BotTorrent: Misusing BitTorrent to Launch DDoS Attacks

Karim El Defrawy*, Minas Gjoka and Athina Markopoulou
University of California, Irvine
{keldefra, mgjoka, athina}@uci.edu

Abstract

BitTorrent is currently one of the most popular peer-to-peer systems. BitTorrent clients are widely spread all over the world and account for a large fraction of today's Internet traffic. In this paper, we show that BitTorrent can be exploited by misdirecting clients to send their traffic toward *any* host on the Internet. The volume of a BitTorrent swarm can thus be converted into firepower for launching a distributed denial-of-service attack that can exhaust the victim's resources, including access bandwidth and connection resources. We identify novel exploits of the BitTorrent system and conduct real-life experiments that demonstrate the feasibility and severity of such attacks. We characterize the volume, duration and spread of attack traffic observed in our experiments. Finally, we discuss possible fixes and the limits of both attack and defense approaches.

1 Introduction

Several things have changed since the original design of the Internet. The Internet has evolved from a small scientific network carrying data between trusted computers to the ubiquitous communications infrastructure carrying all types of traffic including data, voice, video and financial transactions. In the new environment users may have conflicting interests [1] and/or malicious intentions, launching various kinds of attacks against innocent hosts and/or the networking infrastructure. We are interested in a particular type of attacks, namely distributed denial-of service (DDoS) attacks, where a large number of compromised machines coordinate and send traffic toward a victim host thus exhausting its resources and disrupting its normal operation [2].

One of the main mechanisms used today to gain control over a large number of machines is to infect them with a malicious program that takes instructions from the attacker via some communication channel (e.g. IRC) [2]. Another mechanism is to embed the program into a worm and launch it over the Internet to infect a large number of hosts. The Internet has witnessed such large-scale worms in recent years, including Code-Red [3] and Slammer [4].

In this paper, we explore a new way of launching DDoS attacks by hijacking peer-to-peer (P2P) systems. These systems recently became very popular for distributing content to a large number of users. Given the already large population of P2P clients (some claim that P2P traffic constitutes up to 60% of Internet traffic [5]) even a small amount of traffic or connections per user leads to an aggregate that can flood any victim. This type of attack can be quite powerful as it does not require any special infection process or special software to be installed and is very hard to trace.

To the best of our knowledge, this is the first extensive study of BitTorrent-based [7] DDoS attacks against *any* victim host (i.e. the victim does not have to participate in the BitTorrent swarm). We identify vulnerabilities in the design of BitTorrent that can be exploited to use the system as a platform for launching DDoS attacks. We conduct real-life experiments that demonstrate the feasibility of such attacks against our own victim machine at UCI. We keep logs of these attacks and analyze several characteristics of interest including the volume, duration and spread of the attack traffic. Finally, we discuss potential fixes and solutions to these vulnerabilities.

The structure of the rest of the paper is as follows. Section 2 gives an overview of the BitTorrent system and its operation. Section 3 presents the vulnerabilities in BitTorrent that can be exploited to turn it into a platform for launching DDoS attacks. Section 4 presents the results of real-life experiments of such attacks on the Internet. Section 5 discusses directions for fixing the identified weaknesses. Section 6 discusses related work. Section 7 presents future work and concludes the paper.

2 Overview of BitTorrent

In this section, we give a brief overview of the BitTorrent system, with emphasis on those parts that we later exploit to launch DDoS attacks. The system consists of the following main entities:

- *Torrent File:* It contains meta data describing the files to be distributed and is used for bootstrapping the download. Typically, it includes an "announce" section, which specifies the URL of the tracker, and an "info" section that contains suggested names for the files, their lengths, the piece length used and a SHA-1 hash code for each piece [7].

*This work has been partially supported by the Center for Pervasive Communications and Computing (CPCC) at UC Irvine, and by Emulex.

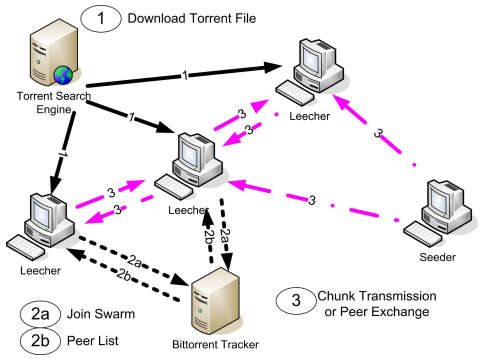

Figure 1: Typical operation of BitTorrent

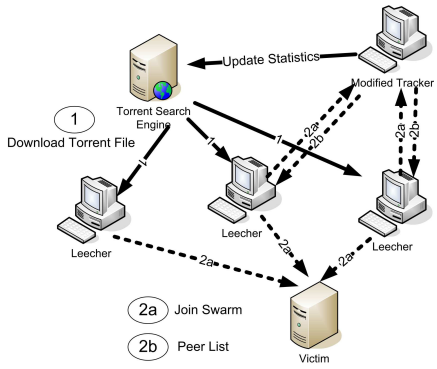

Figure 2: Using BitTorrent to generate a DDoS attack

- *BitTorrent Client:* A program that implements the BitTorrent protocol, allows a host to download/upload the files described in the torrent file from/to other peers. The term seeder is used for peers that already have the whole file and therefore participate only by uploading to other peers whereas the term leecher is used to describe peers which have not yet downloaded the whole file. The collection of seeders/leechers for one torrent is called a swarm.

- *Torrent Web Sites and Search Engines:* These websites publish the torrent files and help users locate them.

- *Tracker(s):* These are hosts responsible for coordinating the file distribution among peers. They keep track of clients downloading a certain file and direct new peers to other peers that have the file or pieces of it.

Fig. 1 shows the typical sequence of operations in BitTorrent while in centralized tracker mode. First, users browse the web to find a torrent file of interest (step 1). This file can be obtained through well-known torrent search engines or by any other means such as personal communication or web forums. Once the user finds the torrent file, he/she downloads it and opens it with a BitTorrent client. The client then connects to the tracker listed in the torrent file. The tracker then provides a sublist of peers currently downloading the file(s) (steps 2a and 2b). Once the client obtains the addresses of other peers in the swarm, it starts downloading pieces of the file in parallel (step 3). BitTorrent can also operate in a distributed hash table(DHT) mode [6], but in this paper we only focus on the centralized tracker mode of operation.

3 Vulnerabilities in BitTorrent

Several of the features that make BitTorrent popular and powerful can be maliciously exploited to turn it into a DDoS platform. For example, the openness of trackers and torrent search engines allows anybody to publish a torrent easily and without authentication but can also cause security threats. There are several attack methods that could be used to launch DDoS attacks using BitTorrent. There are three main types of such attacks, and also combinations of them as shown in Table 1.

The simplest attack is *type 1* in Table 1, which was first described in [10]: sending a spoofed message to the tracker announcing the victim as a participant in the swarm. However, we observed that this attack is less severe than an attack of *type 2* and has an easy fix, as discussed in section 6. For the rest of the paper, we focus on attack *type 2* in Table 1 because it is the most potent. Attack *type 2* reports the victim as one of the trackers. It exploits the fact that BitTorrent relies on central trackers for finding the participating peers and for directing a downloader to different pieces of a file. This requires all clients to contact the tracker at regular intervals and thus can be used to launch a DDoS attack by manipulating the clients to believe that the victim hosts a tracker. Another reason that this attack works is that there is no BitTorrent handshake between the peer and the tracker, although such a handshake exists between peers. This results in peers not recognizing that the victim (which they believe is a tracker) is not running BitTorrent at all.

The easiest way to launch such an attack would be to publish a torrent file that contains the IP address(es) of a victim as the main tracker or as a list of trackers. However, this would not be very effective since the statistics for this torrent on torrent web sites would show a swarm size of zero, since no valid responses can be received from its trackers. This would discourage further participation from the majority of users. Some sites won't even show the torrent in their listings unless the tracker reports a positive number of seeders and leechers. [1]

A more subtle and, as it turns out, more effective attack (type 2 in Table 1) exploits the multi-tracker feature [11] recently introduced to BitTorrent. Allowing multi-trackers

[1] It is interesting to report that we have published torrents without reporting a positive number of seeders/leechers and they were still downloaded by a large number of users. This has two explanations: either these are users who download the torrents hoping that later on the number of seeders/leechers will increase; or these are automated bots downloading any torrent published irrespective of its statistics. For example, these could be organizations that are tracking down illegal copies of movies and music files on the Internet, as also reported in [9] and [8]. Interestingly, we were contacted by some of these organizations, during our experiments.

Table 1: Different Attack Methods

	Attack Method	BitTorrent Mode	Requirements
1	Report victim as a participating peer	Centralized Tracker Mode	Send a spoofed message to the tracker announcing victim as a participating peer in the swarm (mentioned and implemented in [10]). Or if one of the trackers is compromised, include the victim's address in the peer list.
2	Report victim as a tracker	Centralized Tracker Mode	Publish torrent file with multiple trackers. At least one entry contains the address of the victim. Another entry contains the address of a modified tracker, which replies with a fake number of seeders/leechers
3	Report Victim as Peer in DHT	DHT Mode	Send a spoofed `PING` message to the DHT, including the victim's source IP (mentioned in [10], but not implemented)
4	Combine 1,2, 3	both modes	All requirements of 1,2, 3

in the torrent improves reliability, resilience to failure and allows load balancing among several trackers. In addition to a fake torrent file, the attack also requires a machine that runs a modified BitTorrent tracker. The fake torrent file lists multiple trackers, the first of which is the modified tracker while the rest contain the victim's IP address. The modified tracker should respond with (fake) high statistics to requests from the sites where the torrent was published, thus making it appealing for users to download. Fig. 2 outlines the attack steps explained above. Note that the victim can be *any* machine on the Internet, and does not need to be participating in the swarm; it is sufficient that the torrent file lists the victim's IP as one of the trackers.

4 Internet Experiments

In this section we describe our Internet experiments with attack *type 2* in Table 1. This is the most effective attack and also the main contribution of this paper.

4.1 Experiment Setup

We created a list of popular titles by parsing the web sites of known torrent search engines. We then generated random files to match the file sizes of these titles and torrent files for each of them. We published these torrent files on well known torrent search engines. These torrent files contain a tracker list which includes first one entry with the address of our *modified BitTorrent tracker*, and then multiple (IP:Port) entries of our *victim* machine. Clients that downloaded these torrents initially tried aggressively to contact all trackers in the list. After a client established a connection to the first tracker, which is our modified BitTorrent tracker, it obeyed the update interval (announce interval) in that tracker's response message. This interval determines the frequency of connections to the rest of the trackers in the list and therefore directly affects the severity of the attack. We set a small value for the announce interval (less than 30 seconds). This triggered a large number of client requests toward all IP addresses in the tracker list. As a result the modified tracker was also DDoS-ed alongside the victim. This further forced all clients to contact the victim machine more aggressively in hopes of contacting an operational tracker.

We tested the attack extensively toward one open HTTP port and a large number of closed ports. We omit the results from other commonly used services, such as FTP, SMTP, SAMBA and SSH, due to space constraints. Attacking open ports increases the attack effectiveness because TCP connections are kept alive for longer periods and clients send larger packets beyond the TCP handshake.

To maintain the public interest in our torrents (to have large swarms), it is necessary to seed their files. The seeding rate can be as low as one Kbps. Another option is to create specially crafted files for the chosen titles. These files should contain chunks of zero bits so that they match the file initialization of most clients. The clients then wrongfully believe that these zero bits are parts of the file that were successfully downloaded. This leads users behind these clients to believe that these files are being downloaded successfully.

As a proof of concept, we conducted several experiments, launching small scale attacks against our own victim machine at UCI, an Intel Xeon 2.6 GHz running Debian GNU/Linux. The machine has 4GB RAM and is connected to the network via a 100Mbps Ethernet interface. The experiment scale was kept small on purpose to avoid potential interference with the normal operation of BitTorrent and UCI's network. The incoming traffic was logged using tcpdump and analyzed. The parameters we varied in different experiments include the number of torrents and the number of open/closed ports attacked. In the last experiment (IV) we also made the tracker report extra peer entries, with the victim's IP address included several times in the peer list sent to the clients (attack type 1 in Table 1). Table 2 summarizes the setup (number of torrents and ports used) and results (measurements related to the resulting attack) for each experiment.

4.2 Results and Analysis

For every experiment in Table 2 we analyzed the logged packet traces and looked at several characteristics of the DDoS attack launched against our own victim machine. Due to lack of space, we are presenting results only from the last experiment (Experiment IV). In particular, we characterize the following properties of the DDoS attack that we measured during experiment IV: (i) attack volume in terms

Table 2: Summary of Experiments: Setup and Results during the first 56 hours.

Exp. #	# Torrents	Ports Attacked		Throughput(Kbps)		Total Unique	TCP Conn.	New Host Avg
		Open (Freq)	Closed	Avg [a]	Max[a]	# Hosts	Avg/sec	Interarrival Time (sec)
I	10	1 (1)	6	62.77	127.28	25,331	753.93	7.89
II	25	1 (10)	10	137.78	252.40	55,127	1400.74	3.62
III	25	1 (1)	501	132.97	538.38	86,320	1580.88	2.31
IV	25	1 (50)+1(1)	49+201	176.69	482.81	58,046	1440.17	3.44

[a]Excluding the initial transient period (6 hours) of the experiment

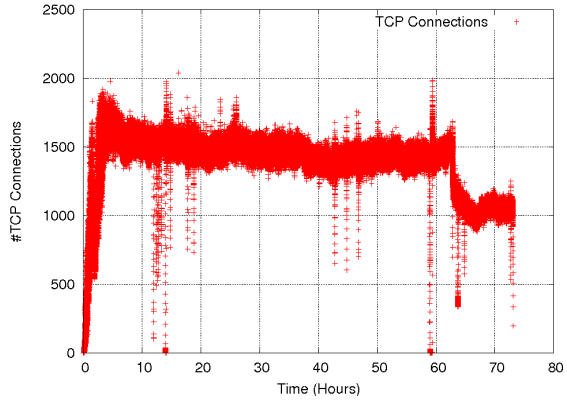

Figure 3: Number of TCP connections (per second) over time

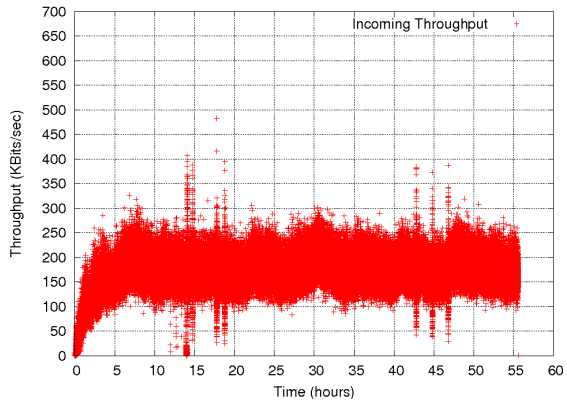

Figure 4: Attack traffic (incoming throughput, calculated in 1 sec intervals) over time.

of number of TCP connections (Fig. 3), aggregate attack bandwidth (Fig. 4) and packet sizes (ii) spread of attack traffic among different sources (Fig. 6) and among different subnets (Fig. 7).

Attack Volume. Fig. 3 shows the number of attempted and open TCP connections per second at the victim. In a couple of hours the attack ramps up and reaches up to 1800 TCP connections. Interestingly, this high rate of connections sustains for about 3 days with an average rate of 1400. This translates to a steady incoming attack throughput as shown in Fig. 4. Note that this duration is not a function of how long we run the experiment because most torrents were removed after one day from the websites. But we still received traffic for a couple of days.

The results from all experiments show that with only 25 torrents we caused an average attack rate of 137-176 Kbps (and a maximum of 252-538) that lasted for more than two days. To put things in perspective, [10] had to use $1,119$ torrents to generate an attack of only $1.5Mbps$. Recall that we kept the attack volume low on purpose during these proof-of-concept experiments. However, it is not difficult to see how this attack could scale up, e.g. if one automated the process and launched a large-scale attack with hundreds or even thousands of torrents. Furthermore, imagine an attack launched by bots using the BitTorrent infrastructure. Each zombie machine in the botnet could self-initiate a small scale attack, such as those in experiments II, III, IV (with itself as the modified tracker), against the victim so as to make such an attack even more distributed. A botnet with n zombies will launch n such independent attacks. We observed practically no overlap (only 1.5%) between the attacking hosts in different experiments (I,II, III, IV). Given the capabilities of today's botnets, the resulting aggregate attack would easily throttle links much larger than E1/T1.

Over 95% of the received packets were small TCP handshake packets ($40-45$ Bytes without the Ethernet header), because a considerable number of ports on the victim machine attacked in experiment IV were closed ports as seen in table 2. We point out that what makes the attack powerful is the large number of hosts that could be achieved and not the packet sizes.

Finally, Fig. 5 shows the percentage of attack traffic received at different ports. As expected, open ports receive more traffic for the reasons discussed in the previous section. Comparing the traffic received in closed ports, we make two interesting observations: (i) all attacked ports in the same category receive more or less the same traffic (ii) attack type 2, as described in table 1, is one order of magnitude more powerful than attack type 1.

Attack Spread. The next question is how much attack traffic is contributed by different attack sources, and how these attack sources are spread on the Internet. Fig. 6 shows the contribution of different sources to the total volume of attack traffic. We can see that not all hosts contribute similar amounts of traffic i.e. 80% of the hosts generate 10% of the traffic (and therefore the remaining 20% of the hosts generate 90% of the traffic). However, this does not necessarily mean that this attack can be easily handled by filtering out a few bad-behaving IPs. In experiment IV, 20% of the hosts translates to more than 11,500 hosts in absolute num-

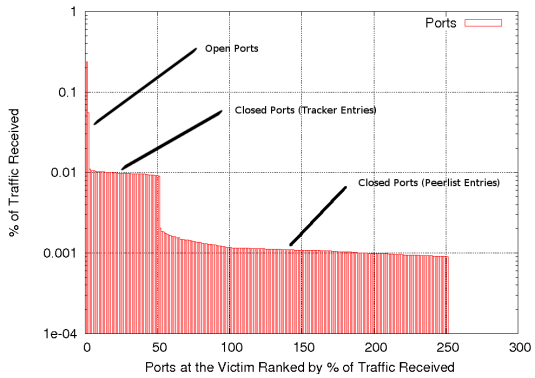

Figure 5: % of traffic received in all attacked ports

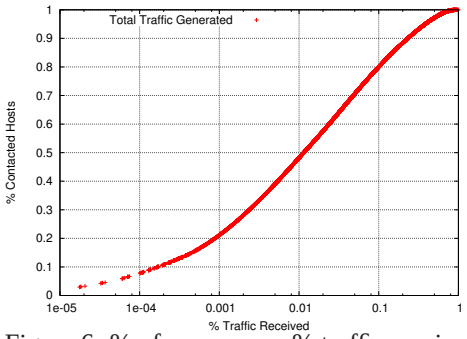

Figure 6: % of sources vs. % traffic received

bers; this number will be even larger if a large-scale attack is launched with hundreds of torrents.

Fig. 7 shows the number of attack hosts per different class A, B and C networks. We observed that 58,046 unique IP addresses contacted our victim machine. About 87% of them are in different class C addresses and 12% of them have different class B networks. The large number of different networks observed confirms our hypothesis that using BitTorrent as an attack platform yields a very distributed DDoS attack. Filtering at the victim's gateway based on the source-IP address at the gateway would not be effective in this case.

We also analyzed the structure of the attack graph. We used traceroute to identify the routers on the paths from the attack sources to our victim. We found that attack sources were located as far as 30 hops away from the victim; 50% of them were within 16 hops away and 90% of them were up to 20 hops away. Analyzing the TTL field of incoming packets also revealed that attackers were on average 17 hops away from the victim. We then looked at nodes/routers at different levels of the attack graph (distance from the victim) and characterized the node degree at different levels. We obtained the statistics for node degree at each level (node degree was larger closer to the victim) and other properties, which we have to omit here for lack of space. We are currently working on developing a model for the DDoS attack graph based on our measurements. This model could be used by the research community to evaluate defense mechanisms (e.g. filtering) against such a DDoS attack.

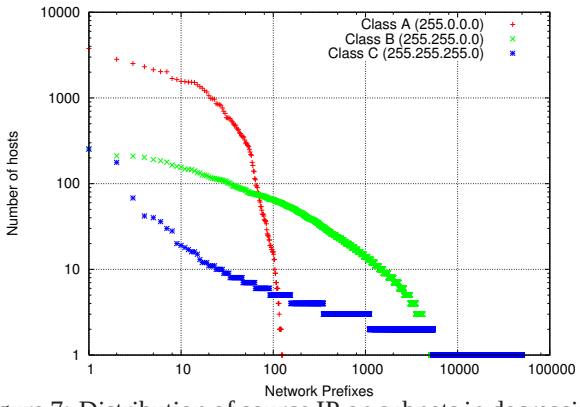

Figure 7: Distribution of source IP on subnets in decreasing order

5 Fixes and Solutions

The main solution to the attack presented in this paper is to have clients parse the response from the tracker. In the case where a host (tracker) does not respond to a peer's request with a valid BitTorrent protocol message it should be inferred that this host is not running BitTorrent. The peer should then exclude that address from its tracker list, or set a high retry interval for that specific tracker. Another fix would be for web sites hosting torrents to check and report whether all trackers are active, or even remove the non-responding trackers from the tracker list in the torrent. Another measure could be to restrict the size of the tracker list to reduce the effectiveness of such an attack. On the downside, these changes may cripple the functionality that the multi-tracker extension was meant to provide, such as load balancing and backup service. Another approach is to avoid user controlled trackers. Some web sites already replace the tracker address in the torrent file with their own controlled tracker. In those cases, the web sites have to deploy several trackers and ensure that they can sustain the load of tracking large numbers of torrents. In all cases, one would have to trade-off openness and scalability, which are the main features that make P2P systems attractive in the first place, for security.

On the client side, BitTorrent clients could detect such malicious torrents by analyzing data about the swarm. For example, if the torrent is malicious peers will have exactly the same pieces of the file (if any); the state of the swarm will remain unchanged for long periods of time; most trackers will be unresponsive, etc.

To prevent the automation of publishing fake torrents, which could dramatically multiply the volume of BitTorrent-based DDoS attacks, we recommend using a reverse Turing test when uploading a torrent to a web site. A few of the torrent search engines already have this implemented.

6 Related Work

The use of BitTorrent as a platform for launching DDoS attacks against any host on the Internet has received little attention so far. Similar ideas have been explored in the context of earlier P2P systems, such as Gnutella [8] and Overnet [9]. In [9] the authors consider two types of attacks on P2P, namely index poisoning and routing table poisoning. The attack we discuss here falls under the category of index poisoning (except that the index is that of the tracker not a peer).

Today, BitTorrent has a much larger user base, and therefore a DDoS attack launched by BitTorrent clients has the potential for more firepower and damage. Our results indicate that an attack generated using BitTorrent can be much larger than what was reported in [9] (300 TCP connections per second and aggregate traffic of 1.6 Mbps by faking announcements corresponding to 7,564 files hashes) if thousands of torrents are also used.

The only work we are aware of that uses the BitTorrent system to launch DDoS attacks is [10]. [10] used a type 1 attack (according to our classification in Table 1) with 1,119 torrents and diverted 30,514 unique IP addresses to contact the victim. The contribution of this paper is the identification and analysis of attack type 2. In the experiments, we used only 25 torrents and diverted traffic from over 58,000 unique IP addresses. A limitation of [10] is that not all tracker implementations accept the field required to send the spoofed message. This attack can also be easily solved by checking if the source IP address in the packet header is the same as the announced one in the BitTorrent message, but this interferes with clients behind NATs.

7 Conclusion and Future Work

In this paper, we showed that it is possible to launch a DDoS attack against *any* machine on the Internet by diverting BitTorrent traffic. As a proof-of concept, we demonstrated the feasibility of such an attack through real-life experiments. Although we purposely kept our experiments small and simple, the resulting attack is large enough to deny service to small organizations and home users. Part of our future work is to study large-scale BitTorrent-based DDoS attacks. We discuss some modifications to the BitTorrent protocol to prevent such attacks. Even if specific exploits are fixed, it is important to recognize the inherent danger of hijacking and misdirecting large volumes of legitimate traffic for malicious purposes. Such considerations should become essential to the design of P2P systems.

Acknowledgements

We would like to thank the Network and Academic Computing Support (NACS) as well as the Networking Lab in CalIT^2 for their support and some of the resources used in these experiments. We would also like to thank CPCC and Emulex for supporting this work with fellowships, and Michael Sirivianos for suggesting the name BotTorrent.

References

[1] D. Clark, J. Wroclawski, K. Sollins, R. Braden, "Tussle in cyberspace: defining tomorrow's Internet", *in IEEE/ACM ToN*, vol.13(3), June 2005, pp. 462-475.

[2] J. Mirkovic, S. Dietrich, D. Dittrich, P. Reiher, "Internet Denial of Service: Attack and Defense Mechanisms", *Prentice Hall*, 2005.

[3] D. Moore, C. Shannon, J. Brown, "Code-Red: a Case Study on the Spread and Victims of an Internet Worm", *in Proc. ACM IMW 2002.*

[4] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, N. Weaver,"Inside the Slammer Worm", *in Proc. of IEEE Security and Privacy*, 2003.

[5] "Cache Logic", *http://www.cachelogic.com/*.

[6] "BitTorrent DHT protocol Specs," *http://www.bittorrent.org/*.

[7] B. Cohen, "BitTorrent Protocol Specs", *http://www.bittorrent.org/*.

[8] E. Athanasopoulos, K. Anagnostakis, E. Markatos, "Misusing Unstructured P2P systems to Perform DoS Attacks: The Network That Never Forgets", *in Proc. ACNS* 2006.

[9] N. Naoumov, K. Ross, "Exploiting P2P Systems for DDoS Attacks", *in Proc. of INFOSCALE*, 2006.

[10] K. Cheung Sia, "DDoS Vulnerability Analysis of BitTorrent Protocol", *UCLA Tech. Report*, Spring 2006.

[11] "BitTorrent Multi-Tracker Protocol Specs," *http://www.bittornado.com/docs/multitracker-spec.txt*.

[12] "List of BitTorrent Search Engines and Open Trackers", *http://www.zeropaid.com/links/bittorrent/*.

[13] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy and M. Faloutsos, "Is P2P dying or just hiding?", *in Proc. IEEE Globecom* 2004.

[14] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garces-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime", *in Proc. PAM*, 2004.

[15] A. Iosup, P. Garbacki, J.A. Pouwelse, D.H.J. Epema, "Analyzing BitTorrent: Three Lessons from One Peer-Level View ", *in Proc.* 11^{th} *ASCI Conf.*, 2005.

[16] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips, "The BitTorrent P2P File-Sharing System: Measurements and Analysis", 4^{th} *IPTPS*, 2005.

[17] N. Liogkas, R. Nelson, E. Kohler, L. Zhang, "Exploiting BitTorrent For Fun (But Not Profit)", 5^{th} *IPTPS*, 2006.

[18] T. Locher, P. Moor, S. Schmid and R. Wattenhofer, "Free Riding in BitTorrent is Cheap", *HotNets V*, 2006.

[19] M. Sirivianos, J. Han Park, R. Chen and X. Yang, "Free-riding in BitTorrent Networks with the Large View Exploit", *IPTPS*, 2007.