Chapter 1

Peer-to-Peer Botnets

Ping Wang, Baber Aslam, Cliff C. Zou
School of Electrical Engineering and Computer Science,
University of Central Florida, Orlando, Florida 32816

"Botnet" is a network of computers that are compromised and controlled by an attacker. 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. Without C&C servers, P2P botnets are more resilient to defense countermeasures than traditional centralized botnets. In this chapter, we systematically study P2P botnets along multiple dimensions: botnet construction, command and control mechanisms, performance measurements, and mitigation approaches.

1.1 Introduction

"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 [1]. These zombie computers are remotely controlled by an attacker, so-called "botmaster". Botnets with a large number of computers have enormous cumulative bandwidth and powerful computing capability. They are exploited by botmasters for initiating various malicious activities, such as email spam, distributed denial-of-service (DDOS) 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. Among them, Internet relay chat (IRC)-based botnets [55] are the most popular ones, which use IRC [33] to facilitate command and control (C&C) communication between bots and botmasters. In a centralized botnet as shown in Fig. 1.1, bots are connected to one or several servers to obtain commands. This architecture is easy to construct and very efficient in distributing botmaster's commands; however, it has a single point of failure - the C&C server. Shutting down the IRC server would cause all the bots lose contact with their botmaster. In addition, defenders can also easily monitor the botnet by creating a decoy to join in the specified IRC channel.

Recently, peer-to-peer (P2P) botnets , such as Trojan.Peacomm botnet [27] and Stormnet [32], have emerged as attackers gradually realize the limitation of traditional centralized botnets. Just like P2P networks, which are resilient to dynamic churn (i.e., peers join and leave the system at high rates [36]), P2P botnet communication won't be disrupted when losing a number of bots. In a P2P botnet as shown in Fig. 1.2, there is no centralized server, and bots are connected to each other topologically and act as both C&C server and client. P2P botnets have shown the advantages over traditional centralized botnets. As the next generation of botnets, P2P botnets are more robust and difficult for security community to defend.

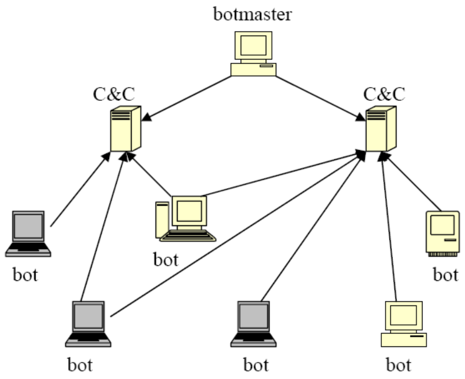


Figure 1.1: Centralized Botnet

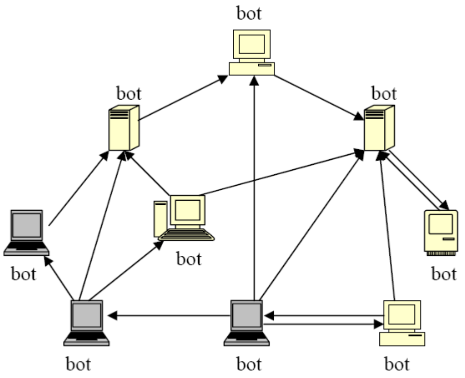


Figure 1.2: P2P Botnet

Researchers have started to pay attention to P2P botnets. Grizzard et al., and Holz et al., dissected Trojan.Peacomm botnet [27] and Stormnet [32] in detail respectively. However, in order to effectively fight against this new form of botnets, enumerating every individual P2P botnet we have seen in the wild is not enough. Instead, we need to study P2P botnets in a systematic way. Therefore in this 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. We hope to shed light on P2P botnets, and help researchers and security professionals be well prepared and develop effective defenses against P2P botnet attacks.

Our contributions are the following:

- We present a systematic comparison and analysis of P2P botnets. We focus on two fundamental aspects of botnets: 1) botnet construction, and 2) botnet C&C mechanisms.
- We propose metrics for measuring performance of P2P botnets.
- We present a number of countermeasures to defend against P2P botnets.
- We obtain one counter-intuitive finding: compared to traditional centralized botnets, by using "index poisoning" technique (detailed discussed in Sect. 1.6.3), it is easier to shut down or at least effectively mitigate P2P botnets that adopt existing P2P protocols and rely on file index to disseminate commands.

The remainder of the chapter is organized as follows. Section 1.2 introduces background knowledge on P2P networks. Construction of P2P botnets is discussed in detail in Sect. 1.3, followed by the design of botnet command and control mechanisms in Sect. 1.4. We present performance metrics for P2P botnets in Sect. 1.5. Section 1.6 discusses possible countermeasures against P2P botnets. Related work are presented in Sect. 1.7 and finally we conclude this chapter in Sect. 1.8.

1.2 Background on Peer-to-Peer Networks

A P2P network is a computer network in which two or more computers are connected and share resources (content, storage, CPU cycles) by direct exchange, rather than going to a server or authority which manages centralized resources [19]. Today one of the major uses of P2P networks is file-sharing, and there are various P2P file-sharing applications, such as eMule [15], Lime Wire [17], BitTorrent [14]. P2P networks can be distinguished in terms of their centralization and structure [19].

Speaking of overlay network [2] centralization, there are two categories (Table 1.1).

Centralized Architectures In this class of P2P networks, there is a central server which maintains directories of metadata and routing information, processes file search requests and coordinates download among peers. Napster [3] was the first widely-used P2P sharing application, which used this centralized architecture. It had a central site, which receives search, browse or file transfer requests sent by peers, and sends responses back to them. But the central site does not participate in actual file downloading. Obviously this central site is a single point of failure and limits network scalability and robustness.

Decentralized Architectures In this class of P2P networks there is no central server. All requests are handled by peers within the network. In purely decentralized architectures, each peer behaviors exactly the same. They act as a server when processing a file search query, and as a client when requesting a file. However in partially decentralized architectures, peers with better computing ability and network bandwidth have the chance to be promoted and become "super peers" playing a more important role in the network. Gnutella network [16] is a partially decentralized P2P network. Normal peers (so- called leaf peers) in Gnutella network is connected to at least one super peer (so-called ultrapeer) and can only send queries to its ultrapeers. An ultrapeer maintains a table of hash values of files which are available in its local leaf peers. Only ultrapeers can forward messages. Therefore, from the leaf peer's perspective, an ultrapeer acts as a local server.

Table 1.1: Classification based on Centralization

Category		**Network or Protocol**	**Applications**
Centralized		Napster	Napster
Decentralized	Purely	Gnutella (before 2001)	LimeWire
	Partially	Gnutella (after 2001) FastTrack	LimeWire Kazaa

Table 1.2: Classification based on Structure

Category	**Network or Protocol**	**Applications**
Structured	Overnet	eDonkey2000, eMule
Unstructured	Gnutella	LimeWire

P2P networks can also be classified as either unstructured or structured networks (Table 1.2).

Unstructured In unstructured P2P networks, content location is completely unrelated to the network topology. A query usually flooded throughout the network, or forwarded in depth-first or breadth-first manner, until a query hit is reached or queries expire. Gnutella network is unstructured. Ultrapeers forward queries to its neighboring ultrapeers, and the query hit will be sent back along the same route as the search query.

Structured In structured P2P networks, a mapping is created between the content (e.g., file identifier) and its location (e.g., node address). A distributed hash table (DHT) for routing is usually implemented in this type of network. CAN, Chord, Pastry, and Tapestry are the first four DHTs introduced in academia [4]. Kademlia [38] is another DHT algorithm, which has been used in Overnet [5], eMule [15] and BitTorrent [14].

1.3 P2P Botnet Construction

We can consider P2P botnet construction as a two-step process. Step one, an attacker needs to compromise many computers in the Internet, so that she can control them remotely. All kinds of malicious software, such as worms, trojan houses, viruses, instance message (IM) malware, can be used to accomplish this task. There are many researches on these malware so they are not the focus of this chapter. Step two, further actions should be taken by these compromised computers to form a botnet. Depending on the scope that the attacker is targeting, the second step could be slightly different. We will discuss P2P botnet construction in detail from two aspects: 1) how to choose and compromise bot candidates; 2) how to form a botnet.

1.3.1 Selection of Bot Candidates

P2P networks are gaining popularity of distributed applications, such as file-sharing, web caching, network storage [21]. In these content trading P2P networks, without a centralized authority it is not easy to guarantee that the file exchanged is not malicious. For this reason, these networks become the ideal venue for malicious software to spread. It is straightforward for attackers to target hosts in existing P2P networks and build their zombie army of botnets.

Many P2P malware have been reported, such as Gnuman [6], VBS.Gnutella [6] and SdDrop [7]. Take P2P worms for example, they can be categorized as either active, such as topological worms, or passive, such as file-sharing worms. A peer which is compromised by an active P2P worm will try to infect other peers in its "hit-list" rather than looking for vulnerable ones blindly through random scans. Peers in the hit-list could be those who have been contacted before, or those who have responded after a file search query. Passive P2P worms duplicate themselves and reside in the local file sharing directory as files with popular names, and expect other peers to download, execute them and get infected. Other P2P malware (e.g., viruses and trojans) propagates in the similar way as passive worms.

Once a vulnerable host in a P2P network has been compromised by botnet malware, it can directly become a bot member without any further joining botnet action. This is because the botnet itself resides within the current P2P network. Bots can find and communicate with each other by simply using current P2P protocol. Up to this point, the botnet construction is done and this botnet is ready to operate by botmaster.

The above discussion shows that it is convenient and simple to build a P2P botnet, if all the bot candidates are chosen from an existing P2P network. For the convenience of further discussion, we call such a botnet a "*parasite P2P botnet*".

However, the scale of a parasite botnet is limited by the number of vulnerable hosts in an existing P2P network, which is not flexible enough and greatly reduces the number of potential bot candidates for botmasters. Therefore P2P botnets we have witnessed recently do not confine themselves to existing P2P networks, but recruit members in the entire Internet through all possible spread mediums like email, instance message and file exchanging. For this type of P2P botnets, upon infection, the most important thing is to let newly compromised computers join in the network and connect with other bots, regardless of whether the connection is direct or indirect. Otherwise, they are just isolated individual computers without much use for botmasters. How to let infected hosts form a botnet is discussed in the following section.

1.3.2 Forming A Botnet

Like what we mentioned above, if all the potential bot candidates are already within a P2P network, it is not necessary to perform any further action to form the botnet. However, if a random host is compromised, it has to know how to find and join the botnet. As we know current P2P file-sharing networks provide the following two general ways for new peers to join a network:

1. An initial list of peers are hard-coded in each P2P client. When a new peer is up, it will try to contact each peer in that initial list to update its neighboring peer information.

2. There is a shared web cache, such as Gnutella web cache [8], stored at some place on the Internet, and the location of the cache is put in the client code. Thus new peers 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 adapted 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 [27] is a piece of malware to create a P2P botnet which uses the Overnet P2P protocol for C&C communication. A list of Overnet nodes which are likely to be online is hard-coded into the bot's installation binary. When a victim is compromised and run a Trojan.Peacomm, it will try to contact peers in this list to bootstrap onto the Overnet network. Another P2P botnet, Stormnet [32] uses the similar bootstrap mechanism: the information about other peers with which the 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.

However, bootstrap is a vulnerable procedure and it could become a single point of failure for botnet construction. If the initial list of peers or web cache is obtained by defenders, they may be able to prevent botnet from growing simply by shutting down those bootstrap peers or web caches. From this perspective, we can see that although a parasite P2P botnet has limitations on bot candidate selection, it does not have the bootstrap vulnerability by choosing bot candidates from an existing P2P network.

To overcome this vulnerability, botnet attackers may think of other ways to avoid introducing bootstrap procedure in P2P botnet construction. For example, in the hybrid P2P botnet introduced in [50], when a bot A compromises a vulnerable host B, A passes its own peer list to this newly infected host B, and B will add A into this neighboring peer list. Any two bots that have found each other (e.g., through Internet scanning) will exchange their peer lists to construct new lists. In this way, the P2P botnet avoid bootstrap procedure relying on hard-coded lists. In [49] a similar procedure was presented to construct a super botnet instead of bootstrapping.

1.3.3 Comparison

Considering bot member selection and the network that a botnet participates in, P2P botnet can be classified into three categories: *parasite P2P botnet* (introduced in Sect. 1.3.1), which refers to a botnet that only targets vulnerable hosts in an existing P2P network (e.g., Gnutella network); *leeching P2P botnet*, in which bots are chosen from vulnerable hosts throughout the Internet, but eventually they will participate in and rely on an existing P2P network; *bot-only P2P botnet*, such as Stormnet [32], which refers to a botnet that resides in an independent network, and there are no benign peers except bots.

Since parasite and leeching P2P botnets are both built upon existing P2P networks, they usually directly employ the protocols of those networks for C&C communication (Sect. 1.4). But as we discussed in Sect. 1.3.2, bootstrap is required during construction process for leeching P2P botnets, while it is not needed for parasite P2P botnets. Bot-only P2P botnets are the most flexible ones among these three types of P2P botnets. Their botmasters can design a new C&C protocol or use an existing P2P protocol, and bootstrap is optional.

1.4 P2P Botnet Command & Control Mechanisms

Botnet C&C mechanism is the major part of a botnet design. It directly determines the topology of a botnet; and hence affects the robustness of a botnet against network/computer failures, security monitoring and defenses.

Traditional botnets, like IRC-based botnets, are referred to as centralized botnets since they have a few central servers where all bots connect to and retrieve commands from. P2P botnets mean that their C&C models are P2P-based, i.e. no central server is used. Each bot member acts as both a command distribution server and a client who receives commands. This explains why P2P botnets are generally more resilient against defenses than traditional centralized botnets.

The C&C mechanisms can be categorized as either *pull* or *push* mechanism. Pull mechanism, also called "command publishing/subscribing", refers to the manner that bots retrieve commands actively from a place where botmasters publish commands. On the contrary, push mechanism means bot members passively wait for commands to come and then they will forward commands to other bots.

For centralized botnets, pull mechanism is commonly used. Take botnets based on HTTP as an example, normally a botmaster publishes commands on a web page, and bots periodically visit this web page via HTTP to check for any command updates. The address of this web page comes with the bot code and can be changed afterwards by issuing an address changing command. IRC-based botnets is another case of pull C&C mechanism: all bots periodically connect to a pre-determined IRC channel, waiting for their botmaster to issue a command in this channel.

As we discussed in Sect. 1.3, in a leeching P2P botnet or a parasite P2P botnet, bots are mixed with normal P2P users, and they can communicate with each other using corresponding P2P protocol. Thus it is natural to leverage the existing P2P protocols for C&C communication. On the other hand, bot-only P2P botnets are not confined to any current P2P protocols. Botmasters have the flexibility to either adopt existing P2P protocols (such as Trojan.Peacomm botnet) or design a new communication protocol (such as the hybrid P2P botnet in [50]). In the following, we will discuss how pull and push C&C mechanisms can be applied in P2P botnets.

1.4.1 Leveraging Existing P2P Protocols

P2P networks are widely used as content exchange applications. New ideas have been proposed to solve problems or improve current P2P protocols in many aspects, such as reducing network traffic, improving communication efficiency, mitigating network churn problem. For example, current version of Gnutella network is more scalable than the original one by changing the network structure from pure decentralized to hybrid decentralized where some peers (ultrapeers) with stable conductivities and high bandwidth are considered more important than others. Therefore, it is attractive for botmasters to directly implement current P2P protocols into their P2P botnets. Next we will discuss the feasibility of adopting existing P2P protocols for P2P botnet C&C communication.

Pull Mechanism - Command Publishing/Subscribing

In P2P file-sharing systems, a peer sends out a query looking for a file. The query message will be passed around in the network according to an application-dependent routing protocol. If a peer who has the searched file receives the query, it will respond with a query hit message to the peer who initiates the query. It is easy to adopt this idea and use it for botnet C&C communication.

For a parasite P2P botnet, a botmaster can randomly choose one or more bots to publish a command, just like letting a normal peer declare that there is a specific file available on it. The title of this file needs to be predetermined or can be calculated using an algorithm which is hard-coded in bot code, such that the other bots know what file to query in order to retrieve the command. Once a query for this specific file reaches a bot possessing the command, a query hit will be sent back to the requesting bot. The command can be directly encoded in the query hit message, or the query hit only includes the address of a place where the command was published. In the latter case, when the requesting bot gets the query hit message, it will go to that place to fetch the command. Here, special crafted query messages are used to retrieve commands by bots instead of searching for real files.

In a centralized P2P Network, such as Napster, there is a central server, which maintains the whole file directory and provides support for file search requests and download requests. If P2P botnet C&C communication relies on this protocol, the central server will be the only place where bots sends queries for retrieving commands. Obviously this C&C model is centralized. Such a botnet degrades to a centralized botnet. Thus in the following discussion, we will not consider this type of P2P architecture.

There are two types of decentralized P2P networks: unstructured and structured, as introduced in Sect. 1.2. Gnutella network is a two-tiered unstructured P2P network [46]. In Gnutella network, only ultrapeers can forward queries and each ultrapeer maintains a directory of files shared by its leaf nodes. When a query comes to an ultrapeer, the ultrapeer will forward the query to all its neighboring ultrapeers and its leaf peers that may have the file. Overnet, on the other hand, is a DHT-based structured P2P network, where a query for the hash value of a file is only sent to the peers whose IDs have a closer distance to the file's hash value.

Currently, we have not witnessed P2P botnets on unstructured P2P networks yet. However P2P botnets have emerged on DHT-based structured P2P networks, such as Trojan.Peacomm botnet [27] and Stormnet [32]. The C&C mechanisms of these two botnets are similar. They both use the standard Overnet protocol for controlling their bot members. Each bot periodically queries a search key, which is calculated by a built-in algorithm. The algorithm takes the current date and a random number from [0-31] as input to calculate the search key. In this way, when issuing a command, the botmaster needs to publish it under 32 different keys.

This command publishing/subscribing C&C mechanism implemented in Trojan.Peacomm botnet and Stormnet, however, may not provide as strong resilience against defenses as botmasters thought. With a copy of captured bot code, it is not hard for defenders to either figure out the query generation algorithm, or observe and predict bot queries. This C&C design makes it possible for defenders to monitor or disrupt a small set of botnet control communication channels. We will provide more detailed discussion on this issue in Sect. 1.6.3.

Push Mechanism - Command Forwarding

Push mechanism means when a botmaster issues a command to some bots, and these bots will actively forward the command to others. In this way, bots can avoid periodically requesting or checking for new command, and hence, reduce the risk of being detected. There are two major design issues for this mechanism:

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

To address the first issue, the simplest way is to let a bot use its current neighboring peers as targets. But the weakness of this approach is that command distribution may be slow or sometimes disrupted, since some bots have a small number of neighbors. One solution might be a bot could initiate a search for a file, and use the peers who respond to the query as the command forwarding targets.

In a parasite P2P botnet or a leeching P2P botnet, since not all members in the P2P network belong to the botnet, some peers in the neighboring list may not be bot members. Thus it is possible that a command is not forwarded to any bot. To solve this issue, botmaster could design some strategies to increase the chance that the command hits an actual bot. For example, after a computer is compromised and becomes a bot, it can claim that it has some popular files available. When a bot is trying to forward a command, it can initiate searches for these popular files, and forward the command to those peers appearing in the search result. This predefined set of popular files behave as the watchwords for the botnet. This approach increases the command dispersion opportunity, but could give defenders a clue to identify bots.

For the second issue, using in-band message or out-of-band message to forward a command depends on what the peers in the target list are. If a bot just targets its neighboring peers, in-band message would be a good choice. A bot could treat a command as a normal query message and send it to all its neighboring peers, and rely on these neighboring peers to continue passing on the command in the botnet. The message would seem as a normal query message to benign peers, but it can be interpreted as a command by bot members. This scheme is easy to implement and hard for defenders to detect, because the command forwarding traffic is mixed with normal search query traffic. On the other hand, if the target list is generated in some other ways, like the previously discussed approaches based on file search results, a bot has to contact those peers using out-of-band message: the bot contacts target peers directly, and encodes the command in a secrete channel which can only be decoded by a bot member. Obviously out-of-band traffic are easier to detect, and hence, can disclose the identities of bots who initiate such traffic. Therefore, as we can see, in botnet design, botmasters will always face the tradeoff between efficiency and detectability of their botnets.

The above discussion mainly focused on unstructured P2P networks, where the query messages are flooded to the network. In structured P2P networks (e.g., Overnet), a query message is forwarded to the nodes whose node IDs are closer to the queried hash value of a file, which means a query for the same hash value is always forwarded by the same set of nodes. Therefore, it would be more efficient that a bot can generate different hash keys associated with the same command. In this way, a single command can be forwarded to different parts of the network, letting more nodes receive the command search query and obtaining the command.

1.4.2 Design A New Communication Protocol

It is convenient and straightforward to adopt existing P2P protocols for P2P botnet C&C communication. However, while taking advantages of nice properties of those protocols, the inherited drawbacks may limit botnet design and performance. On the contrary, a botnet is more flexible if it uses a new communication protocol designed by its botmaster. In this section, we give two examples of P2P botnet C&C communication protocols which are not dependent on existing P2P protocols.

As we mentioned in Sect. 1.3, if a botnet depends on an existing P2P protocols for C&C communication, in most cases, bootstrap procedure is required. The hybrid P2P botnets proposed in [50] effectively avoid bootstrap 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. In addition, to better balance the connectivity among bots, botmaster could ask bots to renew their peer lists from sensors. In the hybrid P2P botnets, both push and pull mechanisms are used. When a bot receives a command, it will try to forward it to all the peers in the list (push mechanism), and for those who cannot accept connection from others, such as bots either with private IP addresses or are behind a firewall, they will periodically contact other bots in the list and try to retrieve new commands (pull mechanism).

The army of botnets, i.e. super botnet, proposed in [49] also implements both pull and push mechanisms in its communication protocol. A super botnet is composed of a number of small centralized botnets. Each C&C server in a small botnet has routing information to a certain number of other C&C servers. When a C&C server receives a command from the botmaster, it will push the command to the C&C servers which appear in its routing table. Meanwhile, bots in a small botnet will pull the command from their C&C server periodically.

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, it may be disabled by unexpected problems.

1.5 Measuring P2P Botnets

Besides botnet propagation and communication schemes, botnet performance is another important issue cared by both botmasters and defenders. In this section, we discuss performance measurements for P2P botnets along three dimensions [22]:

- Effectiveness — how powerful a P2P botnet can be when launching an attack.
- Efficiency — how long it would take for majority members of a P2P botnet to get informed after a command is issued.
- Robustness — how resilient a botnets is to failures in the network, such as bots being removed by defenders.

In addition, we present available statistics related to the metrics presented below on current Gnutella P2P networks.

1.5.1 Effectiveness

Bots take orders from their botmaster to launch malicious attacks, such as DDOS attack, spam and phishing. Usually the more bots participate in an attack, the more damage a botnet would cause. Therefore botnet size is a key metric to estimate the effectiveness of a botnet. As mentioned in [40], botnet size is not a clearly defined term. Some people refer it to the number of concurrent online bots, and some refer it to the total population of a botnet. Both definitions can show the capability of a botnet.

Gnutella crawler Cruiser, developed by Daniel Stutzbach and Reza Rejaie [45], is able to capture a snapshot of top-level overlay including only ultrapeers and legacy peers of the Gnutella network within several minutes. In the snapshot captured on 05/27/08 by Cruiser, the total number of peers that has been successfully crawled is around 450,000 (there were around 700,000 peers the crawler tried to contact, but failed on almost 40% of the peers due to peers being offline, or lost connection, etc.). We can estimate the total size of the crawled Gnutella network, which is $450,000 \times n$, where n is the average number of leave peers each ultrapeer connects to. The size of this network is overestimated based on this method, because a leaf peer may connect to multiple ultrapeers. Nevertheless, this number would be the upper bound of the botnet size, if a parasite P2P botnet was built upon this Gnutella network during the crawling time period.

Overnet, eDonkey2000 and Storm botnet [32] are all DHT-based and utilize the same algorithm — Kademlia. The first two networks have been taken down in 2005 by RIAA because of copyright issues, so it is not possible to obtain the network snapshots and estimate the scale of the network. However, for Storm botnet, there exits estimates of the number of computers infected by the

Storm malware, which vary widely and range from a few hundred thousand to 10 million [35]. Researchers have estimated concurrent online Storm nodes to be between 5,000 and 40,000 [31].

Except parasite P2P botnets whose size is limited by the size of existing P2P networks. The scale of both leeching P2P botnets and bot-only P2P botnets (such as the proposed super botnet [49] and hybrid botnet [50]), depends on when the botmasters stop the construction of their botnets and the population of vulnerable computers in the Internet.

1.5.2 Efficiency

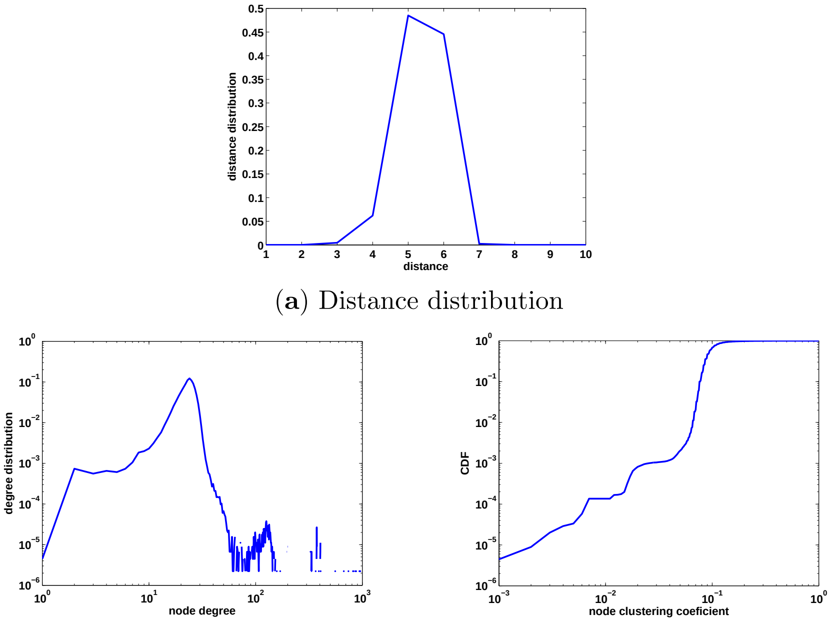
Botnet efficiency means how fast the majority members of a botnet can receive a command after it has been issued by the botmaster. In centralized botnets command delivery is guaranteed by C&C server, while for P2P botnets, the efficiency is affected by several factors.

If a P2P botnet is built upon a network where there is no mapping between a message and where it should go, then the distance between a pair of peers is a good measure for efficiency. Unstructured P2P botnets (such as Gnutella based botnets), the proposed super botnet [49] and hybrid botnet [50] all belong to this category.

Take Gnutella based P2P botnets for instance, we consider the distance between two ultrapeers, since only ultrapeers can forward messages, and denote $D(x, y)$ as the distance between two ultrapeers x and y. In Gnutella network, each search query has a limited lifetime which is defined as the number of hops it can travel, so the distance between two ultrapeers of two bots determines whether or not a command can be successfully received by a bot. By analyzing the snapshot of top-level overlay (the topology is treated as an undirected graph) of Gnutella network obtained by Cruiser [45], we find the distance between most of pairs of ultrapeers are around 5 (shown in Fig. 1.3(**a**)), which is smaller or equal to the threshold set by many Gnutella clients for discarding a message. For example, LimeWire, one of the most popular Gnutella clients, sets the threshold as 7. This means after a botmaster issues a command, most of bots within the network is able to receive the command in a short time.

Moreover, *betweenness* is also relevant measure for efficiency of this type of P2P botnets. Betweenness of a node i, b_i [9], is defined as the total number of shortest paths between pairs of nodes that pass through node i . It indicates the capability of a node to control the flow of traffic between node pairs. If commands are first issued at nodes with high betweenness, they can be passed around in less amount of time. On the contrary, removal of those nodes may result in great increase of distance between others, and reduce command delivery efficiency.

For structured P2P network, the efficiency of message delivery is determined by DHT algorithm it implements. For example, as shown in [38], in Overnet most of query operations take $\lceil \log n \rceil + c$ time, where c is a small constant and n is the size of the network. The probability of finding a key for a $\langle key, value \rangle$ lookup is relatively high.



(**a**) Distance distribution

(**b**) Node degree distribution

(**c**) CCDF of node clustering

Figure 1.3: Statistics of top-level overlay of Gnutella network

1.5.3 Robustness

It is inevitable that nodes in a botnet are not available for some reasons, such as network failures, being turned off by users or defenders, etc. How resilient a botnet is to these situations is important for both botmasters and defenders. *Degree distribution* and *clustering coefficient* are the good measures to express the network robustness.

The degree of a node is the number of edges incident to the node, and the degree distribution is the probability distribution of these degrees over the whole network [10]. The degree distribution expresses how balanced connections to peers in a network are. In [50], it has been shown that if connections to servants bots is more balanced, the hybrid P2P botnet is more resilient.

Clustering coefficient is a measure of how well the neighbors of a given node are locally interconnected. Clustering coefficient of a node c_i [11] is defined as the ratio between the number of edges N_i among the neighbors of a node i of degree k_i and the maximum number of possible edges $k_i(k_i + 1)/2$.

Degree distribution and cumulative distribution function (CDF) of node clustering coefficient of top-level overlay of Gnutella network are shown in Fig. 1.3(**b**) and Fig. 1.3(**c**), respectively. From Fig. 1.3(**b**) we observe most of the node degrees are between 10 and 100 due to the connection limit set by Gnutella clients. This implies Gnutella network has well balanced connectivity. On the other hand, the CDF of node clustering coefficient in Fig. 1.3(**c**) shows neighborhood of a node is not well interconnected. Therefore such a P2P network is vulnerable to random node removal, but not to targeted node removal (discussed in Sect. 1.6.3).

1.6 Countermeasures

Researchers and security professionals have developed many defense techniques for Internet malware. In this section, we will discuss how those techniques can be used to mitigate P2P botnet attacks, and also present several new defense ideas.

1.6.1 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 [34], make it possible for malware to evade such detection systems. Therefore, instead of doing static analysis, defenders start considering dynamic information for detection. In [29] Gu et al., correlated the feedback given by three intrusion detection systems, and tried to determine whether a match with the predefined malware propagation process happens, which indicates a possible malware infection.

Distributed detection systems have been proposed to detect botnet propagation behaviors. In [53] Zhou et al., introduced a self-defense infrastructure inside a P2P network, where a set of guardian nodes are set up. Once the guardians detect worms, they will send out alerts to other nodes, and encourage them to take actions to protect themselves. Xie et al., presented two approaches in [51] to fight against ultra-fast topological worms. The first approach utilizes immunized hosts to proactively stop worm spread, and another approach utilizes a group of dominating nodes in the overlay to achieve fast patch dissemination in a race with the worm. In such distributed systems, the nodes chosen to perform detection functionality need to be closely monitored. Failure to protect them from being compromised may corrupt the whole defense system.

Moreover, anomaly detection is another way to detect and prevent botnet attacks. Gianvecchio et al., used pattern analysis methods to classify human and bots in Internet Chat [26]. Similarly, in a parasite or leeching P2P botnet, bot machines may be distinguished from legitimate P2P user machines according to their behavior patterns. For example, a bot peer usually exhibits behaviors such as sending queries periodically, always querying for the same content, or repeatedly querying but never downloading. These behaviors could be considered as abnormal and be used as detection criteria.

1.6.2 Monitoring

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

Sensors

For parasite botnets and leeching botnets where there are legitimate peers in the P2P networks, we can rely on sensors for botnet monitoring. Sensors are chosen from legitimate peers. Usually they are peers that play more important roles in the network communication such that more information can be collected. For example, in Gnutella networks, only ultrapeers are allowed to pass queries, and a leaf peer will receive queries only when its ultrapeers thinks it has the queried content. It is obvious that ultrapeers can monitor more P2P traffic than leaf peers, and hence, ultrapeers are the candidates for sensors, especially those with high connectivities.

In DHT-based P2P networks, it is a little bit different. Peers in such a network are considered equal according to their roles. What queries a node may get is determined by its ID. Therefore, sensors should be selected from a set of peers whose IDs are evenly distributed in the DHT. In this way, we can capture more queries in the network.

Honeypots

Honeypot and honeynet techniques [12] are widely used for botnet monitoring. Take IRC-based botnets for instance, a compromised honeypot can join a IRC C&C channel and collect important botnet information, such as commands issued by botmasters, identities of bots connected to the same channel.

Facing this threat from defense community, attackers have developed ways to detect honeypots [18, 13]. At the same time, defenders also try to better disguise honeypots and deceive honeypot-aware malware. For example, the "double-honeypot" system presented by Tang and Chen [47] can fool the two-stage reconnaissance worm [56].

1.6.3 Shutdown

The ultimate purpose of studying botnets is to shut them down. Intuitively, we can either 1) remove discovered bots, or 2) prevent bot members from receiving commands issued by their botmaster.

Physically Shutting down P2P Bots

Botnet construction relying on bootstrapping is vulnerable during its early stage. Isolating or shutting down bootstrap servers or the bots in the initial list that is 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 at least partially disabled by removing detected bot members, especially bots that are important for the botnet C&C communication. There are two modes of bot removal: random removal and targeted removal. Random removal of bots means disinfecting the host whenever it is identified as a bot. Targeted removal means removing critical bots when we have the knowledge of the topology or the C&C architecture of a P2P botnet.

To evaluate the effectiveness of targeted removal, Wang et al., proposed two metrics in [50], $C(p)$ is the "connected ratio" and $D(p)$ is the "degree ratio" after removing top p fraction of mostly-connected bots in a P2P botnet. They are defined as:

$$C(p) = \frac{\text{\# of bots in the largest connected graph}}{\text{\# of remaining bots}} \tag{1.1}$$

$$D(p) = \frac{\text{Average degree of the largest connected graph}}{\text{Average degree of the original botnet}} \tag{1.2}$$

The metric $C(p)$ shows how well a botnet survives a defense action; the metric $D(p)$ exhibits how densely the remaining botnet is connected together after bot removal.

Shutting down botnet C&C channel

Shutting down each detected bot is slow in disabling a botnet and sometime impossible to do (e.g., you have no control of an infected machine abroad). So a more effective and feasible way is to interrupt a botnet's C&C communication such that bots cannot receive any order from their botmaster. This defense approach has been carried out well for IRC-based botnets through shutting down the IRC C&C channels or servers, but is generally believed to be much more difficult to do for P2P botnets since there is no centralized C&C servers.

However, we find that this general understanding of "*P2P botnet is much more robust against defense*" is misleading. In fact, P2P botnets that rely on publishing/subscribing mode for C&C communication is as vulnerable as traditional centralized botnets against defense. The reason is because the following "index poisoning attack".

Index Poisoning Attack

Index poisoning attack was originally introduced by companies or organizations who are trying to protect copyrighted content from being distributed illegally by using P2P network [37]. Peers can target a set of files and insert massive number of bogus records. when a peer search for a targeted file, the poisoned index returns bogus result, such that the peer cannot locate the file or download a bad one.

Index poisoning can be used to mitigate P2P botnets. If a P2P botnet adopts an existing P2P protocol and employs pull-based C&C communication mechanism, file index is the media for communications between bots and botmasters (Sect. 1.4). If defenders know the specific values that bots will query for getting commands, then they can try to overwrite corresponding records in file index with false information.

For instance, in P2P botnets Peacomm.Trojan [27] and Stormnet [32], each day there are fixed 32 hash values that may be queried by bots for retrieving commands. By analyzing the bot code or monitoring the bot behavior with the assistance of honeypots, defenders are able to generate or observe these specific hash values. And they can publish bogus information under these hashes. In this way, a bot has little chance to get the correct command and lose contact with its botmaster. Therefore poisoning the indices related to these hash values can interrupt the botnet C&C communication effectively.

We can see that publishing/subscribing based P2P botnets have this fundamental vulnerability against index poisoning attacks. This vulnerability is due to two reasons:

- Any peer in a P2P network can insert or rewrite records in file index without any authentication;

- The P2P botnet uses a limited number of predefined hash values for command communication and these values can be figured out by defenders.

Starnberger et al., provided a method to overcome this index poisoning attack by dynamically changing each bot's command query messages [44]. This method makes defenders unable to predict the content of command query messages. However, to realize this function the proposed botnet needs to set up additional sensors in the P2P network and carry out related bot information collections.

P2P botnets that do not rely on publishing/subscribing C&C mechanisms, such as the hybrid botnet [50] and the super botnet [49], will not be affected by this index poisoning attack.

Sybil Attack

Sybil attack was first discussed in [25]. The way it works is to forge identities to subvert the reputation system in P2P networks. But it can also be turned to a mitigation strategy for fighting against P2P botnets.

In [24], Davis et al., tried to infiltrate a botnet with a large number of fake nodes (sybils). Sybil nodes that are inserted in the peer list of bots can disrupt botnet C&C communication by re-routing or stopping C&C traffic going though them.

The important difference between index poisoning attack and sybil attack is that sybil nodes must remain active and participate in the underlying P2P protocols, in order to remain in the peer list of bot nodes. However, nodes used for index poisoning do not have to stay online all the time. They only need to refresh some specific index records with bogus information periodically.

Blacklisting

Like the DNS blacklist approach used to fight again email spam [41], we can also use the same idea to defend against P2P botnets. A query blacklist which holds query related to botnet command, or a peer blacklist which holds identified bots, can be maintained in a P2P network. So queries appeared in the former list will be discarded by legitimate peers, and messages from or to peers in the latter list will not be processed.

1.6.4 Others

Instead of protecting P2P networks and defending P2P botnets from the outside, we could try to secure existing P2P protocols themselves. For example, what content may contain in a returned query result depends on the protocol itself. If extra information is allowed in the result, it could be abused by botmaster to spread commands. For example, as mentioned in [44], keeping bogus information away from DHT, i.e. trying to let the DHT contain as small information as possible, could make DHT-based P2P networks more secure.

1.7 Related Work

P2P botnets, as a new form of botnet, have appeared in recent years and obtained people's attention. In [27] 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 [32]. 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 [38], which is a DHT routing protocol designed for decentralized P2P networks. And a botnet protocol which is also based on Kademlia was proposed by Starnberger et al. [44]. Moreover, to be well prepared for the future, some other botnets whose architecture is similar to P2P architecture, such as an advanced hybrid P2P botnet [50] and super botnet [49], have been presented as well.

There have been some systematic studies on general botnets. Barfor and Yegneswaran [20] 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 [39] proposed a taxonomy of botnet threads. In [22] 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 [54] done by Zhu et al., classified recent research work on botnets into three categories: bot anatomy, wide-area measurement study and botnet modeling and future botnet prediction. 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 didn't discuss in this chapter. In recent work [43], Ruitenbeek and Sanders presented a stochastic model of the creation of a P2P botnet. In [23], 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. [42] 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 [52] and [48] 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.

There have been some works on botnet defense and mitigation. Gu et al., proposed three botnet detection systems: BotMiner [28] — a protocol- and structure-independent botnet detection framework by performing cross cluster correlation on captured communication and malicious traffic, BotSniffer [30] — a system that can identify botnet C&C channels in a local area network without any prior knowledge of signatures or C&C server addresses based on the observation that bots within the same botnet will demonstrate spatial-temporal correlation and similarity and BotHunter [29] — a bot detection system using IDS-Driven Dialog Correlation according to defined bot infection dialog model.

1.8 Conclusion

In this chapter, we have provided a systematical study on P2P botnets, a new generation of botnets from multiple dimensions. First we discussed the how P2P botnet can be constructed. Then we presented two possible C&C

mechanisms and how they can be applied to different types of P2P botnets. A very interesting finding is that, unlike the general understanding that the C&C channels of P2P botnets are harder to shut down than a centralized botnet, a P2P botnet that relies on publishing/subscribing C&C mechanism can be effectively disrupted by index poisoning attacks. In addition, we introduced several metrics to measure the effectiveness, efficiency and robustness of P2P botnets. Finally, we pointed out possible directions for P2P botnet detection and mitigation.

1.9 Acknowledgements

This material is based upon work supported by National Science Foundation under Grant CNS-0627318 and Intel Research Fund. The authors would also like to thank Amir Rasti and Daniel Stutzbach for providing snapshots of the Gnutella network and their helpful suggestions.

Bibliography

[1] http://en.wikipedia.org/wiki/Botnet.

[2] http://en.wikipedia.org/wiki/Overlay_network.

[3] http://en.wikipedia.org/wiki/Napster.

[4] http://en.wikipedia.org/wiki/Distributed_hash_table.

[5] http://en.wikipedia.org/wiki/Overnet.

[6] http://www.symantec.com/security_response/index.jsp.

[7] http://www.viruslist.com/en/viruses/encyclopedia?virusid=24282.

[8] http://en.wikipedia.org/wiki/Gnutella_Web_Cache.

[9] http://en.wikipedia.org/wiki/Eigenvector_centrality.

[10] http://en.wikipedia.org/wiki/Degree_distribution.

[11] http://en.wikipedia.org/wiki/Clustering_coefficient.

[12] http://www.honeynet.org/.

[13] http://seclists.org/honeypots/2002/q4/0029.html.

[14] BitTorrent, http://www.bittorrent.com/.

[15] eMule, http://www.emule-project.net/.

[16] Gnutella protocol specification,
http://wiki.limewire.org/index.php?title=GDF.

[17] Lime Wire, http://www.limewire.com/.

[18] Send-Safe Honeypot Hunter,
http://www.send-safe.com/honeypot-hunter.html.

[19] S. Androutsellis-Theotokis and D. Spinellis: A survey of peer-to-peer content distribution technologies, ACM Computing Surveys, 36(4), 335–371 (2004).

[20] P. Barford and V. Yegneswaran: An inside look at botnets, Malware Detection, in Series: Advances in Information Security, pp. 171–191 (2006).

[21] K. Bhaduri, K. Das, and H. Kargupta: Peer-to-peer data mining, privacy issues, and games, Autonomous Intelligent Systems: Multi-Agents and Data Mining, Lecture Notes in Computer Science (2007).

[22] D. Dagon, G. Gu, C. Lee, and W. Lee: A taxonomy of botnet structures, Proc. 23rd Annual Computer Security Applications Conf. (ACSAC '07), Honolulu, HI (2007) pp. 325–339.

[23] D. Dagon, C. C. Zou, and W. Lee: Modeling botnet propagation using time zones, Proc. 13th Annual Network and Distributed System Security Symp. (NDSS '06), San Diego, CA (2006).

[24] C. R. Davis, J. M. Fernandez, S. Neville, and J. McHugh: Sybil attacks as a mitigation strategy against the storm botnet, Proc. 3rd Int. Conf. on Malicious and Unwanted Software (Malware '08), Alexandria, VA (2008) pp. 32–40.

[25] J. R. Douceur: The sybil attack, Proc. 1st Int. Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA (2002).

[26] S. Gianvecchio, M. Xie, Z. Wu, and H. Wang: Measurement and classification of humans and bots in internet chat, Proc. USENIX Security Symp. (Security '08), San Jose, CA (2008) pp. 155–169.

[27] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon: Peer-to-peer botnets: Overview and case study, Proc. 1st USENIX Workshop on Hot Topics in Understanding Botnets (HotBots '07), Cambridge, MA (2007).

[28] G. Gu, R. Perdisci, J. Zhang, and W. Lee: BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection, Proc. 17th USENIX Security Symp. (Security '08), San Jose, CA (2008) pp. 139–154.

[29] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee: BotHunter: Detecting malware infection through ids-driven dialog correlation, Proc. 16th USENIX Security Symp. (Security '07), Boston, MA (2007) pp. 167–182.

[30] G. Gu, J. Zhang, and W. Lee: BotSniffer: Detecting botnet command and control channels in network traffic, Proc. 15th Annual Network and Distributed System Security Symp. (NDSS '08), San Diego, CA (2008).

[31] K. J. Higgins: Researchers infiltrate and 'pollute' storm botnet, http://www.darkreading.com/security/encryption /showArticle.jhtml?articleID=211201340 (2008).

[32] T. Holz, M. Steiner, F. Dahl, E. W. Biersack, and F. Freiling: Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm, Proc. 1st Usenix Workshop on Large-scale Exploits and Emergent Threats (LEET '08), San Francisco, CA (2008).

[33] C. Kalt: Internet relay chat: architecture, Request for Comments: RFC 2810 (2000).

[34] O. Kolesnikov, D. Dagon, and W. Lee: Advanced polymorphic worms: Evading ids by blending in with normal traffic, Technical report, Georgia Tech (2004-2005).

[35] B. Krebs: Just how bad is the storm worm?, http://blog.washingtonpost.com/securityfix/2007/10/the_storm_worm_maelstrom_or_te.html (2007).

[36] F. Kuhn, S. Schmid, and R. Wattenhofer: A self-repairing peer-to-peer system resilient to dynamic adversarial churn, Proc. 4th Int. Workshop on Peer-to-Peer Systems (IPTPS '05), Ithaca, NY (2005).

[37] J. Liang, N. Naoumov, and K. W. Ross: The index poisoning attack in p2p file sharing systems, Proc. 25th IEEE Int. Conf. on Computer Communications (INFOCOM '06), Barcelona, Spain (2006).

[38] P. Maymounkov and D. Mazieres: Kademlia: A peer-to-peer information system based on the XOR metric, Proc. 1st Int. Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA (2002) pp. 53–65.

[39] T. Micro: Taxonomy of botnet threats, Technical report, Trend Micro White Paper (2006).

[40] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis: My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging, Proc. 1st USENIX Workshop on Hot Topics in Understanding Botnets (HotBots '07), Cambridge, MA (2007).

[41] A. Ramachandran, N. Feamster, and D. Dagon: Revealing botnet membership using DNSBL counter-intelligence, Proc. 2nd USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI '06), San Jose, CA (2006).

[42] K. Ramachandran and B. Sikdar: Modeling malware propagation in gnutella type peer-to-peer networks, Proc. 20th Int. Parallel and Distributed Processing Symp. (IPDPS '06), Rhodes Island, Greece (2006).

[43] E. V. Ruitenbeek and W. H. Sanders: Modeling peer-to-peer botnets, Proc. 5th Int. Conf. on Quantitative Evaluation of Systems (QEST '08), St Malo, France (2008) pp. 307–316.

[44] G. Starnberger, C. Kruegel, and E. Kirda: Overbot - a botnet protocol based on kademlia, Proc. 4th Int. Conf. on Security and Privacy in Communication Networks (SecureComm '08), Istanbul, Turkey (2008).

[45] D. Stutzbach and R. Rejaie: Capturing accurate snapshots of the gnutella network, Proc. IEEE Global Internet Symp., Miami, FL (2005) pp. 127–132.

[46] D. Stutzbach and R. Rejaie: Characterizing the two-tier gnutella topology, Proc. ACM SIGMETRICS, Poster Session, Alberta, Canada (2005) pp. 402–403.

[47] Y. Tang and S. Chen: Defending against internet worms: A signature-based approach, Proc. 24th IEEE Int. Conf. on Computer Communications (INFOCOM '05), Miami, FL (2005).

[48] R. Thommes and M. Coates: Epidemiological modelling of peer-to-peer viruses and pollution, Proc. 25th IEEE Int. Conf. on Computer Communications (INFOCOM '06), Barcelona, Spain (2006).

[49] R. Vogt, J. Aycock, and M. Jacobson: Army of botnets, Proc. 14th Network and Distributed System Security Symp. (NDSS '07), San Diego, CA (2007) pp. 111–123.

[50] P. Wang, S. Sparks, and C. C. Zou: An advanced hybrid peer-to-peer botnet, Proc. 1st USENIX Workshop on Hot Topics in Understanding Botnets (HotBots '07), Cambridge, MA (2007).

[51] L. Xie and S. Zhu: A feasibility study on defending against ultra-fast topological worms, Proc. 7th IEEE Int. Conf. on Peer-to-Peer Computing (P2P '07), Galway, Ireland (2007) pp. 61–70.

[52] W. Yu, P. C. Boyer, S. Chellappan, and D. Xuan: Peer-to-peer system-based active worm attacks: Modeling and analysis, Proc. IEEE Int. Conf. on Communications (ICC '05), Seoul, Korea (2005) pp. 295–300.

[53] L. Zhou, L. Zhang, F. McSherry, N. Immorlica, M. Costa, and S. Chien: A first look at peer-to-peer worms: Threats and defenses, Proc. 4th Int. Workshop on Peer-To-Peer Systems (IPTPS '05), Ithaca, NY (2005).

[54] Z. Zhu, G. Lu, Y. Chen, Z. J. Fu, P. Roberts, and K. Han: Botnet research survey, Proc. 32nd Annual IEEE Int. Computer Software and Applications (COMPSAC '08), Turku, Finland (2008) pp. 967–972.

[55] J. Zhuge, T. Holz, X. Han, J. Guo, and W. Zou: Characterizing the irc-based botnet phenomenon, Technical report, Peking University and University of Mannheim (2007).

[56] C. C. Zou and R. Cunningham: Honeypot-aware advanced botnet construction and maintenance, Proc. Int. Conf. on Dependable Systems and Networks (DSN '06), Philadelphia, PA (2006) pp. 199–208.

Index