection, bootstrap procedure, network structure, C&C mechanisms and communication protocols.

[1] Index poisoning was introduced by media companies to prevent illegal distribution of copyrighted content in P2P networks [36], while Sybil attack was to subvert a reputation system in P2P networks [17].

- We mathematically analyze the performance of two popular P2P botnet mitigation techniques: index poisoning defense and Sybil defense.
- We also carefully study passive monitoring of P2P botnets: based on the mathematical analysis of the capability of a monitoring node in a P2P botnet, we are able to provide a lower bound for the number of bots that an infiltrated node can monitor.
- We develop a Kademlia-based P2P botnet simulator. All the analytical results presented in this chapter have been shown to be accurate by simulation-based experiments using this simulator.
- From attackers' perspective, we propose a novel and realistic technique that might be deployed by them to counterattack the index poisoning defense. This method guarantees that command related indices published in a P2P botnet can be generated by and only by botmasters, not by ordinary bots.
- We obtain one counter-intuitive finding: if the index poisoning defense is valid (when a botnet adopts existing P2P protocols and relies on indices to disseminate commands), P2P botnets are equally easy (or hard) to defend compared to traditional centralized botnets.
- The mathematical analysis presented in this chapter is also suitable for modeling index poisoning attack and Sybil attack in legitimate P2P networks, and hence, contribute to the security research for legitimate P2P systems as well.

8.1.4 Chapter Organization

The remainder of the chapter is organized as follows. In Section 8.2, we study the life cycle of P2P botnets, which is composed of three stages. Upon our understanding of P2P botnets, a number of countermeasures are presented in Section 8.3, and special attentions are given to two mitigation techniques – index poisoning defense (Section 8.3.2) and Sybil defense (Section 8.3.3), and one passive monitoring technique (Section 8.3.4). We review the related work in Section 8.4 and conclude in Section 8.5.

8.2 A Systematic Study on P2P Botnets

The life cycle of botnets is composed of three stages. Stage one - recruiting members, a botmaster needs to compromise many computers in the Internet, so that he/she can control them remotely. Stage two - forming the botnet, bots need to find a way to connect to each other and form a botnet. Stage three - standing by for instructions, after the botnet is built up, all bots are ready to receive communication from their botmaster for further instructions, such as launching an attack or performing an update. In this section, we will discuss each stage in detail.

8.2.1 Stage One: Recruiting Bot Members

P2P networks are gaining popularity in distributed applications, such as file-sharing, web caching, network storage [9]. In these content-trading P2P networks, without a centralized authority it is not easy to guarantee that the contents exchanged are not malicious. For this reason, these networks become the ideal venue for malicious software to spread. It is straightforward for attackers to target vulnerable hosts in existing P2P networks as bot candidates and build their zombie army. Many P2P malware have been reported, such as Gnuman [1], VBS.Gnutella [1] and SdDrop [4]. They can be used to compromise a host and make it become a bot.

However, in this way, the scale of a botnet will be limited by the size of an existing P2P network, and the network will be the only propagation media. On the contrary, P2P botnets we have witnessed in recent years [19, 27, 57] do not confine themselves to existing P2P networks. They have shown that it is more flexible and practical if bot members are recruited from the entire Internet through all possible spread mediums like emails, instant messages and file exchanging.

8.2.2 Stage Two: Forming the Botnet

Upon infection, the next important thing is to let newly compromised computers join the botnet network and connect to other bots. Otherwise, they are just isolated individual computers without much use for botmasters.

Now for the convenience of further discussion, we first introduce three terms: "*parasite*", "*leeching*" and "*bot-only*" P2P botnets. Each of them represents a class

Table 8.1: Comparison among three types of P2P botnets

Features	**Parasite**	**Leeching**	**Bot-only**
Infection vectors	P2P malware	Any kind of malware	Any kind of malware
Bot candidates	Vulnerable hosts P2P networks	Vulnerable hosts in the Internet	Vulnerable hosts in the Internet
Bootstrap procedure	None	Required	Optional
Members in the network	Legitimate peers & bots	Legitimate peers & bots	Only bots
Communication protocols	Existing P2P protocols	Existing P2P protocols	Self-designed or existing P2P protocols
C&C styles	Pull or push	Pull or push	Pull or push

of P2P botnets. In a parasite P2P botnet, all bots are selected from vulnerable hosts within an existing P2P network. The botnet uses this available P2P network for command and control. A leeching P2P botnet refers to one whose members join an existing P2P network and depend on this P2P network for C&C communication, but the bots could be vulnerable hosts that were either inside or outside of the existing P2P network. For example, the early version of Storm botnet [27] belongs to this class of botnet. A bot-only P2P botnet builds its own P2P network, in which all the members are bots, such as Stormnet [27] and Nugache [57].

If all bots are selected from an existing P2P network, it is not necessary to perform any further action to form the botnet, because bots can find and communicate with each other by simply using current P2P protocol. In other words, for a parasite P2P botnet, up to this point, the botnet construction is done and the botnet is ready to be operated by its botmaster.

However, if a random host on the Internet is compromised, it has to know how to find and join the botnet, which is the case for leeching botnets and bot-only botnets. As we know current P2P file-sharing networks provide the following two general ways for new peers to join a network:

1. An initial peer list is hard-coded in each P2P client. When a new peer is up, it will try to contact peers in that initial list to update its neighboring peer information.
2. There is a shared web cache, such as Gnutella web cache [15], stored at some place on the Internet, and the location of the cache is put in the client code. Thus a new peer can refresh its neighboring peer list by going to the web cache and fetching the latest updates.

This initial procedure of finding and joining a P2P network is usually called "bootstrap" procedure. It can be directly adopted for P2P botnet construction. Either a predetermined list of peers or the locations of predetermined web caches need to be hard-coded in the bot code. Then a newly infected host knows which peer to contact or at least where to find candidates of neighboring peers it will contact later.

For instance, Trojan.Peacomm [19] is a piece of malware to create a P2P botnet which uses the Overnet P2P protocol for controlling the bots. A list of Overnet nodes that are likely to be online is hard-coded into the bot's installation binary. When a victim is compromised and runs a Trojan.Peacomm, it will try to contact peers in this predefined list to bootstrap onto the Overnet network. Another P2P botnet, Stormnet [27], uses a similar bootstrap mechanism: the information about other peers with which a new bot member communicates after the installation phase, is encoded in a configuration file that is also stored on the victim machine by Storm worm binary.

8.2.3 Stage Three: Standing by for Instructions

Once a botnet is built up, all bots in the botnet are standing by for instructions from their botmaster to perform illicit activities or updates. Therefore C&C mechanism is very important and is the major part of a botnet design. It directly determines the communication topology of a botnet, and hence affects the robustness of a botnet against network/computer failures, or security monitoring and mitigation.

The C&C mechanisms can be categorized as either *pull* or *push* mechanism. Pull mechanism, i.e., "command publishing/subscribing", refers to the manner that bots retrieve commands actively from a place where botmasters publish commands. On the contrary, push mechanism, i.e., "command forwarding", means bots passively wait for commands to reach them and then forward received commands to others.

For centralized botnets, pull mechanism is commonly used. Take HTTP-based botnets as an example, a botmaster publishes commands on a web page, and bots periodically visit this web page via HTTP to check for any command updates. In the following, we will discuss how pull and push C&C mechanisms can be applied in P2P botnets.

8.2.3.1 Leveraging Existing P2P Protocols

As we discussed above, both parasite and leeching P2P botnets depend on existing P2P networks. Thus it is natural to leverage the existing P2P protocols used by the host P2P networks for C&C communication. Besides, these protocols have been tested in P2P file-sharing applications for a long time, so they tend to be less error-prone than newly designed ones, and have nice properties to improve performance of P2P systems and mitigate network problems, such as link failure or churn ("churn" refers to network dynamics caused by nodes' joining and leaving activities). The following discussion is based on parasite and leeching P2P botnets, but bot-only botnet can adopt these protocols as well.

In P2P file-sharing systems, file index, which is used by peers to locate the desired content, may be centralized (e.g., Napster), distributed over a fraction of the file-sharing nodes (e.g., Gnutella), or distributed over all or a large fraction of the nodes (e.g., Overnet). A peer can send out query message for the file it is searching for, and the message will be passed around according to the routing algorithm implemented in the system. The search will terminate when query hits are returned or the query message expires.

Botmasters can easily adopt the above procedure to disseminate commands in pull-style. They can insert records associated with some predefined file titles or hash values into the index, but rather than putting the content location information, botnet commands are attached. In order to get commands issued by botmasters, bots periodically initiate queries for those files or hashes, and nodes who preserve the corresponding records will return query hits with commands encoded. In other words, bots subscribe the content (i.e., commands) published by their botmaster.

The early version of Storm botnet [27] is a good example to show how a P2P botnet could leverage an existing P2P network or implement an existing P2P protocol for the pull-style command and control, although it uses the Overnet P2P network to pass the locations of its C&C servers instead of botmaster's commands. In Storm botnet, every day there are 32 keys queried by bots to retrieve important information. These 32 keys are calculated by a built-in algorithm, which takes the current date and a random number from [0-31] as input. Therefore, when issuing a command, the botmaster needs to publish it under 32 different keys. Trojan.Peacomm botnet [19] employs the similar design.

Compared to pull mechanism, implementation of push mechanism on existing P2P protocols is more complicated. There are two major design issues:

- Which peers should a bot forward a command to?
- How to forward commands: using in-band (normal P2P traffic) or out-of-band messages (non-P2P traffic)?

To address the first issue, the simplest way is to let a bot use its current known neighboring peers as targets. But the problem of this approach is that command distribution may be slow or sometimes disrupted, because 1) some bots have a small number of neighbors, or 2) some peers in a bot's neighbor list are not bot members in the case of parasite or leeching P2P botnets. One solution to this problem is that letting bots claim they have predefined popular files available, and forwarding commands to peers appearing in the search results for those files. Thus the chance of commands hitting an actual bot is increased. These predefined popular files behave as the watchwords for the botnet, but could give defenders a clue to identify bots.

For the second issue, whether using in-band or out-of-band message to forward a command depends on what the peers in the target list are. If a bot targets its neighboring peers, in-band message is a good choice. A bot could encode a command in a query message, which can only be interpreted by bots, send it to all its neighbors, and rely on them to continue passing on the command in the botnet. This scheme is easy to implement and hard for defenders to detect, because there is no difference between command forwarding traffic and normal P2P traffic. On the other hand, if the target list is generated in other ways, like using peers in returned search results discussed above, bots have to contact those peers using out-of-band message. Obviously out-of-band traffic are easier to detect, and hence, can disclose the identities of bots who initiate such traffic.

The discussion above mainly focused on unstructured P2P networks, where query messages are flooded to the network. In structured P2P networks (e.g., Overnet), a query message is routed to the nodes whose node IDs are closer to the queried key, which means queries for the same key are always forwarded by the same set of nodes. Therefore, to let more bots receive a command, the command should be associated with different keys, such that it can be sent to different parts of the network.

8.2.3.2 Design a New P2P Communication Protocol

It is convenient to adopt existing P2P protocols for P2P botnet C&C communication, however, the inherited drawbacks of existing P2P protocols may limit botnet design and performance. A botnet can be more flexible if it uses a new protocol designed by its botmaster.

The advanced hybrid P2P botnet [63] and the super botnet [61] are two newly designed P2P botnets, whose C&C communication are not dependent on existing P2P protocols. Both of them implement push and pull C&C mechanisms. In a hybrid P2P botnet, when a bot receives a command, it forwards the command to all the peers in its peer list (push), and those who cannot accept connections from others periodically contact other bots in their peer lists and try to retrieve new commands (pull). A super botnet is composed of a number of small centralized botnets. Commands are pushed from one small botnet to another, and within a small centralized botnet, bots pull the commands from their C&C servers. Furthermore, the hybrid P2P botnet is able to effectively avoid bootstrap procedure (required by most of the existing P2P protocols) by 1) passing a peer list from one bot to a host that is infected by this bot, and 2) exchanging peer lists when two bots communicate.

The drawback of designing a new protocol for P2P botnet communication is that the new protocol has never been tested before. When a botnet using this protocol is deployed, the network may not be as stable and robust as expected due to complex network conditions and defenses.

8.2.4 Discussion

Several features can be extracted to represent a P2P botnet during its life cycle: infection vectors, bot candidates, bootstrap procedure, members in the network, communication protocols and C&C styles. Parasite, leeching and bot-only P2P botnets share common features but differ in others, which is summarized in Table 8.1. It is shown that parasite P2P botnets are less flexible but require no bootstrap procedure. This is an advantage of the parasite botnets over the other two classes. Botnets are most vulnerable during bootstrap stage and the propagation could be stopped if bootstrap information is compromised by defenders. Leeching and bot-only P2P botnets are similar, but the former ones are more stealthy. This is because leeching botnets resides in existing P2P networks, resulting in bot members being mixed with legitimate nodes and hard to be detected.

8.3 Countermeasures

As discussed in Section 8.1.2, we believe P2P botnet defense study should be composed with three areas of research: detection, monitoring, and mitigation. Botnet detection has been widely studied by other researchers such as in [10, 28], and hence, we will not discuss it in this book chapter. Instead, we will exploit and analyze possible solutions for P2P botnet monitoring and mitigation.

In research on P2P file-sharing networks, people have long noticed that most P2P protocols are susceptible to index poisoning attack [36] and Sybil attack [17]. Since existing P2P botnets, such as Trojan.Peacomm and Stormnet, directly utilize existing P2P protocols, security defenders could rely on the same principle to conduct index poisoning defense and Sybil defense. In [16, 19, 27], researchers have pointed out that index poisoning defense and Sybil defense can be used to fight against P2P botnets. However, none of them have presented detailed analysis of the performance of these two mitigation approaches, nor have they discussed in detail how attackers might evade these defenses. In this section, we explain how and why these two mitigation approaches work, how attackers can evade them, and give analytical studies to evaluate their performance. Meanwhile, for P2P botnet monitoring, we study and analyze the effectiveness of using a captured bot or an infiltrated honeypot to monitor the members of a P2P botnet.

Before we introduce our analysis of mitigation and monitoring approaches, we will first provide basic background introduction on the Kademlia P2P protocol, which is the protocol used by the famous Trojan.Peacomm and Stormnet P2P botnets considered in our study.

Notations used in this section are summarized and explained in Table 8.2.

8.3.1 Background on Kademlia P2P Protocol

Kademlia is a distributed hash table (DHT) protocol designed for P2P networks [39]. Since it is the protocol implemented in Overnet, a P2P network used by Trojan.Peacomm and Stormnet, this kind of network is our focus in the following sections. Because of page limit, we cannot provide detailed introduction. For more information about Kademlia and Kad, please refer to [39, 51, 58].

In a Kademlia-based network, each node has a unique node ID, which is represented by an m-bit binary number. Every node has a routing table containing m lists; each list corresponds to one specific bit of the node ID. Such a list is usually referred as a k-bucket, where k is the maximum number of nodes in each list.

Nodes stored in node A's ith k-bucket ($i = 0, 1, ..., m-1$) are the nodes whose node ID must have the first i bits in common with node A's ID, but have a different

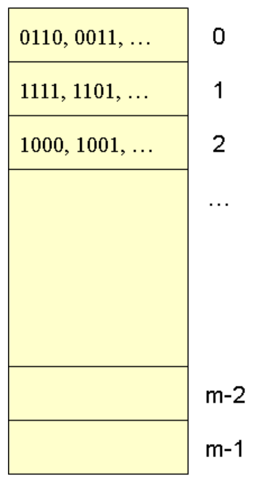
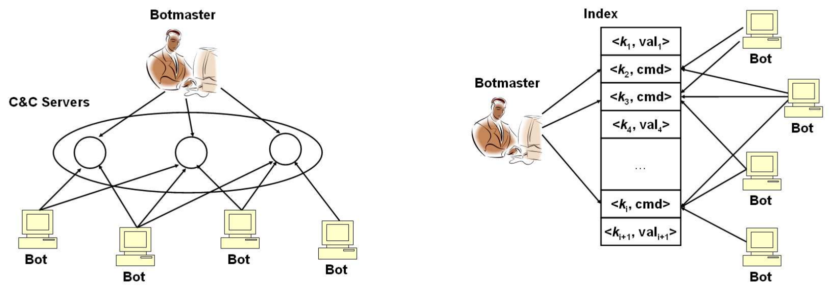


Fig. 8.1 Routing table of a node whose ID has m bits and starts with 1011 (For illustration purpose, we only use 4-bit prefix to represent each node). The table contains m lists; each list holds at most k nodes and is called “k-bucket”. In the 0th k-bucket, the first bit of each node’s ID differs from 1011; in the 1st k-bucket, each node’s ID has the same first bit as 1011, but different second bit. In the 2nd k-bucket, nodes share the first two bits with 1011, but have a different third bit. The rest of the k-buckets follow the same manner.



(**a**) Centralized Botnet (**b**) Index-based P2P Botnet

Fig. 8.2: Similarity of logical C&C structures between traditional centralized botnets and index-based P2P botnets

$(i+1)$-th bit from node A’s ID. Fig. 8.1 is an example of a routing table on a node whose ID starts with 1011.

In the distributed hash table preserved in Kademlia-based network, each entry is a <key, value> pair, in which the key is also an m-bit binary number, and the value part stores the corresponding file or node information. Each <key, value> pair is stored on nodes whose node IDs are the closest ones to the key in the network, and the distance is computed as the exclusive or (XOR) of the key and the node ID. The distance between two nodes is computed in the same way.

Kademlia uses iterative routing mechanism. When node A searches for a key, it first finds α nodes that are the closest ones to the key in its routing table, and then initiates lookup queries to these α nodes. Each one of these nodes will send back a

Table 8.2: PARAMETERS USED IN ANALYSIS

Symbol		Meaning
Kademlia	m	The number of bits used to represent a node ID or a key.
	k	The maximum number of nodes in a bucket in a routing table.
	Δb	The number of bits improved per step for a lookup.
	c	The number of bits two binary numbers (node ID or key) share in common in their prefixes.
Botnet	N	The number of nodes in a P2P network.
	N_{bot}	The number of bots in a P2P network.
	N_{tz}	The number of bots in the target zone.
	N_{index}	The number of nodes poisoned in the target zone.
	N_{Sybil}	The number of Sybil nodes added to the target zone.
	N_{query}	The number of bots sending out queries for commands.
	l_{tz}	The length of a search path in the target zone.
	$P_{success}$	The probability of a bot getting a real command.

response with either the corresponding value part if the <key, value> pair is stored on it, or a certain number of nodes that are the closest ones to the key in its own routing tables if it does not have the pair node A is looking for. A lookup query stops when there is a query hit or when it expires.

Besides Kademlia, Kad is another popular DHT protocol for P2P networks [2]. It has been deployed by eMule [3] file-sharing application. However, Kad is based on Kademlia with a slightly different routing table structure and parameter settings, such as the number of bits of a node ID (it is 160 in Kademlia, but 128 in Kad). These differences do not affect our study on Kademlia-based P2P network in the following, so our analysis applies to both Kademlia and Kad networks. In the later discussion, we do not differentiate Kademlia from Kad, unless it is explicitly mentioned otherwise.

8.3.2 Index Poisoning Defense

8.3.2.1 Defense Idea

Originally, media companies introduced index poisoning attack to prevent illegal distribution of copyrighted content in P2P networks. The main idea is to insert massive number of bogus records into the index system. If a peer receives a bogus record, it could end up not being able to locate the file (nonexistent location), or downloading the wrong file [36].

As we discussed in Section 8.2.3, P2P botnets that implement C&C mechanism of command publishing/subscribing make use of the indices in P2P networks to distribute commands. We refer such botnets as "*index-based*" P2P botnets. If defenders are able to figure out the index keys of the botnet command related index records, they can try to "poison" the index system by announcing false information under the same keys. If the false information overwhelms the real command content, bots that query and retrieve commands will likely end up obtaining false commands. In this way, the C&C channels of the botnet are disrupted.

We believe there are three reasons that index-based P2P botnets are vulnerable to index poisoning defense.

First, a security defect of P2P protocol itself is the root cause. In most P2P networks, there is no central authority to manage the file index system, such that any node, no matter benign or malicious, is able to insert records into the index system. There is no way to authenticate the publishing node and content of the records.

Second, with the help of honeypot and reverse engineering techniques, defenders are able to analyze bot behaviors and bot code, and figure out the bot command related index keys.

Third, in some sense, the C&C architecture of this type of P2P botnets is similar to that of the traditional centralized botnets because of the limited number of index keys for command distribution. As shown in Fig. 8.2, in centralized botnets, commands are published at central sites, where bots are going to fetch the commands; on the other hand, in index-based P2P botnets, commands *cmd* are inserted in the index system by botmasters under special index keys, such as k_2, k_3 and k_i, which are known by bots and queried for retrieving commands.

From the aspect of C&C architecture, index-based P2P botnets logically rely on central points (predefined index keys), while traditional botnets physically rely on central points (predefined C&C servers) for communication. From the aspect of defense, for a traditional C&C botnet, defenders shut down C&C channels by physically removing the C&C servers or blocking access to the servers; while for a P2P botnet, defenders overwhelm real command related records by many bogus records under the same keys (the basic idea of index poisoning technique) to disrupt C&C communication. Therefore, we can draw a conclusion that P2P botnets are not absolutely harder to defend than traditional centralized botnets. If index poisoning defense is valid for a P2P botnet, the P2P botnet is equally easy (or hard) to defend compared with a traditional centralized botnet.

8.3.2.2 Attackers' Possible Counterattack

Although index poisoning defense is effective, it is still possible for attackers to evade it, if they can eliminate the causes discussed in Section 8.3.2.1.

Overbot [50], a new botnet protocol designed by Starnberger et al., addressed the second and the third issues. In Overbot, each bot generates its own index key for retrieving command and that key dynamically changes at a certain rate. In addition the communication between bots and sensors (nodes used by a botmaster to publish commands) is encrypted. Thus, it is very difficult for defenders to crack or predict the index key. Even though defenders are able to do it for one single bot, it is not helpful, because different bots have different index keys. However, for the same reason, sensors have to publish a <key, command> pair for each bot they know periodically, which dramatically increases the sensors' workloads and makes them more susceptible to be detected. In other words, the advantages of Overbot come with the cost of introducing scalability and detectability issues.

Now we present a novel and realistic method that attackers might use to deal with index poisoning defense – an authentication enforcement for command generation and index manipulation. It addresses the first cause of index-based P2P botnets being vulnerable to index poisoning (Section 8.3.2.1). In this approach, only botmasters can insert records to the command index preserved on bots. Bots can only query to fetch commands.

To realize the authentication, a botmaster generates a pair of public/private keys $< K^+, K^- >$, and hard-codes the public key K^+ into the bot code. Later, when the botmaster wants to issue a command m under key k, he/she can insert a record $< k, m, K^-(H(m)) >$ instead of $< k, m >$ into the index on a bot, saying bot A, where $H(m)$ is the hash value of m (i.e., the command is signed by the botmaster). Bot A can decide if the record is created by its botmaster or not by using the public key K^+ to verify the signature. If the signature is authentic, bot A stores this record in the index and waits for others to query, otherwise it discards the fake one. In this way, the index on a bot will not be polluted.

In addition, this authentication mechanism can prevent the spread of false commands. Even if defenders manage to store entries with forged commands in the index on controlled peers (e.g., honeypots infected by a captured bot binary), bots can verify the authenticity of received commands using the public key and disregard the false ones.

Most existing P2P protocols have not implemented this kind of authentication mechanism. Thus in order to deploy it, attackers need to modify the existing P2P protocols, which implies that this counterattack technique can only be applied to bot-only P2P botnets because the botnets cannot join existing P2P file-sharing networks anymore.

8.3.2.3 Analytical Study

In this section, we give an analytical study on performance of index poisoning defense against index-based P2P botnets. Our target is a P2P botnet that implements Kademlia-based DHT protocol for C&C communication. Similar study can be conducted on P2P botnets utilizing other protocols.

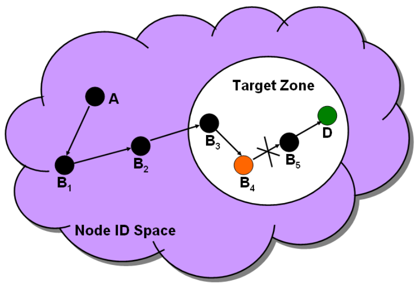
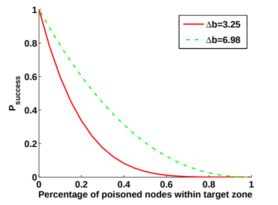
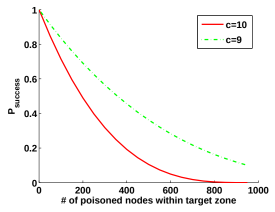


Fig. 8.3 A search path for a key, where node *A* is the initiator and node *D* is the destination. On this path node B_4 could be a node targeted by defenders to interrupt the search.

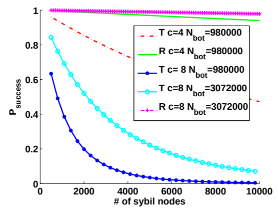
As introduced in Section 8.3.1, in a Kademlia-based DHT, each entry is a <key, value> pair, and each pair is stored on at least one node whose node ID is closest to the key in the network. If defenders want to pollute a P2P botnet's index records under key K, they need to contact nodes (poisoned nodes) whose IDs are close to K, and store pairs like <K, false value> on them. In this way, when a bot queries for key K to retrieve commands, those poisoned nodes will have a good chance to appear on the search path and return false query value, and hence, prevent bots from reaching nodes who possess the real commands. As illustrated in Fig. 8.3, a bot node A initiates a lookup for index key K, the search path is supposed to go through node B_1, B_2, B_3, B_4 and B_5, until it reaches node D who has the pair <K, command>. If a pair <K, false command> has been added in the index on node B_4, when the lookup



(**a**) Index poisoning with different Δb
$N = N_{bot} = 980000$,
$c = 4$



(**b**) Index poisoning with different c
$N = N_{bot} = 980000$,
$\Delta b = 3.25$



(**c**) Sybil attack
(T-targeted Sybil nodes,
R-random Sybil nodes)
$c = 4$, $\Delta b = 3.25$

Fig. 8.4: Performance of index poisoning defense and Sybil defense techniques illustrated by numerical results

message reaches node B_4, the node would return the false command and terminate the search.

We assume that node IDs are uniformly distributed over the entire Kademlia ID space, which is supported by the study in [51]. Suppose defenders choose N_{index} nodes whose IDs have at least the first c bits in common with K to inject and poison their index records. We call the zone around K the "*target zone*" and all poisoned nodes are in the target zone. Only when a lookup path enters the target zone, it is possible that a poisoned node will be chosen, return a search result and terminate the search. Let x be the probability of choosing a poisoned node in one lookup step, then $1-x$ is the probability of not choosing one. Therefore the probability of a bot obtaining the real command is

$$P_{success} = (1-x)^{l_{tz}} \tag{8.1}$$

where l_{tz} is the length of a search path within the target zone.

When a peer initiates a lookup for a key, in general, the expected number of steps required to perform a lookup is given as follows [58]:

$$l = \frac{\log_2 N}{\Delta b} \tag{8.2}$$

where N is the size of the network. We assume all nodes in the P2P network are bots, so $N_{bot} = N$ in this case. Δb is the number of bits improved per step, which depends on the structure of the routing table. Thus within the target zone, $l_{tz} = log_2 N_{tz}/\Delta b$. Since node IDs are uniformly distributed, the number of nodes in the target zone is $N_{tz} = N/2^c$, and $x = N_{index}/N_{tz}$. The complete formula to calculate $P_{success}$ is

$$P_{success} = (1 - \frac{2^c \times N_{index}}{N})^{(\log_2 N - c)/\Delta b} \tag{8.3}$$

According to Equation (8.3), the performance of index poisoning technique depends on four parameters. We have provided numerical results in Fig. 8.4 to show their impacts on $P_{success}$ by changing the parameters.

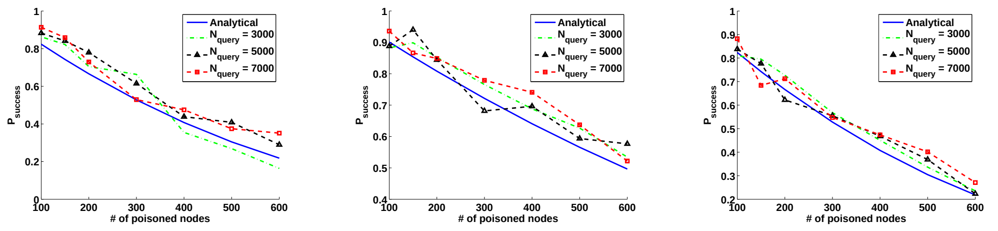
Fig. 8.4(a) illustrates that a botnet would be more robust to index poisoning defense, if for each lookup more bits can be improved, i.e., the average length of search path is shorter. When the search path is short, poisoned nodes have less chance to be chosen along the path[2].

It is shown in Fig. 8.4(b) that in order to achieve better performance, defenders could choose a larger c, i.e. choosing nodes that are closer to the command related key to poison. However it is not always a good idea to choose a large c, because we want to have at least one step along the lookup path in the target zone, otherwise bots can directly get commands without going through any node in the target zone. In other words, $l_{tz} \geq 1$, i.e., $c \leq \log_2 N - \Delta b$. In our case, $N = 980000$, $\Delta b = 3.25$, so $c \leq 16.7$, and for the setup $c = 9$ and $c = 10$ used in Fig. 8.4(b), l_{tz} is 3.35 and 3.05 respectively.

The size of the network would also affect the performance of index poisoning defense. However, it does not matter that much, since given a fixed percentage of poisoned nodes in the target zone, it can barely change l_{tz} due to the log_2 operator ($log_2 980000 = 19.90$ and $log_2(12000 \times 2^8) = 22.29$)[3].

[2] The value of Δb was estimated in [58]. 3.25 is the worst case, while 6.98 is the best case.

[3] An estimate was given in [58] that the Kad network has around 980,000 concurrent peers. Authors of [51] claimed that the population of peers in Kad network is between 12,000$\times 2^8$ and 20,000$\times 2^8$.



(**a**) N_{bot}=N=10000, m=16 (**b**) N_{bot}=N=20000, m=16 (**c**) N_{bot}=N=10000, m=160

Fig. 8.5: Comparison between analytical and simulation results of $P_{success}$ for index poisoning defense. The simulated P2P botnet is Kademlia-based with a $D(1,1,8)$ routing table structure, and $c = 3$.

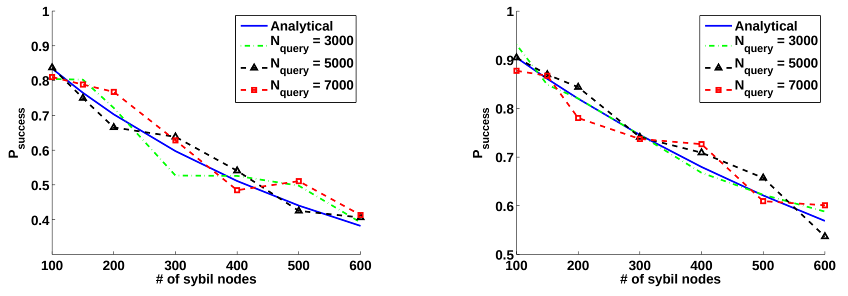
8.3.2.4 Simulation Evaluation

To evaluate the accuracy of our analysis, we develop a P2P botnet simulator based on OverSim [8], an open source P2P simulator. The P2P botnet we simulate employs Kademlia protocol for the C&C communication.

In [58], Stutzbach et al. defined a system $D(b,r,k)$, which uses b-bit symbols with r-bit resolution and k-buckets, to represent the routing table structure of a Kademlia-based DHT protocol. According to this definition, the routing table structure implemented in our simulator can be denoted as $D(1,1,8)$ (i.e., $b = 1$, $r = 1$, $k = 8$), which is consistent with the basic Kademlia design. Correspondingly, the average bits improved per lookup step in our simulated system is $\Delta b = 4.41$[4]. To reduce the computation time, we set m to be 16 instead of 160 which is the default setting in Kademlia protocol. Experiments on comparing the performance of index poisoning defense with different values of m (Fig. 8.5(a)(c)) show that the value of m does not matter.

We consider two different sizes of such botnets with N=10000, and N=20000, respectively. The parameters we change are N_{query}, the number of bots who queries for commands and N_{index}, the number of bots whose indices have been poisoned. The node IDs of these N_{index} poisoned nodes share at least the first $c = 3$ bits with a given command related key. In each simulation run, every bot in the set of N_{query} query bots looks for the command once; and we calculate $P_{success}$ based on how many of them actually obtain the real command. To derive the average value of $P_{success}$, we conducted at least 20 simulation runs for each botnet configuration.

[4] Please refer to the paper [58] for the detailed formulas to compute Δb given the routing table structure $D(b,r,k)$.



(**a**) N_{bot}=10000 (**b**) N_{bot}=20000

Fig. 8.6: Comparison between analytical and simulation results of $P_{success}$ for Sybil defense. The simulated P2P botnet is Kademlia-based with a $D(1,1,8)$ routing table structure, and $c = 3$.

Fig. 8.5 shows the experiment results comparing to the analytical result obtained from our analysis. According to Equation (8.3), $P_{success}$ does not depend on N_{query}. Therefore, only one curve is plotted as the analytical result (the solid blue line in the figure). As we can see, the analytical result matches with simulation results of $P_{success}$ with around 10% of errors. Fig. 8.5(c) plots the results from another simulation with the same settings as Fig. 8.5(a), except that $m = 160$. According to our analysis, Equation (8.3), $P_{success}$ does not depend on the value of m. Fig. 8.5(c) confirms this conclusion.

8.3.3 Sybil Defense

8.3.3.1 Defense Idea

In a normal P2P file-sharing network, "Sybil attack" is referred as the forging of multiple identities by attackers to subvert the reputation system [17]. The reason of P2P networks being vulnerable to Sybil attack is that peers can join the network without authentication or validation of their identities. It is an inherent vulnerability for most P2P networks and protocols [52, 64].

For the same reason, an index-based P2P botnet that implements a traditional P2P protocol will also be susceptible to Sybil-based defense as well. With the knowledge of index keys used for command distribution, defenders can add Sybil nodes (such as honeypots) into the botnet to re-route or monitor the command related traffic.

How to set up Sybil nodes depends on the actual P2P system implementation. In an unstructured P2P network, in order to capture more botnet traffic, defenders will set up Sybils to be peers with more important roles, e.g. setting up Sybil nodes as ultrapeers in Gnutella because only ultrapeers are allowed to forward messages. In a structured P2P network, such as Kad, the node IDs of Sybil nodes should not be chosen randomly, but be close to a known command related index key, as discussed in [16, 27]. In this way, command query traffic for the key will go through Sybil nodes with a high probability according to the Kad's routing algorithm. We call such defense "*targeted*" Sybil defense.

For defenders, the cost for Sybil defense is usually higher than index poisoning defense. This is because either a physical or a virtual machine is needed to set up a Sybil node; in other words, more Sybil nodes require more computer resources, while publishing different records to poison index system can be done by a single node.

8.3.3.2 Attackers' Possible Counterattack

Similarly, approaches used for protecting today's P2P networks from Sybil attack may also work for botmasters to prevent defenders from infiltrating their P2P botnets using Sybil nodes. Here we briefly introduce possible counterattack methods.

In Kademlia-based P2P networks, a node ID can be constructed by hashing the node's IP address as what Chord does [55], rather than being randomly generated by a joining node itself like what Kad does [51]. If the network uses a node's IP address to generate the node ID, Sybil nodes will not be able to choose any IDs they want. When a botmaster applies this scheme in his/her P2P botnet, defenders cannot target a specific key to set up their Sybil nodes. In this case, Sybil nodes are just randomly added into the botnet, which is referred as "*random*" Sybil defense. This kind of Sybil defense is much less effective than targeted Sybil defense as explained in the next section.

Furthermore, caching technique [39], which was meant to solve "hot spots" problem, can also be utilized by a P2P botnet to reduce the effectiveness of Sybil defense. Because the command related index records will be stored not only on bots that were chosen at the beginning by their botmaster (e.g., unstructured network) or according to the protocol (e.g., structured network), but also on bots that may not be easily identified. Thus even targeted Sybil defense cannot cover all the bots that possess the command information.

8.3.3.3 Analytical Study

Now we analyze Sybil defense on the same type of P2P botnets as in Section 8.3.2.3, Kademlia-based P2P botnets. The notations have the same meaning unless explicitly mentioned otherwise.

If node IDs can be chosen randomly, defenders can create special N_{Sybil} Sybil nodes, whose node IDs share at least the first c bits with an index key K, and add them into the botnet. Once a Sybil node is on the path of a command lookup, it can re-route the message or return a false command and terminate the search, and hence, prevent the query bot from obtaining the real command.

As we can see, Sybil defense shares the same defense principle with index poisoning defense. They both try to manipulate the command lookup path, as shown in Fig. 8.3. Sybil defense achieves this manipulation by adding new special nodes (controlled by defenders) to the network, i.e., node B_4 in Fig. 8.3 is a Sybil node added by defenders, while index poisoning defense achieves this by poisoning the nodes (bots probably) already in the network.

Following the same analysis procedure as what we used in Section 8.3.2.3, the probability of a bot successfully getting the real command $P_{success}$ can be calculated using Equation (8.1), except that x becomes the probability of choosing a Sybil at each step along the search path within the target zone, which is $N_{Sybil}/(N_{Sybil}+N_{tz})$. So

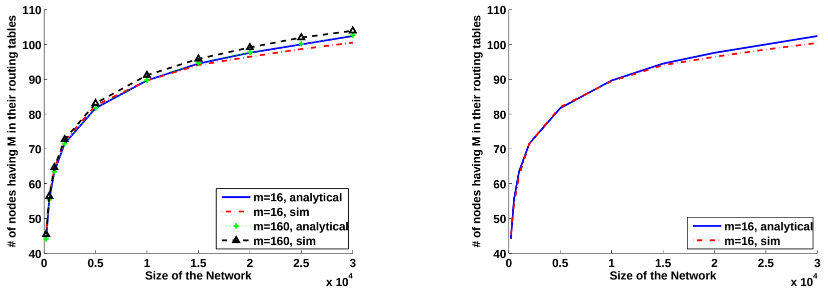
$$P_{success} = (1 - \frac{N_{Sybil}}{N_{Sybil}+N_{tz}})^{l_{tz}} \tag{8.4}$$

where $l_{tz} = log_2(N_{Sybil}+N_{tz})/\Delta b$, and $N_{tz} = N_{bot}/2^c$.

Differing from what used in the index poisoning defense analysis, the size of the network used in Sybil defense analysis is not the number of the bots, but the total number of bots and Sybil nodes, i.e., $N = N_{bot} + N_{sybil}$, since Sybil nodes added by defenders are not real bots.

When a verification mechanism for node ID is applied in the botnet (Section 8.3.3.2) such that defenders can only conduct random Sybil defense, the whole network becomes the target zone, i.e., $N_{tz} = N_{bot}$. Simply substituting N_{tz} in Equation (8.4) by N_{bot}, we can get the following formula to compute $P_{success}$ in this "random" Sybil defense.

$$P_{success} = (1 - \frac{N_{Sybil}}{N_{Sybil}+N_{bot}})^{log_2(N_{Sybil}+N_{bot})/\Delta b} \tag{8.5}$$



(**a**) Without churn (**b**) With churn

Fig. 8.7: Comparison between analytical and simulation results of $\overline{N}_{routing}$. The simulated P2P botnet is Kademlia-based with a $D(1,1,8)$ routing table structure.

It is shown in Fig. 8.4(c) that under the same circumstance targeted Sybil defense greatly outperforms the random one. This is because in the former case, Sybil nodes with specially chosen IDs have more chances to appear along a search path than those in the latter case. With limited resources that defenders may use to launch Sybil defense, if the Sybil nodes are closer to the key K (i.e., larger c), the defense performance would be better. Furthermore, Sybil defense is more effective if the network is smaller.

8.3.3.4 Simulation Evaluation

We use simulation experiments as well to verify our analysis. The network settings and parameter configurations are the same as those used in Section 8.3.2.4.

We assume N_{Sybil} Sybil nodes, whose node IDs share at least the first 3 bits ($c = 3$) with a given key, are added by defenders during the construction of the botnet. When the botnet is built up, the whole network has $N = N_{bot} + N_{Sybil}$ nodes, and N_{query} bots will start querying for commands. Again, we use 20 simulation runs to obtain the average simulation results. The simulation results along with the numerical results are plotted in Fig. 8.6, which shows that our analysis is consistent with the simulations.

8.3.4 P2P Botnet Passive Monitoring

Botnet monitoring is an important component in the overall botnet defense. A good monitoring could collect valuable information about the botnet under observation, such as the size of the botnet, the unique features of the botnet network traffic, and the identities of bots, etc.

Monitoring systems can be classified as either active or passive. Active monitoring usually starts with one or a couple of known bots within the network. By actively contacting these bots, defenders could get to know the identities and information of more bots, and make contacts with those newly discovered bots in the next round. The monitoring is done actively and iteratively until no more unknown bots can be discovered. Passive monitoring is carried out by Sybil nodes put by defenders in the botnet. Unlike active monitoring, these nodes do not actively contact other nodes in the network; they only perform the routine tasks like other normal nodes, such as forwarding traffic and responding to queries. Nodes that have contacted the monitoring nodes are recorded for further analysis. Passive monitoring has the advantage of being stealthy, and hence, harder for botmasters to detect and remove those monitoring nodes from their botnets.

In this section, we provide mathematical analysis of the effectiveness of passive monitoring. In other words, we would like to figure out how many bots in a P2P botnet a passive monitoring node can monitor after a certain time period. In this section we address this problem in a Kademlia-based P2P botnet as well.

8.3.4.1 Analytical Study

Suppose defenders have set up one passive monitoring node in a P2P botnet. We want to estimate the number of bots that have this monitoring node in their routing tables, denoted as $N_{routing}$. According to Kademlia protocol, a node would contact nodes in its routing table from time to time because of query or routing table refresh activities. Therefore, $N_{routing}$ is the lower bound for the number of bots that can be observed by a passive monitoring node.

In a Kademlia-based P2P botnet with N nodes, we denote each node as B_i, where $i = 1,2,...,N$, and the time of node B_i joining the network is denoted as t_i, where we assume $t_i < t_j$, if $i < j$, and $t_i \neq t_j$, if $i \neq j$, i.e, no two nodes join the network at the same time. Moreover, when node B_i joins the network, the current size of the network is denoted as N_i. Because of the way we index the nodes, we can easily know that $N_i = i$.

To compute the number of nodes who have a specific node B_i in their routing tables, we need to consider two types of nodes: the nodes joining the network before B_i, referred as $Nodes_{before}$, and the nodes joining the network after B_i, referred as $Nodes_{after}$.

When node B_i joins the network, there are already $N_{i-1} = i-1$ nodes in the network. We can classify these N_{i-1} nodes into m groups. The c-th group ($c = 0,1,2,\cdots,m-1$) contains the nodes whose IDs share the first c bits with node B_i's ID but differ at the $(c+1)$-th bit. Because node IDs are uniformly distributed, the number of nodes in the c-th group is $N_{share}(c) = N_{i-1}/2^{c+1}$. If a node in the c-th group whose c-th bucket is not full (i.e., $N_{share}(c) \leq k$), it will add node B_i in this bucket, otherwise it will not contain node B_i in its routing table. As c increases, the size of c-th group monotonously decreases. When $0 \leq c < c_0$ where $c_0 = \lceil \log^{N_{i-1}/k} - 1 \rceil$ (c_0 is obtained by letting $N_{share}(c) = k$), $N_{share}(c) > k$, and hence, we do not need to consider nodes in these groups. Therefore the number of $Nodes_{before}$ who would add node B_i into their routing tables can be calculated as follows:

$$N_{before}(i) = \sum_{c=c_0}^{m-1} \frac{N_{i-1}}{2^{c+1}}, \tag{8.6}$$

where $c_0 = \lceil \log^{N_{i-1}/k} - 1 \rceil$, which is obtained by letting $N_{share} = k$.

After node B_i has joined the network, for the nodes joining in later on, they may add node B_i into their routing tables as well. Let's consider a node $B_j, i < j$, the probability of these two nodes' IDs sharing the first c bits but differing at the $(c+1)$-th bit is

$$P_{share}(c) = \frac{2^{m-(c+1)} - \frac{N_j}{2^{c+1}}}{2^m - N_j} = \frac{1}{2^{c+1}}, c = 0,1,...,m-1. \tag{8.7}$$

Suppose node B_i and node B_j share the first c bits but differ at the $(c+1)$-th bit in their IDs, there are $N_{share}(c) = N_{j-1}/2^{c+1}$ candidates for node B_j to pick and add to its c-th k-bucket, and node B_i is in this candidate set. We can consider two possible scenarios. When $N_{share}(c) > k$, node B_j randomly picks k nodes from the candidate set to put in its routing table; when $N_{share}(c) \leq k$, all the nodes in the candidate set will be chosen. Therefore, the probability of node B_j adding node B_i into its routing table is

$$P_{add}(c) = \begin{cases} \frac{k}{\frac{N_{j-1}}{2^{c+1}}}, & \frac{N_{j-1}}{2^{c+1}} > k \\ \\ 1, & \frac{N_{j-1}}{2^{c+1}} \leq k \end{cases} \tag{8.8}$$

Let $c_1 = \lceil \log^{N_{j-1}/k} - 1 \rceil$, i.e., $N_{j-1}/2^{c_1+1} = k$, we can rewrite Equation (8.8) and get Equation (8.9).

$$P_{add}(c) = \begin{cases} \frac{k}{\frac{N_{j-1}}{2^{c+1}}}, & c < c_1 \\ \\ 1, & c \geq c_1 \end{cases} \tag{8.9}$$

Therefore, the number of $Nodes_{after}$ that would have node B_i in its routing table can be calculated as follows:

$$N_{after}(i) = \sum_{c=0}^{m-1} P_{share}(c) \times P_{add}(c) \tag{8.10}$$

For a specific node B_i, the total number of nodes having it in their routing tables is

$$N_{routing}(i) = N_{before}(i) + N_{after}(i) \tag{8.11}$$

and the average number of nodes that have a monitoring node in their routing tables is

$$\overline{N}_{routing} = \frac{1}{N} \sum_{i=1}^{N} N_{routing}(i) \tag{8.12}$$

8.3.4.2 Simulation Evaluation

Still we simulate the same Kademlia-based P2P botnet as the one in Section 8.3.2.4 and Section 8.3.3.4. We carry out two types of experiments: P2P botnets without churn and botnets with churn, where churn refers to the network dynamics caused by nodes' joining and leaving activities.

For P2P botnets without churn, we consider once a botnet is constructed, the botnet is stable, i.e., no nodes will leave the network and no more nodes will join the network as well. $\overline{N}_{routing}$, the average number of nodes which have a given monitoring node in their routing tables for different scales of networks is shown in Fig. 8.7(a). We can see that our analysis precisely estimates $\overline{N}_{routing}$.

However, in the real world, the churn does exist in P2P botnets. To make our experiment more realistic, we introduce the churn events in our simulations. In our simulated P2P network, node joining and node leaving events will happen, and we assume the time interval between two churn events t_{churn} follows a truncated normal distribution (i.e., $t_{churn} \sim N(\mu, \sigma^2)$ and $t_{churn} > 0$). In order not to favor any one of the node's joining and node's leaving, when a churn event happens, we set the probability of it being a node's joining P_{in} and of it being a node's leaving P_{out} to be the same, i.e., they are both 50%.

Fig. 8.7(b) shows our simulation results when considering churn. In our experiments, all simulations run for the same amount of time (30,000 unit time) and t_{churn} follows the same distribution ($\mu = 15$ and $\sigma = \mu/3$). As a result, in each simulation, there are around 2000 node joining/leaving events. If the size of the network is small, only a small fraction of original bot nodes (e.g., the first N nodes) still exist in the network when the simulation ends. But in a relatively large network, a large fraction of original nodes still exist in the network. For example, when N=200, there are around 4%-5% of the first 200 nodes still in the network at the end of the simulation; while when N=30,000, 96% of the original 30,000 nodes remain in the network. From another perspective, we can view this phenomenon as the illustration of monitoring performance under different churn intensities. Since in our experiments, we cover the sizes of network ranging from 200 to 30,000, we have considered the monitoring performance under different churn intensities. As what is shown in Fig. 8.7(b), our analysis can still well evaluate $\overline{N}_{routing}$ even with churn.

8.3.5 Others Countermeasures

In the following, we present several general ideas to defend against P2P botnets.

8.3.5.1 Detection

Being able to detect bot infection can stop a new-born botnet in its infant stage. Signature-based malware detection is effective and still widely used. But anti-signature techniques, such as polymorphic technique [31], make it possible for malware to evade such detection systems. Therefore, instead of doing static analysis, defenders start considering dynamic information for detection. For example, the system proposed by Gu et al. [21] is based on dynamic pattern matching.

Anomaly detection is another direction, since bots usually exhibit different behaviors from legitimate P2P users, such as sending queries periodically, always querying for the same content, or repeatedly querying but never downloading.

In addition, distributed detection is another approach, such as the self-defense infrastructure presented in [69], and two approaches against ultra-fast topological worms in [67].

8.3.5.2 Monitoring

Monitoring botnets help people better understand their motivations, working patterns, evolution of designs, etc. There are two effective ways to conduct P2P botnet monitoring.

For parasite and leeching P2P botnets, we can choose legitimate nodes in the host P2P networks as sensors for botnet monitoring. Usually sensors are peers that play important roles in the network communication, such as ultrapeers in Gnutella networks, such that more information can be collected. In DHT-based P2P networks, the search path of a specific key is relatively fixed, even if the search starts at different nodes. So the sensor selection depends on the monitoring targets and routing algorithm implemented in the system.

Honeypot techniques [49] are widely used for botnet monitoring. The way to set up honeypots in a P2P botnet is similar to choosing sensors. The difference is that honeypots are hosts added to the network on purpose by defenders, while sensors are chosen from the nodes who are already in the network.

8.3.5.3 Mitigation

The ultimate purpose of studying botnets is to shut them down. We can either 1) remove discovered bots, or 2) shut down C&C channels of botnets.

A botnet that relies on bootstrapping for construction is vulnerable during its early stage. Isolating or shutting down bootstrap servers or the bots in the initial list that are hard-coded in bot code can effectively prevent a new-born botnet from growing into a real threat.

P2P botnets can also be shut down or partially disabled by removing bot members. There are two modes of bot removal: random and targeted. The former means disinfecting the host whenever it is identified as a bot. The latter means removing critical bots, such as the ones that are important for C&C communication, when we have the knowledge of the topology or C&C architecture of a P2P botnet. Two metrics to evaluate the effectiveness of targeted removal were proposed in [63].

Shutting down detected bots is slow in disabling a botnet and sometimes impossible to do (e.g., you have no control of infected machines abroad). So a more effective and feasible way is to interrupt botnet C&C communication such that bots cannot receive orders from their botmaster. This approach has been carried out well for centralized botnets through shutting down the central C&C sites, but is believed to be more difficult to do for P2P botnets.

However, we find that this general understanding of "*P2P botnet is much more robust against defense*" is misleading. In fact, index-based P2P botnets are as vulnerable as centralized botnets, if the counter defense methods we presented in Section 8.3.2.2 and 8.3.3.2 are not implemented. Index poisoning defense (Section 8.3.2) and Sybil defense (Section 8.3.3) can be quite effective to fight against such botnets.

8.3.6 Discussion

It is worthy to point out that the search process we discussed in Section 8.3.2.3 and Section 8.3.3.3 can be performed in two different manners – iterative and recursive. Let us use the scenario presented in Fig. 8.3 to explain the difference between these two search modes: the iterative search route would be $A \to B_1 \to A \to B_2 \to A \to B_3 \to A \to B_4 \to A \to B_5 \to A \to D$, while the recursive search route would be $A \to B_1 \to B_2 \to B_3 \to B_4 \to B_5 \to D \to B_5 \to B_4 \to B_3 \to B_2 \to B_1 \to A$. A P2P protocol could use either one of them. For instance, Kademlia employs the iterative search algorithm, and the Nugache P2P botnet has implemented the recursive routing. Although the routes are different, our analysis applies to both of the search algorithms. This is because, in our analysis, what we care about is the number of distinct nodes that a search message would go through besides node A and node D within a target zone; this number depends on the length of the path $A \to B_1 \to B_2 \to B_3 \to B_4 \to B_5 \to D$, and in both search cases this length is the same.

In addition, as we mentioned before, the ideas of index poisoning and Sybil were first introduced in legitimate P2P networks, and passive monitoring can also be deployed in current P2P file sharing networks. When P2P botnets use the same P2P protocols, these techniques can be leveraged to fight against these botnets as well. Therefore, our analysis of these three techniques is applicable to not only P2P botnets, but also to legitimate P2P systems. Moreover, in our analysis, we mainly talked about P2P botnets utilizing Kademlia for command and control; however, index poisoning defense and Sybil defense techniques are also valid for P2P botnets that rely on P2P networks for other communication, such as Storm botnet, which utilizes a P2P network to help bots join its hierarchical multi-tier command and control network. Therefore, our analysis is valid for general P2P botnets, no matter whether they use P2P networks for command dissemination, or for other communications.

8.4 Related Work

P2P botnets, as a new form of botnet, have appeared in the last few years and obtained people's attention. In [19] Grizzard et al., conducted a case study on Trojan.Peacomm botnet. Later on, Holz et al., adapted tracking technique used to mitigate IRC-based botnets and extended it to analyze Storm worm botnets [27]. Trojan.Peacomm botnet and Stormnet are two typical P2P botnets. Although bots in these two botnets are infected by two different malware, Trojan.Peacomm and Storm worm respectively, both of their C&C mechanisms are based on Kademlia [39]. And a botnet protocol which is also based on Kademlia was proposed by Starnberger et al. [50]. Moreover, to be well prepared for the future, there are some other botnets whose architecture is similar to P2P architecture, such as an advanced hybrid P2P botnet [63], super botnet [61] and the Loosely Coupled P2P botnet (lcbot) [11]. Ping et al. [62] studied P2P botnets along multiple dimensions including botnet construction, command and control mechanisms, performance measurements, and mitigation approaches. Rossow et al. [45] used formal graph model to capture the intrinsic properties and fundamental vulnerabilities of P2P botnets; however, this work does not provide mathematical modeling of the mitigation techniques against P2P botnets. Han et al. [25] presented a matrix model for P2P botnets and provided formulas of five performance metrics including connection degree, connection degree ratio, connection ratio, exposure ratio and average hop count. Singh et al. [48]

Built a distributed intrusion detection framework that can be used to detect P2P Botnet by machine learning approach.

There have been some systematic studies on general botnets. Barfor and Yegneswaran [7] studied and compared four widely-used IRC-based botnets from seven aspects: botnet control mechanisms, host control mechanisms, propagation mechanisms, exploits, delivery mechanisms, obfuscation and deception mechanisms. Considering aspects such as attacking behavior, C&C model, rally mechanism, communication protocol, evasion technique and other observable activities, Trend Micro [40] proposed a taxonomy of botnet threads. In [13] Dagon et al., also presented a taxonomy for botnets but from a different perspective. Their taxonomy focuses on the botnet structure and utility. And in 2008, a botnet research survey [70] done by Zhu et al., classified research work on botnets into three categories: bot anatomy, wide-area measurement study and botnet modeling and future botnet prediction. Bailey et al. presented another survey, which provided an overview of current botnets, discussed how different types of networks can affect the effectiveness of botnet detection mechanism, and talked about various detection techniques that have been proposed [6]. What differs our work from theirs is that we focused on newly appeared P2P botnets, and tried to understand P2P botnets along four dimensions: P2P botnet construction, C&C mechanisms, measurements and defenses.

Modeling P2P botnet propagation is one dimension we did not discuss in this chapter. Król [34] presented theoretical study of malware propagation in various complex networks. In the research work [46], Ruitenbeek and Sanders presented a stochastic model of the creation of a P2P botnet. In [14], Dagon et al., proposed a diurnal propagation model for computer online/offline behaviors and showed that regional bias in infection will affect the overall growth of the botnet. [44] formulated an analytical model that emulates the mechanics of a decentralized Gnutella type of peer network and studied the spread of malware on such networks. Both [68] and [59] presented an analytical propagation model of P2P worms, but the former targets topological scan based P2P worms, while the latter targets passive scan based P2P worms.

Many researchers have investigated on detection and mitigation of traditional centralized C&C botnets. Wurzinger et al. presented an approach to automatically generate models for botnet detection [66]. Their models are generated based on the fact that every bot responds to the botmaster in a specific way. Researchers try to distinguish bot behavior from human behavior, in order to detect botnets. For example, in [38], malicious channels created by bots are differentiated from normal traffic generated by human beings; and in [22], hypothesis testing is used to separate botnet C&C dialogs from human-human conversations. Pattern recognition approaches and clustering algorithms are widely used for botnet detection. Chang and Daniels proposed a node behavior profiling approach to capture the node behavior clusters in a network for botnet C&C communication detection [10]. And a Bayesian approach for detecting bots based on the similarity of their DNS traffic to that of known bots is presented in [60]. In addition, Gu et al., proposed three botnet detection systems: BotMiner [20] – a botnet detection framework by performing cross cluster correlation on captured communication and malicious traffic, BotSniffer [23] – a system that can identify botnet C&C channels in a local area network based on the observation that bots within the same botnet will demonstrate spatial-temporal correlation and similarity and BotHunter [21] – a bot detection system using IDS-Driven Dialog Correlation according to defined bot infection dialog model.

Furthermore, botnet infiltration and monitoring is also an very active topic in botnet research community. In [29], Kang et al., presented a passive P2P monitor, which can enumerate the infected hosts regardless whether or not they are behind a firewall or NAT, and conducted an empirical study on Storm botnet. Li et al. [35], monitored botnets probing activities and addressed the problems like botnet's scanning strategies and attack target selection policies. In [30], Kanich et al., pointed out a number of challenges that arise in using crawling to measure the size, topology, and dynamism of distributed botnets. People infiltrate specific botnets, such as MegaD botnet [12], Torpig bot [56] and [41], in order to understand their architectures, communication protocols, behaviors, etc. In addition, botnet infiltration and monitoring can be very helpful for fighting against malicious activities. In [32, 33, 42], the data collected through infiltrating and monitoring botnets are used for spam detection and analysis.

Some researchers have studied theoretical models of complex network in terms of network robustness against general network failure or malicious attacks. Schneider et al. [47] presented mathematical analysis of complex networks and introduced a new measure for robustness. They have demonstrated that electricity grid and Internet can significantly improve their robustness against malicious attacks with small changes in the network structure. Hayes et al. [26] presented a new algorithm to improve self-healing in peer-to-peer networks against node insertion or deletion attacks. Louzada et al. [37] presented a new rewiring method to modify a network topology improving its robustness, based on the evolution of the network largest component during a sequence of targeted. attacks.

8.5 Conclusion

P2P botnets, as a new advanced form of botnets, have attracted attentions from both botmasters and security defenders. In this chapter, we first presented a systematical study on P2P botnets. We discussed in detail each stage in the life cycle of P2P botnets, and classified P2P botnets into three categories: parasite, leeching and bot-only P2P botnets. Then among possible directions for P2P botnet defense, we focused on two mitigation techniques against P2P botnets – index poisoning defense and Sybil defense, and one monitoring technique – passive monitoring, and analyzed their effectiveness in terms of several factors, such as the size of a botnet, the settings of the communication protocol, the range of the defense deployment. Simulation-based experiments have shown that our analysis is accurate. This work provides guidance for security professionals on how to carry out these three defense techniques to achieve better performance. In the mean time, we discussed how attackers might react to avoid or reduce the effectiveness of index poisoning defense and Sybil defense techniques, which help people get prepared for the future in case such methods are deployed by attackers. Furthermore, based on our study, we obtained a counterintuitive finding: because of the similar information dissemination structure, P2P botnets that rely on index for command or other critical information dissemination may be as easy (or as hard) to be shut down as the centralized botnets.

References

[1] Http://www.symantec.com/security_response/index.jsp
[2] Http://en.wikipedia.org/wiki/Kad_network
[3] emule. Http://www.emule-project.net/
[4] SdDrop. Http://www.viruslist.com/en/viruses/encyclopedia?virusid=24282
[5] Andriesse, D., Rossow, C., Stone-Gross, B., Plohmann, D., Bos, H.: Highly resilient peer-to-peer botnets are here: An analysis of Gameover Zeus . In: Proc. of 8th International Conference on Malicious and Unwanted Software: ”The Americas” (MALWARE) (2013)
[6] Bailey, M., Cooke, E., Jahanian, F., Xu, Y., Karir, M.: A Survey of Botnet Technology and Defenses. In: Proc. of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security (2009)

[7] Barford, P., Yegneswaran, V.: An Inside Look at Botnets. In Series: Advances in Information Security (2006)

[8] Baumgart, I., Heep, B., Krause, S.: OverSim: A Flexible Overlay Network Simulation Framework. In: Proc. of the 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM '07, Anchorage, AK (2007)

[9] Bhaduri, K., Das, K., Kargupta, H.: Peer-to-peer data mining, privacy issues, and games **4476**, 1–10 (2007)

[10] Chang, S., Daniels, T.E.: P2P botnet detection using behavior clustering & statistical tests. In: Proc. of the 2nd ACM workshop on Security and artificial intelligence (AISec '09), Chicago (2009)

[11] Chang, S., Zhang, L., Guan, Y., Daniels, T.E.: A Framework for P2P Botnets. In: Proc. of the 2009 International Conference on Communications and Mobile Computing (CMC '09), Kunming, Yunnan, China (2009)

[12] Chox, C.Y., Caballeroyx, J., Grierx, C., Paxsonzx, V., Song, D.: Insights from the Inside: A View of Botnet Management from Infiltration. In: Proc. of the 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats, San Jose, CA (2010)

[13] Dagon, D., Gu, G., Lee, C., Lee, W.: A Taxonomy of Botnet Structures. In: Proc. of the 23rd Annual Computer Security Applications Conference (ACSAC'07) (2007)

[14] Dagon, D., Zou, C.C., Lee, W.: Modeling Botnet Propagation Using Time Zones. In: Proc. of the 13th Annual Network and Distributed System Security Symposium (NDSS'06) (2006)

[15] Damfling, H.: Gnutella web caching system.
http://www.gnucleus.org/gwebcache/specs.html

[16] Davis, C.R., Fernandez, J.M., Neville, S., McHugh, J.: Sybil Attacks as a mitigation strategy against the Storm botnet. In: Proc. of the 3rd International Conference on Malicious and Unwanted Software (Malware'08) (2008)

[17] Douceur, J.R.: The Sybil Attack. In: Proc. of the 1st International Workshop on Peer-to-Peer Systems (2002)

[18] Enright, B., Voelker, G., Savage, S., Kanich, C., Levchenko, K.: Storm: When Researchers Collide. In: USENIX ;login: 33(4) (2008)

[19] Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B., Dagon, D.: Peer-to-Peer Botnets: Overview and Case Study. In: Proc. of the 1st USENIX Workshop on Hot Topics in Understanding Botnets (HotBots '07), Cambridge, MA (2007)

[20] Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In: Proc. of the 17th USENIX Security Symposium (Security'08) (2008)

[21] Gu, G., Porras, P., Yegneswaran, V., Fong, M., Lee, W.: BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In: Proc. of the 16th USENIX Security Symposium (Security'07) (2007)

[22] Gu, G., Yegneswaran, V., Porras, P., Stoll, J., Lee, W.: Active Botnet Probing to Identify Obscure Command and Control Channels. In: Proc. of the Annual Computer Security Applications Conference (ACSAC'09), Honolulu, Hawaii (2009)

[23] Gu, G., Zhang, J., Lee, W.: BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In: Proc. of the 15th Annual Network and Distributed System Security Symposium (NDSS'08) (2008)

[24] Ha, D.T., Yan, G., Eidenbenz, S., Ngo, H.Q.: On the Effectiveness of Structural Detection and Defense Against P2P-based Botnets. In: Proc. of the 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '09), Estoril, Lisbon, Portugal (2009)

[25] Han, Q., Yu, W., Zhang, Y., Zhao, Z.: Modeling and evaluating of typical advanced peer-to-peer botnet. Performance Evaluation **72**(0), 1 – 15 (2014). DOI http://dx.doi.org/10.1016/j.peva.2013.11.001

[26] Hayes, T.P., Saia, J., Trehan, A.: The forgiving graph: a distributed data structure for low stretch under adversarial attack. Distrib Comput **25**, 261 – 278 (2012)

[27] Holz, T., Steiner, M., Dahl, F., Biersack, E.W., Freiling, F.: Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm. In: Proc. of the 1st Usenix Workshop on Large-scale Exploits and Emergent Threats (LEET), San Francisco, CA, USA (2008)

[28] Jelasity, M., Bilicki, V.: Towards automated detection of peer-to-peer botnets: On the limits of local approaches. In: Proc. of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09), Boston, MA (2009)

[29] Kang, B.B., Chan-Tin, E., Lee, C.P., Tyra, J., Kang, H.J., Nunnery, C., Wadler, Z., Sinclair, G., Hopper, N., Dagon, D., Kim, Y.: Towards Complete Node Enumeration in a Peer-to-Peer Botnet. In: Proc. of the 2009 ACM Symposium on Information, Computer and Communications Security (ASIACCS), Sydney, Australia (2009)

[30] Kanich, C., Levchenko, K., Enright, B., Voelker, G.M., Savage, S.: The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff. In: Proc. of the USENIX Workshop on Large-Scale Exploits and Emergent Threats, San Franciso, CA (2008)

[31] Kolesnikov, O., Dagon, D., Lee, W.: Advanced Polymorphic Worms: Evading IDS by Blending in with Normal Traffic. Tech. rep., Georgia Tech (2004-2005)

[32] Kreibich, C., Kanich, C., Levchenko, K., Enright, B., Voelker, G.M., Paxson, V., Savage, S.: On the Spam Campaign Trail. In: Proc. of the USENIX Workshop on Large-Scale Exploits and Emergent Threats, San Franciso, CA (2008)

[33] Kreibich, C., Kanich, C., Levchenko, K., Enright, B., Voelker, G.M., Paxson, V., Savage, S.: Spamcraft: An Inside Look at Spam Campaign Orchestration. In: Proc. of the USENIX Workshop on Large-Scale Exploits and Emergent Threats, Boston, MA (2009)

[34] Król, D.: Propagation phenomenon in complex networks: Theory and practice. New Generation Computing **32**(3-4), 187–192 (2014)

[35] Li, Z., Goyal, A., Chen, Y., Paxson, V.: Automating Analysis of Large-Scale Botnet Probing Events. In: Proc. of ACM Symposium on Information, Computer and Communications Security (2009)

[36] Liang, J., Naoumov, N., Ross, K.W.: The Index Poisoning Attack in P2P File Sharing Systems. In: Proc. of the Infocom, Barcelona (2006)

[37] Louzada, V.H.P., Daolio, F., Herrmann, H.J., Tomassini, M.: Smart rewiring for network robustness. IMA Journal of Computer Networks **1**, 150C159 (2013)

[38] Lu, W., Tavallaee, M., Ghorbani, A.A.: Automatic discovery of botnet communities on large-scale communication networks. In: Proc. of the 2009 ACM Symposium on Information, Computer and Communications Security (ASIACCS), Sydney, Australia (2009)

[39] Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Proc. of the 1st International Workshop on Peer-to-Peer Systems, pp. 53–65 (2002)

[40] Micro, T.: Taxonomy of Botnet Threats (2006)

[41] Nunnery, C., Sinclair, G., Kang, B.B.: Tumbling Down the Rabbit Hole: Exploring the Idiosyncrasies of Botmaster Systems in a Multi-Tier Botnet Infrastructure. In: Proc. of the 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats, San Jose, CA (2010)

[42] Pitsillidis, A., Levchenko, K., Kreibich, C., Kanich, C., Voelker, G.M., Paxson, V., Weaver, N., Savage, S.: Botnet Judo: Fighting Spam with Itself. In: Proc. of the Network and Diestributed System Security Symposium (NDSS), San Diego, CA (2010)

[43] Porras, P., Saidi, H., Yegneswaran, V.: A Multi-perspective Analysis of the Storm (Peacomm) Worm. Tech. rep., SRI (2007)

[44] Ramachandran, K., Sikdar, B.: Modeling malware propagation in Gnutella type peer-to-peer networks. In: Proc. of the 20th International Parallel and Distributed Processing Symposium (IPDPS '06), Rhodes Island, Greece (2006)

[45] Rossow, C., Andriesse, D., Werner, T., Stone-Gross, B., Plohmann, D., Dietrich, C.J., Bos, H.: SoK: P2PWNEDł Modeling and Evaluating the Resilience of Peer-to-Peer Botnets. In: Proc. of 2013 IEEE Symposium on Security and Privacy (2013)

[46] Ruitenbeek, E.V., Sanders, W.H.: Modeling Peer-to-Peer Botnets. In: Proc. of the 5th International Conference on Quantitative Evaluation of Systems (QEST '08), St Malo, France (2008)

[47] Schneider, C.M., Moreira, A.A., Andrade, J.S., Havlin, S., Herrmann, H.J.: Mitigation of malicious attacks on networks. In: Proc Nat Acad Sci USA 108, p. 3838C3841 (2011)

[48] Singh, K., Guntuku, S.C., Thakur, A., Hota, C.: Big data analytics framework for peer-to-peer botnet detection using random forests. Information Sciences (0) (2014). DOI http://dx.doi.org/10.1016/j.ins.2014.03.066

[49] Spitzner, L.: Honeypots (2003). Http://www.tracking-hackers.com/papers/honeypots.html

[50] Starnberger, G., Kruegel, C., Kirda, E.: Overbot - A botnet protocol based on Kademlia. In: Proc. of the 4th International Conference on Security and Privacy in Communication Networks (SecureComm) (2008)

[51] Steiner, M., En-Najjary, T., Biersack, E.W.: A Global View of KAD. In: Proc. of the ACM Internet Measurement Conf. (IMC), San Diego, USA (2007)

[52] Steiner, M., En-Najjary, T., Biersack, E.W.: Exploiting KAD: possible uses and misuses **37**(5), 65–70 (2007)

[53] Stewart, J.: Inside the Storm: Protocols and Encryption of the Storm Botnet (2008).
http://www.blackhat.com/presentations/bh-usa-08/Stewart/BH_US_08_Stewart_Protocols_of_the_Storm.pdf

[54] Stock, B., Goel, J., Engelberth, M., Freiling, F.C.: Walowdac - Analysis of a Peer-to-Peer Botnet. In: Proc. of the European Conference on Computer Network Defense (EC2ND ’09) (2009)

[55] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, AC. In: Proc. of the ACM SIGCOMM, San Deigo, CA (2001)

[56] Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., Vigna, G.: Your Botnet is My Botnet: Analysis of a Botnet Takeover. In: Proc. of the ACM CCS, Chicago, IL (2010)

[57] Stover, S., Dittrich, D., Hernandez, J., Dietrich, S.: Analysis of the Storm and Nugache Trojans: P2P is here. USENIX ;login: **32**(6), 18–27 (2007)

[58] Stutzbach, D., Rejaie, R.: Improving Lookup Performance over a Widely-Deployed DHT. In: Proc. of the IEEE INFOCOM, Barcelona, Spain (2006)

[59] Thommes, R., Coates, M.: Epidemiological Modelling of Peer-to-Peer Viruses and Pollution. In: Proc. of the IEEE Infocom, Barcelona, Spain (2006)

[60] Villamarín-Salomón, R., Brustoloni, J.C.: Bayesian bot detection based on DNS traffic similarity. In: Proc. of the the 24th Annual ACM Symposium on Applied Computing (SAC ’09), Honolulu, Hawaii (2009)

[61] Vogt, R., Aycock, J., Jacobson, M.: Army of Botnets. In: Proc. of the 2007 Network and Distributed System Security Symposium (NDSS) (2007)

[62] Wang, P., Aslam, B., Zou, C.C.: Peer-to-peer botnets. In: P. Stavroulakis, M. Stamp (eds.) Handbook of Information and Communication Security. Springer (2010)

[63] Wang, P., Sparks, S., Zou, C.C.: An Advanced Hybrid Peer-to-Peer Botnet. In: Proc. of the 1st USENIX Workshop on Hot Topics in Understanding Botnets (HotBots ’07), Cambridge, MA (2007)

[64] Wang, P., Tyra, J., Chan-Tin, E., Malchow, T., Kune, D.F., Hopper, N., Kim, Y.: Attacking the Kad Network. In: Proc. of the 4th international conference on Security and privacy in communication netowrks (SecureComm ’08) (2008)

[65] Wang, P., Wu, L., Aslam, B., Zou, C.C.: A Systematic Study on Peer-to-Peer Botnets. In: Proc. of International Conference on Computer Communications and Networks (ICCCN) (2009)

[66] Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C., Kirda, E.: Automatically Generating Models for Botnet Detection. In: Proc. of the 14th European Symposium on Research in Computer Security (ESORICS), Saint Malo, France (2009)

[67] Xie, L., Zhu, S.: A Feasibility Study on Defending Against Ultra-Fast Topological Worms. In: Proc. of The 7th IEEE International Conference on Peer-to-Peer Computing (P2P'07), Galway, Ireland (2007)

[68] Yu, W., Boyer, P.C., Chellappan, S., Xuan, D.: Peer-to-Peer System-based Active Worm Attacks: Modeling and Analysis. In: Proc. of the IEEE International Conference on Communications (ICC) (2005)

[69] Zhou, L., Zhang, L., McSherry, F., Immorlica, N., Costa, M., Chien, S.: A First Look at Peer-to-Peer Worms: Threats and Defenses. In: Proc. of the 4th International Workshop on Peer-To-Peer Systems (IPTPS '05) (2005)

[70] Zhu, Z., Lu, G., Chen, Y., Fu, Z.J., Roberts, P., Han, K.: Botnet Research Survey. In: Proc. of the 32nd Annual IEEE International Computer Software and Applications (COMPSAC '08) (2008)