Turning P2P Networks into DDoS Engines: A Survey

Hamid Farhadi [a], Behzad Akbari [b *], Shahab Rajaee [a], Mohammad Farahani [a]

[a] *School of Science and Eng., Sharif University of Technology (International Campus), Kish, Iran*
[b] *Department of Electrical and Computer Eng., Tarbiat Modares University, Tehran, Iran*

Received 9 November 2009; revised 8 May 2010; accepted 25 May 2010

Abstract

Recently, Peer-to-Peer (P2P) networks contribute to a large fraction of the Internet backbone traffic. Consequently, misusing such networks for malicious purposes is a potential side effect. In this review article, we investigate different techniques of misusing P2P overlay networks to launch large-scale next-generation Distributed Denial of Service (DDoS) attacks. In particular, we investigate representative systems of the structured (Overnet), unstructured (Gnutella) and hybrid (BitTorrent) P2P overlay networks. Real world experiments indicate the high performance, difficulty in detection and tracking, and the low cost of launching such attacks.

Keywords: P2P overlay Network, DDoS Attack, Overnet, Gnutella, BitTorrent.

1. Introduction

The Internet has experienced considerable changes since its creation. It has grown from a small scientific prototype, for exchanging data among trusted parties, to a complex communication infrastructure, transmitting various types of data such as multimedia and financial transactions. Conflict of interests, is a natural phenomenon in such environments, and criminal motivations are inevitable [1]. This leads to attacks from malicious users against innocent victims or the infrastructure itself. One of the most important types of attacks over the Internet is the Denial of Service (DOS) attack which causes a service or property (e.g. web server or network connectivity) to be disrupted [2]. An elementary implementation mechanism of the DoS attack involves flooding the target with either ordinary (i.e. non-attack) or attack traffic in a manner that the requests belonging to the legitimate users have almost no chance to be responded. By the attack traffic, we mean the useless traffic that is generated by a peer unintentionally because of some protocol exploitation by the attacker. Such attacks, can be performed separately or as a part of a complicated attack.

Distributed Denial of Service (DDoS) attack is an improved version of DoS attack in which redundant DoS attacks are coordinated as a single attack which is geographically distributed. Millions of contributors may cooperate in forming an effective attack that can severely impact even on high bandwidth backbone facilities. Several types of DDoS attacks exist that are much more sophisticated than the ordinary overloading of resources via bombarding the target with a large number of packets. Launching such attacks usually needs little knowledge both about the attack and about the target and its security state. Mitigation methods against DDoS attacks are a live research topic in the community, and are still far from a full-proof solution.

DDoS attacks can be applied for different purposes; they can be used for terrorism and vandalism or as an aid to the other attacks. For example, it can be used to overload an Intrusion Detection System by many fake attacks, targeting different hosts, in order to mask the main attack.

Traditional DDoS attacks include two steps: Firstly, using different techniques such as viruses, Trojan Horses, buffer overflow attacks, etc., many hosts are compromised, organizing them into a type of overlay network (also known as Botnet or zombie network) with a single Command and Control (C&C) host; secondly, the C&C host (also known as botmaster) sends the attack command to the overlay, and each member starts attacking the victim. The first step can be a time-consuming task if it is carried out manually. Therefore, many attackers try to automate the botnet construction process using techniques such as

* Corresponding Author. Email: b.akbari@modares.ac.ir

Internet worms that continuously seek for a new target to infect. That explains why botnets are sold expensively in the black market. Another difficulty in conducting a DDoS attack is the heavy cost associated with botnet maintenance. While more and more Internet users recognize the importance of the security of computer systems and networks, they are more likely to remove their systems from the botnet and provide some protection against possible re-infection. Moreover, most powerful members of botnets are server hosts that possess more computational capability and connection bandwidth, and are usually maintained by professional administrators who detect malicious activities in their systems, sooner or later.

In this paper, we narrow in on a particular type of DDOS attacks which exploit P2P networks as their traffic generation engine.

The rest of the paper is organized as follows: first, in the remaining of this section, we review the preliminary concepts of P2P networks and briefly explain how they can be exploited for conducting DDOS attacks over the Internet. We then investigate the exploitation of different P2P file sharing networks for running DDOS attacks in Sections II, III and IV. In particular, Gnutella, as an unstructured P2P network, is investigated in Section II. Section III is centered on Overnet as a representative example of a structured network. The BitTorrent network along with the specifics of its exploitation as a DDOS attack engine is presented in section IV. The paper ends with concluding remarks.

1.1. Peer-to-Peer Overlay Networks

P2P overlays are application layer networks, made up of several peers organized into a self-managing network on top of the Internet Protocol. It is reported that P2P file sharing applications contribute to more than 70% of the backbone traffic in some areas, and are getting more prevalent among Internet users [3]. P2P overlay networks are divided into two groups: *Structured* and *Unstructured*.

Technically, in a structured P2P topology, a mapping is defined between a peer's ID and the shared content in order to support more efficient queries. These networks usually use a hash table in which the location information associated with the data object (or value) is placed deterministically at the peers with identifiers corresponding to the unique key of the given data object. This table is stored in a distributed fashion amongst a selective number of peers, and acts as a routing table by consistently assigning uniform unique numbers to both peers and shared files, i.e. in the form of Peer-IDs and keys, respectively [4].

In Unstructured P2P networks, the location of the data objects has no logical relation with the member's location. In other words, new peers join the network without having a clue about the network's topology. In effect, the members rely on a query flooding mechanism to transmit/pass their messages over the network. Depending on the network policies, these queries get disseminated in a depth-first or breadth-first manner, until their time-to-live (TTL) reaches zero and the query expires. *Hybrid* P2P networks, on the other hand, are defined from the combination of structured and unstructured P2P networks, exhibiting characteristics from both paradigms [4].

Napster, a precursor P2P file sharing system, was introduced in 1999. It attracted mainstream attention and has been popularly used for music sharing. Napster relies on a centralized server to maintain a list of live clients together with the shared files. Napster, of course, was shut down due to copyright violations [5]. Gnutella was the second P2P file sharing network that also received a great deal of attention. It is a purely distributed P2P network that works independently of any single server. Each new peer needs to be aware of the address of a single remote peer during the bootstrap phase [6]. FastTrack is yet another P2P network which makes use of some powerful and long-lasting nodes (i.e. super-nodes) as the distributed directory service of the overly, with each super-node in charge of serving a number of peers as their children. Kazaa was one of the popular FastTrack-based P2P overlays. The notion of super-nodes in Kazaa allows the joining and searching overhead to be balanced out over a number of more stable and powerful hosts, leading to a more scalable design [7]. Later, Bram Cohen introduced a new P2P file sharing network using hybrid characteristics from both unstructured and structured P2P overlays, the so-called BitTorrent syetm. It was designed for efficient sharing of large files over the Internet with fairness mechanisms (e.g., the tit-for-tat policy) that revolves the users' habits from *download only* to *download and upload* behavior. BitTorrent is efficient in transferring large files among peers, and has, thus, gained popularity within the last several years [8,9].

In the future, we may witness P2P networks completely free of any centralized server and with all nodes providing the same functionality. In addition to traffic encryption/obfuscation, we expect some anonymity preserving improvements that hide the source and the destination of the connections. Moreover, long-life connections may also be used in conjunction with the currently short-term connections between peers.

1.2. Using P2P Networks as DDoS Attack Frameworks

A hijacked P2P network allows launching large-scale DDoS attacks against a host even outside the overlay. That is, while there is no compromised host in the overlay, the P2P protocol itself is exploited and compromised. Ease of operation, low cost, high performance and difficulty in defense and detection have made the misuse of P2P systems for DDoS attacks a new research topic in the community [10-13]. In this sub-section, we discuss how the effectiveness of using P2P networks as DDoS attack engines is justified in the context of next generation DDoS attacks.

Turning P2P overlays into DDoS attack engines is advantageous to attackers for the following reasons: First,

in traditional DDoS attacks, the intruder compromises a large number of hosts by turning them into zombies and by installing a small malware (usually undetectable by anti-malware software) to control the zombies. This process can be done either manually or in an automated fashion using robot software (e.g. worms). On the other hand, in P2P-based DDoS attacks, there is not any direct attack to the hosts in the overlay at all; instead, the P2P protocol vulnerabilities are subject to exploitation. Misusing the P2P protocol, the attacker is able to mislead the traffic to a victim host. The traffic can be either legitimate or abnormally generated using P2P protocol vulnerabilities. In the latter case, the traffic is generated in the peers that are maliciously stimulated by the attacker node. In addition, using P2P protocol vulnerabilities, one may hijack a million peers in a day, as opposed to the traditional methods in which much more time is usually needed for constructing a botnet of the same size.

Second, in traditional methods, the constructed botnets may not be reusable after a month. That is, due to the growing awareness of both desktop users and system administrators with respect to security issues, the botnet members will be detected by their owners and removed from the network. Additionally, anti-malware vendors may detect the controlling software on zombie hosts and come up with a remedy for its removal through their malware definition updates. In contrast, as P2P users are growing, the size of botnets in such networks can even increase in time. In P2P-based DDoS attack botnet construction process, the intruder neither exploits any software or service on the hosts, nor does it gain any kind of access on peer systems. Thus, there is not any malware installed on peer hosts. Moreover, even a user with above-average knowledge of computer security would not be able to easily understand such an anomalous behavior in his P2P client software. In this case, the botnet size is not expected to be subject to a considerable decrease in time.

Third, the undetectability of the DDoS attack strongly depends on the geographical distribution of the botnet. The more the botnets are geographically dispersed the more chance a typical attack has of going undetected. In traditional DDoS Attacks, zombie hosts are compromised either manually or automated. In the former case, the attacker needs to manage the distribution of botnet himself, while in the latter case, the automated robots seek for new victims usually in the same range as their own machine. As a result, we may see many zombies from a single service provider (e.g. a large ADSL service provider) in the botnet, a fact which eases the detection and tracking of the attack. In P2P-based DDoS attack, however, the attack's geographical distribution obviously mirrors the distribution of the overlay. Due to the popularity of P2Ps, thousands of Autonomous Systems may contribute to the overlay network, essentially making it difficult to trace back the botmaster.

Fourth, in attacker trace-back investigations, the computer forensic specialists first detect a zombie host, and then try to find the C&C channel between the zombie host and the botmaster in order to determine the main attacker. In traditional DDoS attack methods, C&C is usually performed using an available infrastructure such as IRC [14] or Twitter [15]. Therefore, the investigators seek for evidence on such connection infrastructures to detect the botmaster. However, in P2P-based DDoS attacks, the legitimately running P2P protocol is hijacked to facilitate the communication. Therefore, it is more difficult to determine the attack source via inspection of a zombie host.

Fifth, the generated traffic from the DDoS attack engine plays an important role in the success of the attack. In traditional methods, the engine's output depends on the total bandwidth of the botnet. Conversely, in P2P-based DDoS attacks, the total overlay traffic can potentially appear at the engine's output. Given that a large fraction of the Internet backbone traffic is generated from such overlays, it is unlikely to have as little traffic as in traditional botnets. Hence, P2P-based DDoS attacks are potentially capable of generating more traffic compared to the traditional DDoS attacks.

However, turning P2P overlays into DDoS attack engines is also associated with a number of drawbacks. A hijacked P2P overlay can only be used as a DDoS attack engine; this is while the traditional botnets are essentially multipurpose platforms for sabotaging activities. For instance, a botnet can be used as a chain of proxies for hiding the attacker from the victim. The longer the chain, the more anonymity the attacker can achieve. Yet another example is the case of gathering all documents or credit card information from the compromised systems for possible trade in the black market [16,17].

2. Gnutella: An Unstructured P2P Network

The Gnutella network is an unstructured P2P system. Every node in this system, called *servent,* has the roles of both server and client. Each node provides a client-side interface. Through this interface, users can issue queries and view search results, accept queries from other servents, check for matches against its local data set and respond with the corresponding results.

Since Gnutella network features a pure Ad-Hoc topology, the clients may join/leave the network at any time. Thus, the protocol has been designed as a highly fault-tolerant mechanism to minimize the effects of *peer churns*. However, this leads to a large overhead induced by synchronization messages, traveling amongst the peers in the network. Every peer, at first, searches the entire network for its desired files and establishes connections with peers who own the searched files. Then, the peer begins downloading directly from the peers. There are different kinds of messages in the Gnutella network [18]:

● Group Membership (*PING* and *PONG*) Messages; every new node needs to find a connection point to join the Gnutella network. Thus, most client vendors set up a number of host-cache servers which maintain the IP address of the other peers currently connected to the

network. After obtaining a random number of available peers, the new node broadcasts a *PING* message to the other nodes to announce its presence. Upon receipt of the *PING* message, each node issues a back-propagated *PONG* message towards the sender. The *PONG* message contains some information about the node such as its IP address, port number, the number and the size of the shared files, etc. At the same time, the node forwards the received *PING* message to its neighbors.

- Search (*QUERY* and *QUERYHIT*) Messages; in order to look up its desired files, each node broadcasts *QUERY* messages, containing the user specified search string. In order to respond to a *QUERY*, the other nodes, receiving the *QUERY* message, are free to send back a *QUERYHIT* message to the *QUERY* sender. The *QUERYHIT* message includes the information needed for downloading the specified file(s).
- File Transfer (*GET* and *PUSH*) Messages; file downloads are done directly between two peers using *GET/PUSH* messages. The *PUSH* message is used when the node receiving the *QUERY* is firewalled. In this situation, the *QUERY* sender can identify its request to the firewalled node with a *PUSH* message to push the file.

2.1. Gnutella's Vulnerabilities and Attack Description

Gnutella's designers had never considered security as a basic design principle; security, in the context of these systems, has always been sacrificed for the sake of more simplicity and better performance. In particular, since messages are sent in plain text format, an attacker is able to read and modify the messages. Attackers may also exploit this vulnerability to run spamming DDoS attacks, as will be explained later.

Furthermore, due to processing overhead and response time considerations, encryption is not accounted for in the Gnutella protocol. Hence, a malicious peer can spy on other peers' activities and violate their personal privacy.

In the following section, we explain these vulnerabilities and discuss possible DDoS attacks over the Gnutella network.

2.1.1. DDoS Attack through Spamming

In Email world, "spam" is a familiar word given that typically everyone receives unsolicited emails occasionally. It is, however, rather easy to delete the spam emails with little nuisance. This is not the same when it comes to a spammed query result in the Gnutella network. A spammed query could make a peer an active part of a DDoS attack against any arbitrary host.

There is no provision for dealing with DDoS attacks in the Gnutella network. A malicious peer can launch a DDoS attack via answering positively to any query it receives, causing every peer to download a resource from the victim. In order to get a concrete idea, let P_1 and P_2 be two Gnutella peers and let *M* be a malicious peer in the network. Finally, let A be a host, subject to the attack and claimed to have the searched files by M. P_1 and P_2 send *QUERY* messages to peer M in order to search for and download different files. M responds to both peers with a *QUERYHIT* message, directing P_1 and P_2 to download their interested files from peer A. To render the *QUERYHIT* message more appealing to the peers, M can also indicate in the message that A is a server with a LAN connection. In addition, it tries to come up with a file name as relevant to the initial *QUERY* as possible. This can be achieved by using the initial *QUERY* keywords appended with a random extension from the popular media files. With this understanding, it is possible that a large number of Gnutella peers connect to the victim and try to download a non-existent file. The host can respond with an HTTP 404, which means that it was unable to locate the requested file. Since an HTTP 404 response code may not be too difficult to handle for a server, authors in [19] have presented a method so as to force the Gnutella peers to actually perform a real download from the server. They have essentially embedded specially composed file names in the *QUERYHIT* message, making the attack more efficient.

In order to maximize the number of polluted peers in the network, the malicious node M first connects to a Gnutella IP host-cache server so as to obtain a random number of available hosts which are connected to the network. After polluting them with spammed *QUERYHIT* messages, following the aforementioned mechanism, M disconnects from these peers. It, however, immediately re-connects to the host-cache server to obtain another random number of hosts for repeating the same procedure [20].

The interesting aspect of spamming DDoS attack in Gnutella network is that most of the poisoned peers insist on downloading from the victim for a long time. They keep trying even after the malicious peer was shut down given that many Gnutella clients are designed to endlessly and periodically retry downloading in case the initial download attempt fails.

2.1.2. The PONG Attack

The other type of a DDoS attack is the *PONG* attack. Although this attack is not as powerful as the spamming attack, yet it can illustrate the other weaknesses of the Gnutella protocol. As mentioned before, when a peer P sends a *PING* message to the other peers, they reply to the received *PING* message with a *PONG*. A *PONG* message typically contains the IP address and the port number of the replier. Let M be a malicious peer and H be a typical Internet host which is not connected to the Gnutella network. M could easily respond with a *PONG* message which contains the IP address and the port number of a host like H. Now P, receiving a *PONG* message from M, sets up a connection with H, stores the H's address in its cache and forwards all its *QUERY* messages to H. By sending the *PONG* message as a response to many *PING* messages, the malicious peer could direct a bunch of *QUERY* messages to

the host H. Every peer in the Gnutella network re-sends a *PING* message to all the neighbors in its cache after a short time period to discover the changes in the network topology. Hence, the *PONG* attack only lasts for a short time period, given that H is not a real Gnutella peer and cannot respond to the P's *PONG* message. Finally, P removes H from its cache and the attack terminates.

2.2. Implementation Achievements

Authors in [19] have employed five malicious peers and five distinct web servers. The peers simultaneously generated 10K, 100K, 1M, 10M and 100M *QUERYHIT* messages. The traces from the first two peers have been discarded, since their query rate was quite low in comparison with the other three. The recorded log files have shown that for 10M *QUERYHITs*, more than 10 million downloads have been recorded from 52,473 unique IP addresses within about one day. On the other hand, the larger attack trace, with 100M *QUERYHIT* messages, embeds nearly 71 million downloads issued by 421,217 unique IP addresses within more than 12 days. The observation indicated that nearly 80% of the IP addresses issue less than 100 requests and about 0.5% issue thousands of requests. One interesting observation has been that the web servers logged some download requests even 10 days after the end of the experiment. It suggests that some peers that could not download the content they were looking for kept on trying for many days, insinuating that Gnutella has a kind of *memory*. Similar results are obtained in [20].

3. Overnet, a Structured P2P Network

In this section, we examine, in detail, the exploitation of Overnet as a DDoS attack engine. Overnet is a part of the eDonkey client software. Kad, deployed in eMule, is an open-source version of Overnet. Both Overnet and Kad are based on the Kademlia Distributed Hash Table (DHT) [21] which is similar in many aspects to Pastry [22] and Tapestry [23] protocols.

3.1. Overview

To understand how Overnet can be exploited, we first review some of its relevant features and then describe its vulnerabilities.

3.1.1. Join Process

A new node has to know at least one live peer in the overlay to be able to join the network. The known node may have been either cached from previous sessions or provided manually by the user. New nodes use a 128-bit key as their unique identifier and then search for their own IDs through the DHT. The search process uses iterative UDP messages. By sending a series of UDP messages from the querying peer, the Kademlia DHT would be able to find peers with the closest IDs, using a simple XOR-based distance measurement function.

3.1.2. Routing Table Construction and Maintenance

During the search process initiated by the newly joined peer X, each intermediate peer Y routes the search messages to a Y' with an ID which is closer to the ID of X. Intermediate peers also send their own routing tables to the querying peer. Using the received routing tables, X is able to construct its own table by aggregating entries of the received tables [21]. Following the successful creation of the routing table, X announces itself to all its neighbors in the routing table. Each peer receiving such presence announcements adds an entry for X in its own routing table.

3.1.3. File Advertisement

The new peer starts advertising its shared data objects (files in this case) after introducing itself to the overlay in the joining process. The file publishing process consists of two steps:

1. *Publishing the location information*: first, all files are hashed to yield a representative identifier. Then, peer *X* sends a *location publish message*, including the file ID *H* and its location (i.e. the IP address and the corresponding port number) to the DHT. The publish message is routed to the peer with the closest ID to H in the ID space via iterative routing through the overlay. The closest peer to H updates its local index of files.
2. *Publishing the metadata information*: in the second step, the source peer extracts keywords from the file name. Then, it hashes each keyword separately and makes a 128-bit key for each keyword G. All keywords are then sent to the DHT via a *metadata publish message*. Each message contains the keyword hash plus the related file hash and some metadata information (e.g. title, size, type, etc.). Similar to information publish messages, the metadata publish messages are routed iteratively through the DHT to the peer with the closest ID to G in the ID space. The peer with the closest ID to G updates its local index with the newly received keyword hash G.

3.1.4. File Searching

The search mechanism is similar to the file advertisement process. The difference is that during the search, the two steps of the advertisement process are performed in reverse order. First of all, the client software GUI receives *n* keywords from the user to initiate the search. Then, it calculates *n* hashes from the keywords and sends *n* search messages (one for each) to the overlay. The searching message crawls iteratively through the DHT to reach the peer with the closest ID to the keyword hash. The search results in the identification of a peer that holds some records stating where the corresponding files are located. The matching records are sent to the querying peer, passing the file's ID (i.e. the hash of the file) plus all available metadata. Thus, the querying peer receives various sets of IDs, one for each keyword. Depending on the client

configurations, the raw search results may be filtered and then shown to the user. The user can select some to be downloaded via the client GUI. When the user selects a file, a location search message, including the file identifier, is sent to the overlay. After a number of iterations on the DHT, the message reaches to the corresponding peer. The peer holds a set of location information, such as pairs of IP addresses and port numbers, each representing the location of a file copy. Finally, the client software establishes TCP connections to those peers to receive the file.

3.2. Overnet's Vulnerabilities and Attack Description

In this section, we discuss different vulnerabilities and attacks in the context of Overnet and KAD.

3.2.1. Asymmetric Attack

Request and response messages in a given protocol come in varying sizes. Exploiting this feature, it is possible to send smaller or fewer messages while posing them to the victim as larger or more response messages. For example, a bootstrap request in the eMule client software includes only 2 bytes of data, while the bootstrap response message, that contains information from 20 peers, needs 527 bytes of data. Using this feature of the protocol, the attacker can achieve a considerable amplification with which to increase the attack's performance. In the case of 20 peers, the response message amplified more than 260 times that of the request message. However, this method is only applicable to the victims inside the overlay [24].

3.2.2. Index Poisoning Attack

In the index poisoning attack, the underlying idea is that the attacker advertises the victim as a source of some popular files. Therefore, many overwhelming requests are routed to the victim. It can be realized since there is no validity checking for the advertisements in the overlay. To add more details, the attacker first crawls on the network and grabs as many node addresses as possible. Then, it sends a location publish message to each of these nodes. The messages are spoofed with the victim's IP address and port number and also contain an arbitrary file hash. Any peer, receiving such a message, adds it into its local index along with the location of the victim. In fact, the peer neither checks whether the file exists at the claimed source, nor does it verify if the source peer is actually a member of the overlay. Eventually, any matching search query for the advertised file is redirected to the victim. Later, the querying peer establishes a fully open TCP connection to the service running on the target. Depending on the service configurations at the target, the connection may last for a minute or more [25].

3.2.3. Routing Table Poisoning Attack

The idea is that the attacker falsely advertises the victim as the neighbor of many nodes in the overlay. Hence, the victim receives many messages of different types routed by the DHT and exhausting its resources. It can be realized since there is not any mechanism to verify whether an announced peer is a valid overlay member or not. Particularly, the attacker sends an announcement message to every crawled peer. These messages are spoofed with the victim's IP address, making it possible that the compromised node be added into the routing table of any crawled peer (of course depending on the node ID included in the message). Let Y denote the peer currently visited by the crawler and Z denote the victim system. During crawling, the Y's node ID can be retrieved, and by using close IDs to that of Y's in the announcement message, the attacker can boost the probability of adding Z to the routing table of Y. Such poisoning is feasible since there is no mechanism in Y to check if Z is a valid peer in the overlay. The victim Z receives many messages if (i) a lot of crawled nodes add a record to their routing tables, stating Z as their neighbor, and (ii) Z's ID is such that it is often selected as a neighbor from the routing table of Y. The routing table poisoning attack can not only be launched using announcement messages, but also one may perform the attack using publish and query messages [26].

3.3. Implementation Achievements

In [26], a list of 7,564 file hashes has been used. The routing table poisoning attack generated on average about 1.5 Mbps of upstream traffic at the victim. Moreover, the traffic has been launched from hundreds of thousands of different Overnet nodes. Index poisoning caused over 300 TCP connections to hang at the victim, which persisted for hours after the attack. This can be attributed to the fact that for the entire duration of the attack, the target host received traffic from 340,274 peers from 22,484 Autonomous Systems. As a result, a good geographical distribution has been achieved. Similar results have been presented in [25] and [26].

4. BitTorrent: A Hybrid P2P Network

4.1. Overview

A recent analysis of the latest P2P trends worldwide shows that BitTorrent is still the most popular file sharing protocol [27]. Although emerging technologies to enforce traffic shaping restrictions (applied by some ISPs) have declined P2P traffic, the BitTorrent traffic is still on the rise (Figure 1). It is responsible for more than 45-78% of all P2P traffic, i.e. roughly 27-55% of all Internet traffic across different regions, as documented by Ipoque Internet Study group [28]. The key fact that can be inferred from this report is that in most regions, BitTorrent is the dominating protocol where HTTP could best stand in the second place. As can be seen in the figure, the usage share of BitTorrent clients is much higher than that of the other P2P protocols.
BitTorrent has proved to be extremely effective in

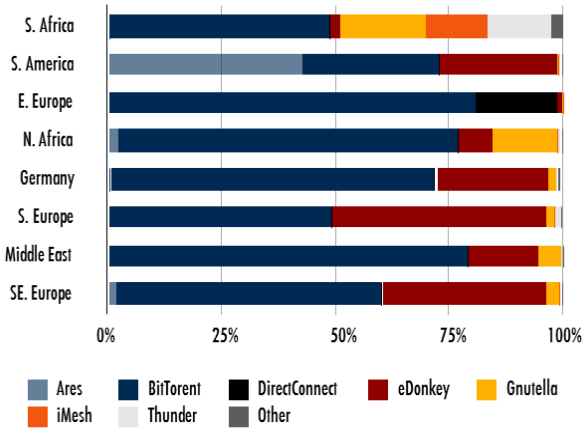

Fig. 1. Most popular P2P protocols [28].

distributing files and it has gained great achievements as a massive-scale content distributer. BitTorrent's users doubled from 2006 to 2007 [28], with also an outstanding increase in 2008; for instance, downloading torrent files from Minova, a popular torrent search engine website, doubled and reached to 7 million in 2008 [29]. All these achievements have rendered BitTorrent into a successful blueprint for other P2P-based systems; therefore, it is particularly important to know the possible vulnerabilities and hazards associated with the exploitation of such a popular protocol.

4.1.1. Terminology

In this section, we give a brief overview of the BitTorrent system with special emphasis put on those parts which could be exploited to launch DDoS attacks. The BitTorrent system is made up of the following main entities:

- *BitTorrent Client*: A client-side software that provides facilities for P2P file sharing (upload & download) via the BitTorrent protocol.
- *Leecher*: Peers that do not have the whole shared file(s).
- *Seed(er)*: A peer who owns all blocks of the file and provides pieces to other peers.
- *Torrent File*: A file which includes metadata for describing the shared files' attributes and is used for bootstrapping the download. Its main entities are: an *announce* section for specifying the address (URL) of the corresponding tracker, an *info* section that consists of the proposed names for the files, their lengths, the length of the pieces and a SHA-1 hash code for each piece.
- *Swarm:* A group of peers who join in downloading a common file.
- *Tracker:* An element which stores and manages the list of the contributing peers of a swarm together with their current communication in the BitTorrent Protocol.

4.1.2. How Does BitTorrent work?

Whenever one decides to share content over the BitTorrent network, he should first generate a corresponding torrent file which contains the tracker URL(s) and some other meta-data about the shared content. He also notifies the tracker that he is the initiator peer who shares the content described in the torrent file. The next step is to publish the produced torrent file in (popular) torrent search engine sites or discussion forums. The interested users will then be able to download the torrent file and use their BitTorrent client program to decrypt its content. The BitTorrent clients start downloading the file by connecting to the tracker periodically. As pointed out earlier, in order to have each swarm coordinated, the members should periodically declare their existence together with their current status to the tracker. The tracker has the responsibility of keeping and distributing the list of peers for different swarms individually.

The BitTorrent P2P protocol is different from the earlier peer-to-peer protocols such as FastTrack (iMesh [30], Morpheus [31], etc.) or eDonkey [32] (eMule [33], Overnet [34], etc.). One of the most significant characteristics of this fabulous protocol is that it divides each file into several smaller parts (*chunks*) in order to use peers' bandwidth more efficiently. Therefore, each peer is able to download not only from the peers which have all chunks (i.e. seeds), but also from the peers that just own some of the chunks (Figure 2). In comparison with the older versions of other P2P protocols like Napster, Gnutella, or KazaA [35], this mechanism also outstandingly lessens the load on both the content distributors and seeds.

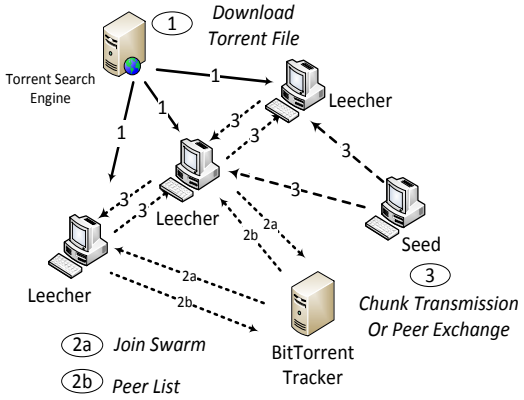

Fig. 2. How BitTorrent works [36].

4.2. Vulnerabilities and Attack Description

Before describing the BitTorrent's vulnerabilities, it is important to have a better understanding of the underlying connection in this protocol. The communications in BitTorrent can be classified into two main categories based on whether the exchange of data occurs between the peers or between the peers and the trackers. Peers communicate with each other either for the transmission of (blocks of) files or for finding out more about other existing peers by exchanging their peer list. To the best of our knowledge,

only peer-tracker connections are vulnerable to the DDoS attacks. All transmissions taking place over this type of connection are based on the HTTP protocol, and each tracker has to process hundreds of different HTTP requests in every given moment. Hence, the tracker developers have to forgo some security features for mitigating the load on the tracker. As will be discussed later in subsequent sections, an attacker may exploit some of these vulnerabilities to run a DDoS attack.

We divide BitTorrent attacks into two categories based on whether they lead to a DDoS attack or not. Authors in [36-39] have shown how it is possible to exploit this protocol in different ways to run a successful DDoS attack. Other studies have discussed several possible attacks on a BitTorrent network by exploiting the weaknesses within the BitTorrent P2P protocol. These attacks are focused on the BitTorrent members and their aim is to collapse or at least to slow down the BitTorrent network. In this paper, we discuss the ideas which may lead to DDoS attacks and the type(s) of ignorance that lay the ground for such vulnerabilities.

The very facilities that help BitTorrent become the most popular P2P application may also be exploited maliciously to turn it into a powerful DDoS engine. For example, the lack of tracker validation mechanisms by most of the popular torrent search engines could prepare for some security threats. As will be explained shortly, an attacker could exploit this vulnerability and publish fake torrent file(s) to run a DDoS attack against a target.

An interesting fact about the BitTorrent DDoS attack is that its victim could have attacked by *internal* IP address, we mean it exists inside the BitTorrent network as a peer who has the client-side BitTorrent application running; *external* IP address, on the other hand, applies to the clients outside the overlay that cannot process BitTorrent messages. It is of note that the victims of a BitTorrent DDoS attack are not limited to hosts with an internal address. It is possible to have both internal and external victims in DDoS attacks over a BitTorrent network. Table 1 comes with more details on these issues. Our upcoming discussion in the subsequent section will be according to this classification. Each attack will be evaluated separately and we will identify the weaknesses that lead to the corresponding vulnerabilities.

4.2.1. Victim as a Peer

As explained previously, a peer could have all pieces of a shared content only if it connected to the other peers asking for the blocks of shared file(s) they already have downloaded. Accordingly, there must be an entity that lets the members of each swarm know a number of participating peers to start exchanging data with. Initially,

Table 1

Different DDoS attacks over the BitTorrent network

Attack	Mode			How
Victim in place of a peer	Tracker mode (centralized)	Victim has no BitTorrent client (Outer mode)		1- The attacker make the tracker advertise the victim as a cooperative peer by sending a spoofed message to the tracker in the swarm (reported and fully implemented in [37]). 2- The attacker modifies/hacks a tracker to insert the victim's address in the peer list. (reported and fully implemented in [38]).
		Victim has a BitTorrent client (inner mode)		
	DHT mode	Victim has no BitTorrent client (Outer mode)		Sending a spoofed PING message on behalf of the victim (by using the victim's IP address) to the DHT (reported in [37], but not implemented yet)
		Victim has a BitTorrent client (inner mode)	Inner node is a tracker	Same as above but the victim is part of the BT network (not reported or implemented in the literature; just envisioned by the authors).
			Inner node is a peer	
Victim in place of a tracker	Tracker mode (centralized)	Victim has no BitTorrent client (Outer mode)		Publishing a modified torrent file with multiple trackers and having at least one entry including the IP address of the victim, also having another entry with the address of an evil tracker which answers with a fake number of seeders / leechers. (reported and fully implemented in [36]).
		Victim has a BitTorrent client (inner mode)	Inner node is a tracker	Same as above but the victim is part of the BT network (not reported or implemented in the literature; just envisioned by the authors).
			Inner node is a peer	
Amalgam mode	Both modes	Both modes		All the above.

peers know the tracker(s) (from the torrent file) but not the other peers. The tracker is the sole entity which maintains the list of participating peers for each swarm. As a result, an attacker should somehow modify the tracker's peer list to be able to introduce a victim as a peer. In the subsequent sections, we explain two possible approaches to members into connecting to the victim as a peer.

4.2.1.1. Spoofed Message Attack with a Centralized Tracker

In this model, there is a central entity which accepts queries from peers over the HTTP protocol (in CGI format). However, depending on the BitTorrent client's policies, the peers may communicate with the tracker over the TCP or UPD protocol. Once a peer establishes connection, it sends a GET request to the tracker to receive a peer list while also letting the tracker know about its status (connected/disconnected) by including its new statistics (e.g., the remaining blocks) and other parameters, as discussed previously. The tracker also returns a list of other peers who are currently sharing files in that Swarm. Since the tracker trusts the peers, it does not authenticate the requester. An attacker could thus send a spoofed message (by changing the IP address and the port number of the Peer-Tracker request message) on behalf of the victim to the tracker, announcing the victim as a participating peer (Figure 3). The tracker, believing the spoofed message, adds the fake peer (victim) to its peer list. Consequently, the members with the given victim (in place of a real peer) in their peer list, attempt to connect to the victim to ask for a block of file, without paying attention to the invalid response(s) (error messages) generated as a result. Such ignorance can be attributed to the imperfect handling of all possible exceptions by BitTorrent client developers.

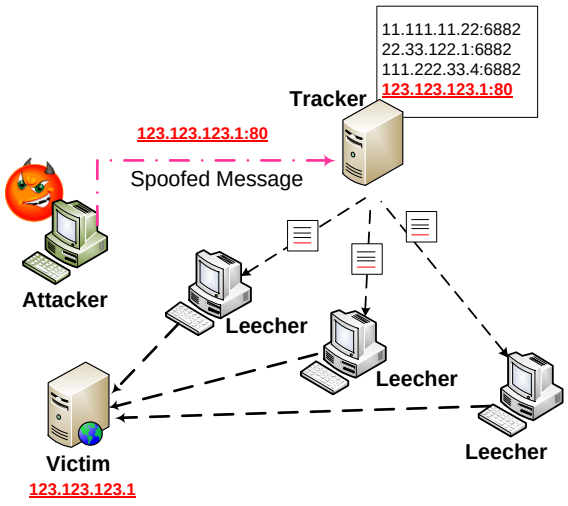

Fig.3. Attack using spoofed message.

In effect, peers simply assume that they have received a faulty message and thus should try again and again to obtain a correct response. Even if the victim does not respond to the queries issued from the P2P members, the victim's connection resources can get exhausted since most of the web services hold a connection in open state for several minutes.

This type of attack is more like index poisoning in the context of P2P DDoS attacks. In both cases, the attackers redirect the peers' queries towards the victim's machine by falsely introducing the victim as a member of the P2P network.

4.2.1.2. Spoofed Message Attack with a Decentralized Tracker

As explained before, in DHT-based networks, the tracker is not a central entity anymore. Peers (and accordingly attackers) cannot specify their own source address in the *announcement message*. Instead, since all connections in the DHT-based BitTorrent are based on UDP protocol, the source address of the *UDP announcement* message is used by the peers [37].

In DHT-based BitTorrent network, each peer sends out *Ping* messages to announce its presence to other peers. One can manipulate the source address of the *Ping* message, redirecting a huge volume of DHT query traffic towards the victim's machine. The spoofed Spoofed message in DHT-based BitTorrent is analogous to the routing table attack in the context of Overnet network.

4.2.1.3. DDoS Attack via Malicious Tracker

If the attacker owns a tracker or somehow gains the control of a tracker, he can directly change the participating peers' list by adding the victim's IP address and port number, effectively introducing it as a cooperating peer in the network. Consequently, the peers try to establish a connection with the victim for downloading the file blocks. This vulnerability is rooted in the peers' complete trust in the tracker and not validating the peers in the peer list.

4.2.2. Victim as a Tracker

In this type of attack, the attacker exploits three facts to pretend the victim is a tracker. First of all, in each swarm, the BitTorrent's members rely on a central server (tracker) for finding other participating peers and updating their own statistics. Accordingly, the peers have to contact the tracker in a periodic manner. Consequently, in comparison with the previous attack scenarios, the victim would face heavier attacks if the attacker announced it as a tracker. Secondly, most of the torrent search engines do not validate the trackers' URLs when a new torrent file is registered. They virtually accept any address in the torrent files. Additionally, no BitTorrent handshaking is performed between the peer and the tracker, although such a handshaking exists between the peers themselves.

A simple way to launch such an attack is to publish several manipulated torrent files. As shown in Figure 4, these files refer to the IP address of the victim instead of the real tracker. Not surprisingly, the deceived peers connect to the victim instead of the tracker. This can prove troublesome especially if the manipulated torrent files have a huge number of fans. However, this type of attack does not last enough since the torrent search engines do not receive valid responses from the victim and thus show zero

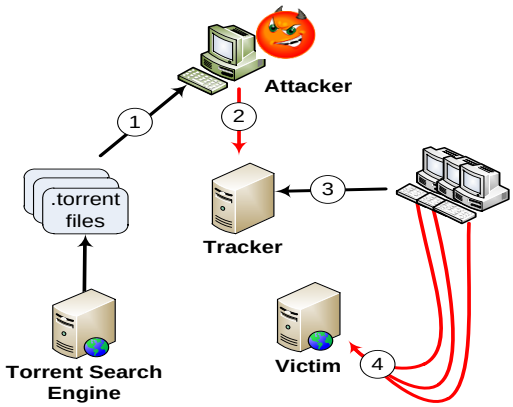

Fig. 4. DDoS attack by announcing the victim as a tracker using fake torrent files.

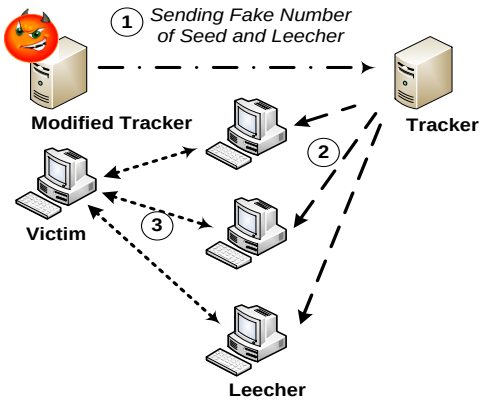

Fig. 5. DDoS attack by announcing the victim as a tracker using a Dishonest tracker.

number of leechers and seeders (*zero-sized swarm*) for the torrent file. Accordingly, the other users would not be willing to download torrent files with small swarm sizes. Also, some torrent search engines would not list a torrent file within the search results, unless the corresponding tracker reports a positive number of seeders and leechers.

Besides changing the tracker's list in the torrent file, the attacker can also employ a customized tracker to increase the effectiveness of the attack. He changes the tracker's list in such a way that the first address refers to the modified tracker while all the others are replaced with the victim's IP address. As shown in Figure 5, the duty of this customized tracker is to return a fake *swarm-size* to the torrent search engines. Further details concerning this attack can be found in [3].

4.3. Imlementation Achievements

The spoofed message attack has been reported and practically implemented in [37]. Around 1900 popular torrent files have been gathered from a popular tracker, and more than 600 connections have been made per second for about 9 hours. It is worth noting that most popular sites cannot handle more than 400 connections in every second. As argued in [37], this type of attack may also be categorized as a connection attack since the other services provided by the victim remained up during the attack time. The spoofed message attack with decentralized tracker over BitTorrent, although mentioned in [37], has not been implemented yet. The authors in [38] have discussed similar results by announcing a peer as a tracker.

5. Conclusion

In this paper, we have investigated different methods to generate unwanted traffic on third parties through misusing P2P systems. This issue needs more attention as future P2P protocols are preferably required to be designed void of such vulnerabilities. Even though some specific vulnerabilities can be mitigated and patched, it is important to recognize the inherent hazard associated with the hijacking and misdirection of P2P networks such as the generation of considerable volumes of traffic with sabotaging motives.

Acknowledgment

We would like to thank Adel Ghanbari for his comments and help during this research.

References

[1] D.D. Clark, J. Wroclawski, K.R. Sollins, and R. Braden, Tussle in cyberspace: defining tomorrow's internet, IEEE/ACM Trans. on Networking, vol. 13, pp. 462–475, 2005.

[2] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, Internet Denial of Service: attack and defense mechanisms (Radia Perlman Computer Networking and Security), Prentice Hall PTR Upper Saddle River, NJ, USA, 2004.

[3] "The NGN forum" [Online], Available: http://www.catr.cn/zhthg/ngn/2007/ [Accessed: May 2010].

[4] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, A survey and comparison of peer-to-peer overlay network schemes, IEEE Communications Surveys & Tutorials, vol. 7, pp. 72–93, 2005.

[5] Napster - Wikipedia, the free encyclopedia, May 2010, [Online]. Available: http://en.wikipedia.org/wiki/Napster [Accessed: May 2010].

[6] Gnutella - Wikipedia, the free encyclopedia, May 2010, [Online]. Available: http://en.wikipedia.org/wiki/Gnutella [Accessed: May 2010].

[7] FastTrack - Wikipedia, the free encyclopedia, May 2010, [Online]. Available: http://en.wikipedia.org/wiki/Fasttrack [Accessed: May 2010].

[8] B. Cohen, Incentives build robustness in BitTorrent, The First Workshop on Economics of Peer-to-Peer systems, Berkeley, CA, USA, June 2003.

[9] BitTorrent (protocol) - Wikipedia, the free encyclopedia, May 2010, [Online]. Available: http://en.wikipedia.org/wiki/

BitTorrent_(protocol) [Accessed: May 2010].

[10] K. El Defrawy, M. Gjoka, and A. Markopoulou, Bottorrent: Misusing Bittorrent to launch DDoS attacks, the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet (SRUTI'07), Berkeley, CA, USA, June 2007.

[11] Y. Liu, X. Liu, C. Wang, and L. Xiao, Defending P2Ps from overlay flooding-based DDoS, The Int. Conf. on Parallel Processing *(*ICPP*)*, Xian, China, September, pp. 28–28, 2007.

[12] X. Sun, R. Torres, and S. Rao, DDoS attacks by subverting membership management in P2P systems, Secure Network Protocols, pp. 1–6, 2007.

[13] X. Sun, R. Torres, and S. Rao, Preventing DDoS Attacks with P2P systems through robust membership management, Technical Report TR-ECE-07-13, Purdue University, USA, February 2007.

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

[15] Twitter - Wikipedia, the free encyclopedia, May 2010, [Online]. Available: http://en.wikipedia.org/wiki/Twitter [Accessed: May 2010].

[16] N. Friess and J. Aycock, Black Market Botnets, Jul. 2007.

[17] Z. Li, Q. Liao, and A. Striegel, Botnet Economics: Uncertainty Matters, *Managing* Information Risk and the Economics of Security, pp. 245-267, 2009.

[18] M. Ripeanu, Peer-to-Peer Architecture Case Study: Gnutella Network, University of Chicago, Department of Computer Science, 2001.

[19] E. Athanasopoulos, K.G. Anagnostakis, and E.P. Markatos, Misusing unstructured p2p systems to perform dos attacks: The network that never forgets, Lecture Notes in Computer Science, vol. 3989, p. 130, 2006.

[20] D. Zeinalipour-yazti, Exploiting the security weaknesses of the Gnutella protocol, Course Project for Seminar in Computer Security at University of California - Riverside, Department of Computer Science, March 2002.

[21] P. Maymounkov and D. Mazieres, Kademlia: A peer-to-peer information system based on the xor metric, First International Workshop on Peer-to-Peer Systems, pp. 53–65, 2002.

[22] A. Rowstron and P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, IFIP/ACM Int. Conf. on Distributed Systems Platforms (Middleware), pp. 329–350, 2001.

[23] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, Tapestry: a fault-tolerant wide-area application infrastructure, ACM SIGCOMM Computer Communication Review, vol. 32, pp. 81–81, 2002.

[24] Z. Li and X. Chen, Misusing Kademlia protocol to perform DDoS Attacks, IEEE Int. Symp. on parallel and distributed processing with applications, Sydney, Australia, pp. 80–86, 2008.

[25] J. Liang, N. Naoumov, and K.W. Ross, The index poisoning attack in p2p file sharing systems, IEEE INFOCOM 2006, Barcelona: 2006.

[26] N. Naoumov and K. Ross, Exploiting P2P systems for DDoS attacks, The first Int. Conf. on Scalable information systems (InfoScale '06), NY, USA, pp. 47, 2006.

[27] A. Belapurkar et al., Infrastructure-Level threatsand vulnerabilitie, distributed systems security: issues, processes and solutions, Wiley, pp. 71-98, 2009.

[28] H. Schulze and K. Mochalski, Internet Study 2008/2009 [Online]. Available: http://www.ipoque.com/resources/internet-studies/ internet-study-2008_2009.pdf [Accessed: October 2009].

[29] Mininova's torrent downloads double to 7 billion in a year | TorrentFreak [Online]. Available: http://torrentfreak.com/ mininovas -torrent- downloads -doubled-in-a-year-090105/ [Accessed: February 2010].

[30] iMesh Homepage [Online]. Available: http://www.imesh.com/ [Accessed: May 2010].

[31] KaZaA Homepage [Online]. Available: http://www.kazaa.com/ [Accessed: May 2010].

[32] Morpheus [Online]. Available: http://en.wikipedia.org/wiki/ Morpheus_(computer_program) [Accessed: May 2010].

[33] eDonkey network [Online]. Available: http://en.wikipedia.org/ wiki/EDonkey_network [Accessed: May 2010].

[34] Official eMule Homepage. [Online]. Available: http://www. emule-project.net/home/perl/general.cgi?l=1 [Accessed: May 2010].

[35] Overnet [Online]. Available: http://en.wikipedia.org/wiki/ Overnet [Accessed: May 2010].

[36] K.C. Sia, DDoS vulnerability analysis of Bit-Torrent protocol, Technical Report, UCLA: 2006.

[37] K. El Defrawy, M. Gjoka, and A. Markopoulou, Bottorrent: Misusing bittorrent to launch ddos attacks, USENIX SRUTI, Santa Clara, 2007.

[38] J. Harrington, C. Kuwanoe, and C.C. Zou, A BitTorrent-driven distributed denial-of-service attack, 3rd Int. Conf. on Security and Privacy in Communication Networks, Nice, France: , pp. 17–20.

[39] N. Naoumov and K. Ross, Exploiting p2p systems for ddos attacks, Proc.s of the 1st Int. Conf. on Scalable information systems, Hong Kong, pp. 47-53, 2006.