IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 4, Ver. IV (Jul – Aug. 2014), PP 35-42
www.iosrjournals.org

Study of P2P Botnet

Avadhoot Joshi[1], Prof. M. S. Chaudhary[2]
[1](CSE, SIT, Lonavala/ Pune University, India)
[2](CSE, SIT, Lonavala/ Pune University, India)

Abstract: *Today, centralized botnets are still widely used. In a centralized botnet, bots are connected to several servers (called C&C servers) to obtain commands. This architecture is easy to construct and efficient in distributing botmaster's commands; however, it has a weak link - the C&C servers. Shutting down those servers would cause all the bots lose contact with their botmaster. In addition, defenders can easily monitor the botnet by creating a decoy to join a specified C&C channel. Today several P2P botnets have emerged Just like P2P networks, which are resilient to dynamic churn (i.e., peers join and leave the system at high rates), P2P botnet communication won't be disrupted when losing a number of bots. In a P2P botnet, there is no central server, and bots are connected to each other and act as both C&C server and client. P2P botnets have shown advantages over traditional centralized botnets. As the next generation of botnets, they are more robust and difficult for security community to defend. Researchers have started to pay attention to P2P botnets. 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.*

Keywords: *Botnet, Botmaster, KAD, P2P Botnet.*

I. 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. 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.

II. P2p Botnet

The lifetime 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 communicate with their botmaster for further instructions, such as launching an attack or performing an update.

In this section, we will discuss each stage in detail.

- **Stage one: recruiting bot members**

P2P networks are gaining popularity of distributed applications, such as file-sharing, web caching, network storage. 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 vulnerable hosts in existing P2P networks as bot candidates and build their zombie army. Many P2P malware have been reported, such as Gnuman, VBS.Gnutella and SdDrop. 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 the existing P2P network, and the network will be the only propagation media. On the contrary, P2P botnets we have witnessed recently 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.

- **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 of P2P botnets. In a parasite P2P botnet, all the bots are selected from vulnerable hosts within an existing P2P network, and it 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 belongs to this class of botnet. A bot-only P2P botnet builds its own network, in which all the members are bots, such as Stormnet and Nugache.

If all the 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. 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 neighbouring peer information.
2) There is a shared web cache, such as Gnutella web cache, 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 neighbouring 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 neighbouring peers it will contact later.

For instance, Trojan.Peacomm is a piece of malware to create a P2P botnet which uses the P2P protocol implemented on Overnet for C&C communication. 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 list to bootstrap onto the Overnet network. Another P2P botnet, Stormnet, 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.

- **Stage three: standing by for instructions**

Once a botnet is built up, all the bots 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 defences.

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 come and will forward receive 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& mechanisms can be applied in P2P botnets.

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. 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 botmaster.

Take the early version of Storm botnet for example, it utilizes the Overnet, and implements pull C&C mechanism. In this botnet, every day there are 32 hash keys queried by bots to retrieve commands. 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 employs the similar C&C design.

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

i. Which peers should a bot forward a command to?
ii. 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 neighbouring 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 neighbours, or 2) some peers in a bot's neighbour lists 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 certain popular files available which are predefined, 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 neighbouring 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 neighbours, 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 above discussion 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 hash key, which means queries for the same hash 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 hash keys, such that it can be sent to different parts of the network.

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 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 and the super botnet are two newly designed P2P botnets, whose C&C communication are not dependent on existing P2P protocols. Both of them implements 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 the list (push), and those who cannot accept connection from others periodically contacts other bots in the list 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 command from their C&C servers. Furthermore, the hybrid P2P botnet is able to effectively avoid bootstrap procedure, which is 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 defences [9].

2.1. Functionalities of P2P Botnet

In this section we detail two key functionalities of P2P bot: C&C functionality and P2P functionality. And then, several features are proposed to describe P2P bot. Finally, we analyse Storm bot and Nugache bot as an example.

A. Functionalities of P2P Bot

1) Command-and-control Functionality

The defining characteristic of bots is the remote control mechanism, by which we can distinguish bots from conventional viruses and worms. And we usually call it "command and control", C&C for short. Command-and control functionality enables bots to request, send, interpret commands and return results.

- Request: a bot asks another bot for command information. The bot has to know whom it should request, and we call the destination "predecessor".
- Send: a bot send command information to another bot. The bot has to know to whom it should send command information, and we call the destination "successor".
- Interpret: command information is interpreted as concrete instructions which bots can execute.
- Return: a bot sends execution results to its controller.

2) Peer-to-peer Functionality

The defining characteristic of P2P botnets is the peer-to peer communication style. Due to this communication style, P2P botnets are more resilient than centralized botnets.

Peer-to-peer functionality enables bots to construct and maintenance an overlay network, to route and locate, and to deal with joining, leaving and failure of peers. Generally P2P communication in P2P botnets is similar to that in common P2P applications, file-sharing systems for example.

B. Features of P2P Bot

Different functionalities correspond to different services for users. Features of a service consist of attributes including protocol, version and patch, and configurations including programs and parameters.

1) Command-and-control Features

C&C attributes consist of C&C-protocol, C&C-version and C&C-patch. C&C configurations consist of C&C programs and parameters. The later includes predecessor-table, successor-table, commands, and requests.

- Predecessor-table contains all predecessors a bot knows.
- Successor-table contains all successors a bot knows.
- Commands represent the set of command information.
- Requests represent the set of request information.

2) Peer-to-peer Features

P2P attributes consist of P2P -protocol, P2P -version and P2P -patch. P2P configurations consist of P2P programs and parameters. The later includes boots-table and routing-table.

- Boots-table contains all boots-peers a peer knows. A peer joins in a P2P network by at least one peer existing in the network, i.e. boots-peer. The content of boots-table may vary from P2P protocol to P2P protocol.
- Routing-table contains several nearest peers' information. A peer asks these peers to route queries and to locate resources. The same as boots-table, the content of boots-table may be different due to different P2P protocols.

C. Analysis of Storm and Nugache

Storm and Nugache are typical P2P botnets in the wild. We analyse them to verify our descriptive model with respect to describing a single P2P bot.

A Storm bot, by connecting to one or more bots in its boots-table, joins in a structured Overnet-based P2P network. Some bots publish their locations in the network. Others search for them as their predecessors, and initiate HTTP connection to them, requesting and downloading commands. Keys for publishing and searching are generated by built-in algorithms. In contrast with Storm, Nugache creates an unstructured P2P network in an ad hoc way. A bot connects to its neighbours in boots-table by encrypted TCP connection. There is no need for routing in Nugache, and accordingly no routing-table. In addition, when a bot receives commands, it passes commands to all its neighbours. TABLE I shows the difference of some features between Storm and Nugache [22].

Table I – Comparison Between Storm And Nugache [22]

	Features	
	Storm	Nugache
P2P Protocol	Overnet	Customized Protocol
C&C Protocol	HTTP	Customized Protocol
Boots-table	In a.ini file <p2pid, ip, port, flag>	In windows registry <ip, port>
Routing Table	Defined by overnet	None
Predecessor-table	Generates keys by built-in algorithms; determine successors by querying in P2P	Bots connected to it as its successors

2.2. Structures of P2P Botnets

According to the features mentioned above, we discuss relations between bots and structures in P2P botnets. To make the point clearer, we give a conceptual P2P botnet in FIGURE. 2.1. Vertices with letters "A" to "I" represent bot A to bot I respectively. Beside every vertex, it's the routing-table of every bot, containing two neighbours' information. Here, information of a bot may include IP address, port number and other properties, i.e. <IP, port, others>. Edges between every two vertices represent bots' relations, which are introduced in detail below.

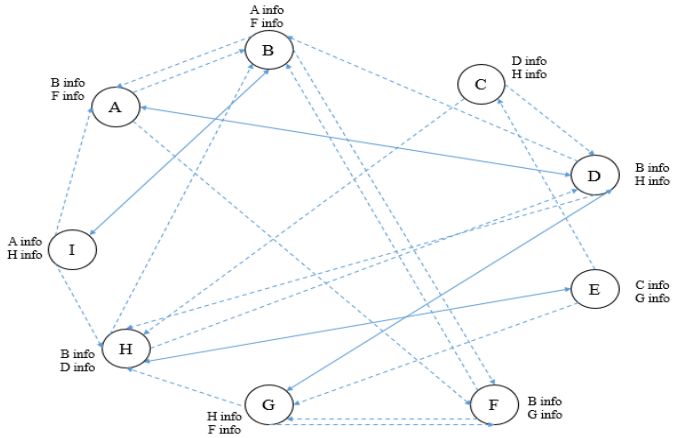

Figure 2.1 – A Conceptual P2p Botnet [22]

A. Peer-to-peer Structure

There are two relations in P2P communication: boots-relation and routing-relation. If bot_j appears in bot_i's boots-table, we define their relation as boots-relation. If bot appears in bot_i's routing-table, we define their relation as routing-relation.

Boots-table will be of no use after bots join in P2P network, while routing-relation can reflect possible connection in P2P communication. As a result, we discuss the structure based on routing-relation, which we call "P2P structure" in this paper. We define P2P structure as the set of bots and the set of their routing-relations, and represent it as P2PST ::=< BOTS, $R_{routing}$>.

In Fig. 2.1, every bot has two neighbours' information in its routing-table, and we draw dashed lines from bot to its neighbours' to represent the relations. For example, A has B and F in its routing-table, and then we draw dashed lines from A to B and from A to F. All vertices and dashed lines represent the P2P structure of the proposed botnet.

B. Command & Control Structure

There are two relations in C&C procedure: request-relation and send-relation. Request-relation is the relation that bot_i sends requests to bot_j. Send-relation is the relation that bot_i sends commands to bot_j.

Bots usually respond to requests automatically. Namely, if bot_i requests boy for commands, bot_j will immediately send commands it received to bot_i. For simplicity, we treat the to-and-fro communication as one C&C connection. Thus, we define C&C structure as the set of bots and the set of their C&C connections, and represent it as CCST ::=< BOTS, $R_{c\&c}$>.

Before establishing C&C connections, locations of bots' predecessors or successors are determined by P2P functionality. Take the proposed botnet as an example. By built-in algorithms, E knows that it should ask a bot with peer id "H" for commands. Then E will iteratively query for H's address information. One possible query procedure is: E sends query to its neighbour C and G; G returns H's information to E. At last, E requests and H responds, establishing a C&C connection, which is represented by a bidirectional line in Fig. 1. There are four C&C connections at the moment. All vertices and bidirectional lines represent C&C structures of the proposed botnet [22].

III. Related Work

As one of the most emergent threats to the Internet, botnets have attracted much attention. In this section we will discuss some research works which is already carried out for P2P botnet detection.

Some general characteristics of network traffic after a host infection [2]:

1. The communication pattern between hosts is abnormal:

For example:

- TCP SYN packet floods without any corresponding ACK packets

- ICMP echo request/reply packet floods.

2. **The number of incoming connections is very large**:

For example:

- Large amount of ICMP packets sent to a broadcast address with the IP address of the victim as the source address. This make all the hosts in the network reply to victim with ICMP reply packets. This is a smurf attack.
- A rise in the number of incoming C&C requests as bots connect to IRC server after infection.

3. **An unusual number of outgoing packets**:

For example: bots carrying out email spamming and denial of service attacks may show this characteristic.

4. **DNS queries to unusual servers** and establishing other communication with the IP address returned.
5. **Hosts trying to connect using unusual ports or protocols**. For example, if the network in question is never expected to use the IRC protocol and IRC transactions are observed.
6. **Virus or worm signatures** observed in packets entering or leaving a network boundary.

A number approaches for intrusion detection have been suggested in the past most of these approaches can be categorized into two categories: Artificial immune system & soft computing. A gossip protocol is used for detecting botnet as a distributed & cooperative approach which is discussed by Manoj Thakur [3]. The suggested approach is instrumental in providing robust spread of information related to compromised hosts in a given network. The use of gossip protocol provides the advantage of achieving convergent consistency even in the presence of an un-reliable underlying communication medium.

Another architecture proposed by Robert Erbacher, Adele Cutler, Pranab Banerjee, and Jim Marshall is a Multi-layered Approach. The key advantage of this architecture is that it allows for the integration of wide ranging techniques. This architecture do not limit to supporting only a single class of detection algorithms. By allowing algorithms from other researchers to be integrated through open architecture they allow for the greatest possible detection strategy. This architecture provides extensive set of capabilities such as detection of bot activities, mitigation of bot threats and attacks, Ability to extend the system with new antibot modules, Unification of all gathered data [4].

There are various mitigation techniques for IRC based botnet some which we have mentioned above but as we have discussed P2P botnet have already shown effectiveness over IRC based botnets, the researchers started thinking of mitigation of P2P based botnets. Reinier Schoof & Ralph Konig first published paper in 2007 for detecting peer to peer (P2P) botnets [7].

Current work on P2P-based botnets mainly focuses on monitoring, either passively or actively, behaviours of some existing P2P-based botnets, such as the Storm botnet. Although these bodies of work offer insights on how those botnets operate underground in reality, they have the following disadvantages. First, botnet monitoring usually takes place from a single or a few vantage points, thus cannot provide a full and consistent picture of the entire network. Second, researchers who attempt to actively measure an existing botnet may interfere with each other, potentially rendering information collected highly biased [8].

Another paper published "Detecting P2P – controlled bots on the host" by Antti Nummipuro discusses various solutions such as tracking data received over the network where it applies the idea of hooking system service table (SST), another solution suggested is Analysing the data received over network by analysing characteristics of network packets, As well as it suggests that Antivirus & behaviour blocking can also be used for detecting P2P controlled bots on the host [10].

Paper published by Thorsten Holz, Moritz Steiner, Frederic Dahl, Ernst Biersack, and Felix Freiling on "Measurements and mitigation of peer to peer based botnets: a case study on Storm Worm" in this paper, they showed how to generalize the methodology of botnet tracking for botnets with central server to botnets which use P2P for communication. They exemplified our methodology with a case study on Storm Worm, the most wide-spread P2P bot currently propagating in the wild. Their case study focussed on the communication within the botnet and especially the way the attacker and the bots communicate with each other. Storm Worm uses a two-tier architecture where the first-tier is contained within the P2P networks OVERNET and the Stormnet and used to find the second-tier computers that send the actual commands. They could distinguish the bots from the benign peers in the OVERNET network and identify the bots in the Stormnet and give some precise estimates about their numbers. Moreover, they presented two techniques how to disrupt the communication of the bots in both networks. While eclipsing is not very successful, polluting proved to be very effective. In future work, we plan to analyse in detail the second-tier computers and try to find ways to identify the operators of the Storm Worm [11].

Various techniques for mitigation of P2P botnet are suggested in "On the effectiveness of structural Detection and Defence Against P2P-Based Botnet" paper published by Duc. T. Ha, Guanhua Ya, Stephan

Eidenbenz, Hung Q. Ngo such as Poisoning based mitigation, Sybil based mitigation, Eclipse based mitigation [12].

Sybil attack is one of the renowned attack that is going to be implemented for mitigation of P2P based botnet network. This concept is first coined John R. Douceur of Microsoft Research in his paper "The Sybil Attack" which suggest the injection of Sybil node i.e. the node which do not forward the commands botmaster to the bots connected to it [13].

The concept of Sybil attack is further elaborated by Zhou Hangxia in his paper "Mitigating Peer-To-Peer Botnets by Sybil attacks" in year 2010 in publication of Asia-Pacific conference on information technology & ocean engineering in this paper Zhou suggest two variants of Sybil attack random Sybil attack which suggests to random injection of Sybil nodes into a botnet network, also he described one more variant of Sybil attack named d-choice Sybil attack [14].

The paper published in IEEE ICC 2010 proceeding by Oliver Jetter, Jochen Dinger, and Hannes Hartenstein named "Quantitative analysis of the Sybil Attack and Effective Sybil Resistance in Peer-to-Peer Systems" this paper discusses the effectiveness of Sybil attack with available Sybil resistance in existing P2P protocols [16].

Paper published by Moritz Steiner, Taoufik En-Najjary, an Ernst W. Biersack on "Exploiting KAD – Possible uses and misuses" describes about possible threat in KAD network implemented with the help of Kademlia Protocol. Few of which are Sybil attacks in KAD such as spying on publish and search traffic, Eclipsing content, DDOS attack author also discusses about preventing Sybil attack in KAD [23].

Thibault Cholez, Isabelle Chrisment and Oliver Fester have published a paper named "Evaluation of Sybil attack Protection schemes in KAD" which describes various experiments performed to avoid or to reduce the effects of Sybil attacks in KAD network with the help of different experimental approaches such as packet tracking and flood protection, IP address limitation, identities verification etc. [35].

IV. Conclusion

From this paper we have studied the structure of P2P botnet which highlights the different stages in botnet lifecycle in which it describes about how a P2P botnet can be constructed this paper elaborates further how botnet will be prepared for sending and receiving commands to and from the botmaster. This paper highlights the functionalities and features of P2P botnet which have lead many researchers and scientists to think about prevention or detection methodologies for P2P botnet. This paper summarizes the various research carried out for mitigation of P2P botnet which creates a solid ground for further research work. Also this paper elaborates the future work in P2P botnet prevention with the help of Sybil attack and use of kademlia protocol.

References

[1]. Abhijeet B. Potey, Prof.Anjali B.Raut "Defending Sybil Using Social Network", International Journal of Engineering and Computer Science (IJECS) Volume 2 Issue, Page No. 196-199, 2 Feb 2013.
[2]. "Botnets - the evolving nature of adversaries, tactics, techniques and procedures" Georgia Tech Cyber Security Summit, 2011, Pages 6-7.
[3]. Joseph Massi, Sudhir Panda, Girish Rajappa, Senthil Selvaraj and Swapana Revankar "Botnet Detection and Mitigation" Proceedings of Student-Faculty Research Day, Pace University, 2010.
[4]. Andrew White, Alan Tickle, and Andrew Clark "Overcoming Reputation and Proof-of-Work Systems in Botnets" Fourth international Conference on Network and System Security, 2010.
[5]. Zhou Hangxia "Mitigating Peer-to-Peer Botnets by Sybil attacks", International Conference on Innovative Computing and Communication and Asia-Pacific Conference on Information Technology and Ocean engineering © IEEE, 2010.
[6]. Oliver Jetter, Jochen Dinger, and Hannes Hartenstein "Quantitative Analysis of the Sybil Attack and Effective Sybil Resistance in Peer-to-Peer Systems", IEEE ICC proceedings, 2010.
[7]. Junfeng Duan, Jian Jiao, Chunhe Xia, Shan Yao, and Xiaojian Li "Descriptive Model of Peer-to-Peer Botnet Structures", International Conference on Educational and Information Technology (lCEIT), 2010.
[8]. Ping Wang, Sherri Sparks, and Cliff C. Zou "An Advanced Hybrid Peer-to-Peer Botnet", IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 7, NO. 2, Pages 113-127, APRIL-JUNE 2010.
[9]. Ping Wang, Lei Wu, Baber Aslam and Cliff C. Zou "A Systematic Study on Peer-to-Peer Botnets" IEEE, 2009.
[10]. Carlton R. Davis, Jos´e M. Fernandez, and Stephen Neville "Optimising Sybil Attacks against P2P-based Botnets", 2009.
[11]. Ping Wang, Lei Wu, Baber Aslam and Cliff C. Zou "A Systematic Study on Peer-to-Peer Botnets" IEEE, 2009.
[12]. Duc T. Ha, Guanhua Yan, Stephan Eidenbenz, and Hung Q. Ngo "On the Effectiveness of Structural Detection and Defence Against P2P-based Botnets", 2009.
[13]. Thibault Cholez, Isabelle Chrisment and Olivier Festor "Evaluation of Sybil Attacks Protection Schemes in KAD", published in 3rd International Conference on Autonomous Infrastructure, Management and Security – AIMS, 2009.
[14]. Robert F. Erbacher, Adele Cutler, Pranab Banerjee, and Jim Marshall "A Multi-Layered Approach to Botnet Detection", 2008.
[15]. Thorsten Holz, Moritz Steiner, Frederic Dahl, Ernst Biersack, and Felix Freiling "Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm", 2008.
[16]. Reinier Schoof and Ralph Koning "Detecting peer-to-peer botnets", 2007.
[17]. Antti Nummipuro "Detecting P2P-Controlled Bots on the Host" Seminar on Network security, 2007.
[18]. Daniel Stutzbach and Reza Rejaie "Improving Lookup Performance over a Widely-Deployed DHT", infocom, 2006.
[19]. "Kademlia: A Design Specification", the XLattice Project, 2003-2006.
[20]. John R. Douceur "The Sybil Attack", in the proceeding of first international workshop on peer-to-peer systems (IPTPS), Pages 251-256, 2002.

[21]. Petar Maymounkov and David Mazi`eres "Kademlia: A Peer-to-peer Information System Based on the XOR Metric".
[22]. Benedikt Westermann, Andriy Panchenko, and Lexi Pimenidis "A Kademlia-based Node Lookup System for Anonymization Networks".
[23]. Isabel Pita and Adrian Riesco "Specifying and Analysing the Kademlia Protocol in Maude*".
[24]. Manoj Rameshchandra Thakur "Distributed and Cooperative Approach to Botnet Detection Using Gossip Protocol".
[25]. Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack "Exploiting KAD: Possible Uses and Misuses".
[26]. M. Patrick Collins, Timothy J. Shimeall, Sidney Faber, Jeff Janies, Rhiannon Weaver, and Markus De Shon "Using uncleanliness to predict future botnet addresses".
[27]. Kademlia - Wikipedia, the free encyclopedia.
[28]. Distributed hash table - Wikipedia, the free encyclopedia.