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 (P2P) systems: its 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 (DDOS) 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 as well as 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). 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].

*Contact author: *keldefra@uci.edu*; tel: (949) 824-1637; fax: (949) 824-1637; mail: 4100 Calit2, Univ. of California, Irvine, CA 92697.

In this paper, we explore a new way of launching distributed denial-of-service (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 it 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, spread of attack traffic. Our measurements capture realistic DDoS attacks, that can be used as input to other studies that need attack models to design and evaluate defense approaches. Finally, we discuss potential fixes of these vulnerabilities as well as design principles that can help P2P systems become more robust to such attacks.

The structure of the rest of 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 in 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 metadata describing the files to be distributed and it is used for bootstrapping the

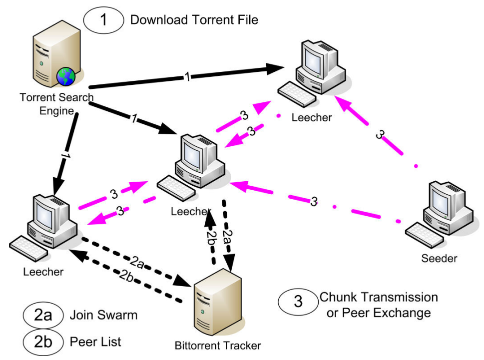

Figure 1: Typical operation of the BitTorrent protocol

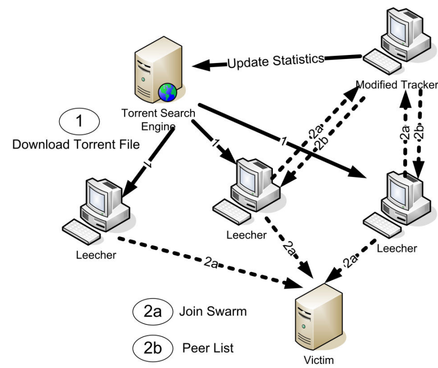

Figure 2: Using BitTorrent to generate a DDoS attack

download. Typically, it includes an "announce" section, which specifies the URL of the tracker, and an "info" section which contains suggested names for the files, their lengths, the piece length used, and a SHA-1 hash code for each piece [7].

- *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:* They publish the torrent files and help users locate them.

- *Tracker(s):* These are hosts responsible for coordinating the file distribution among peers. [1]

Fig. 1 shows the typical sequence of operations in BitTorrent. 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).

3 Vulnerabilities in BitTorrent

Several of the features that make BitTorrent popular and powerful can also be maliciously exploited to turn it into a DDoS platform. For example, the openness of trackers and torrent search engines allow anybody to publish a torrent easily and without authentication, but can also cause security threats. We identify 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, shown in Table 1.

The simplest attack is *type 1* in Table 1, which was first described in [10]: send 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 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. Attack *type 2* reports the victim as one of the trackers. It exploits the fact that, although content delivery is distributed, BitTorrent still 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 by manipulating the clients to believe that the victim hosts a tracker.

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 will not be very effective since the statistics for this torrent on torrent web sites will show a swarm size of zero, since no valid responses can be received from its trackers; this will 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. [2]

A more effective attack (type 2 in Table 1) exploits the multi-tracker feature. In addition to a fake torrent file it also

[1]A recent extension to BitTorrent introduced the use of multiple trackers [11], to improve reliability, resilience to failure and allow load balancing of a single tracker among several trackers. Although the main way for learning about other peers in the swarm is still through the tracker(s), additional ways include distributed hash tables (DHT) [6] and peer exchange (PEX).

[2]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 authorities, 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 [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 [10]
4	Combine 1,2, 3	both modes	All requirements of 1,2, 3

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 make 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. (We omit the results of the other types of attack due to lack of space.)

4.1 Experiment Setup

We implemented the attack as follows. 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 size of these titles and torrents files for each of them. We created a torrent file [3] that contain a tracker list which includes first one entry with the address of our *modified BitTorrent tracker* and then multiple [4] (IP:Port) entries of the *victim*. Initially, until the client connects to at least one tracker, an aggressive request behavior was observed towards the available trackers (most of which contain the victim's IP address). After a client has established a connection to the first tracker, which is our modified BitTorrent tracker, it will have to obey the update interval, also called announce interval, dictated by the tracker's response message. Since this interval determines the frequency of connections to the trackers and therefore directly affects the perceived attack we set a small value of announce interval (less than 30 seconds). This triggers a multitude of frequent client requests towards all IP addresses in the tracker list. As a result the modified tracker will also be DDoS-ed alongside the victim.This further forces all clients to contact more aggressively the victim machine in hope of contacting an operational tracker.

We tested extensively the attack towards 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 since TCP connections are kept alive for longer periods and clients send larger packets beyond the TCP handshake.

To maintain the interest in each torrent and thus a large swarm size it is necessary either to seed the file with a rate even as low as one Kbps or create specially crafted random files for the chosen titles (i.e. some chunks full of zero bits which will be perceived as downloaded chunks in most clients, due to common memory initialization techniques used, and further propagated in the system).

As a proof of concept, we conducted several experiments, launching small scale[5] 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 incoming traffic was logged using tcpdump and further analyzed. 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 several entries with the victim IP address 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 we looked at several characteristics of the DDoS attack launched against our own victim machine.

[3] All torrents were injected in a small number of popular torrent search engines. We only needed to upload the torrents in well known torrent search engines since we have observed that less known torrent sites replicate the contents of the former. As a result our injected torrents spread around the Internet very fast (within a couple of hours) and stayed around for much longer (weeks).

[4] The multiple entries with the victim address are inserted to take advantage of the greedy implementation of the multi-tracker feature in most BitTorrent clients. Some clients initiate connections to all trackers present in the torrent in random order within a small time interval (seconds) so as to maximize the swarm view available to them.

[5] The scale was kept small on purpose to avoid potential interference with the normal operation of BitTorrent, especially since we have no control over the torrents once published

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

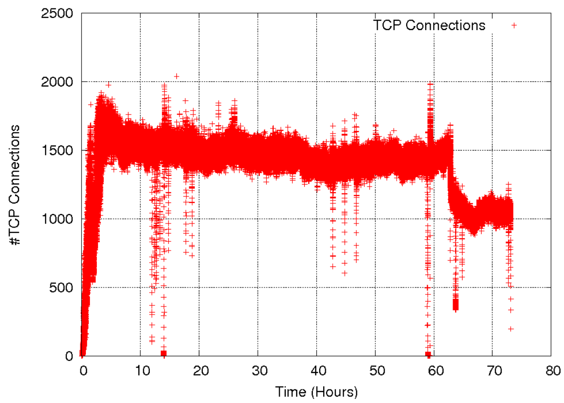

Figure 3: Number of the TCP connections (per second) over time

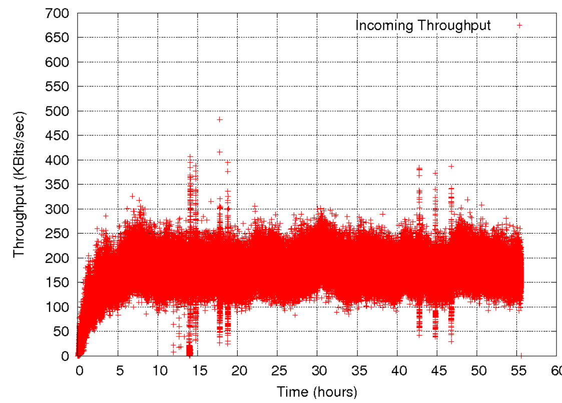

Figure 4: Attack traffic (incoming throughput, calculated in 1 sec intervals) over time.

Due to lack of space, we are going to present results only from the last experiment (Experiment IV). In particular, we will characterize the following properties of the DDoS attack that we measured during experiment IV: (i) attack volume in terms of number of TCP connections (Fig. 3), aggregate attack bandwidth (Fig. 4) and packet sizes (ii) spread of attack traffic to different sources (Fig. 6)) and to 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 keeps up for as long as 3 days with an average rate of 1400. This translates to a steady incoming attack throughput as shown in Fig. 4.

The results from all experiments show that with only 25 torrents we caused an average attack rate of 137-176 (and a maximum of 252-538) Kbps, that lasts 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.[6]. 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 independent[7] such attacks. 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. Nevertheless, a small increase in the average packet size could considerably multiply the strength of the attack.

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 in the Internet. Fig. 6 shows the contribution of different sources to the total volume of

[6]We do not attempt to make a projection about the volume of such an attack, as the effect of increasing the torrents is not linear. We just note that, in our experiments, and increase in the number of torrent by 2.5 lead to an increase in attack traffic by 2.8

[7]We observed practically no overlap (only 1.5%) between the attacking hosts in different experiments (I,II, III, IV)

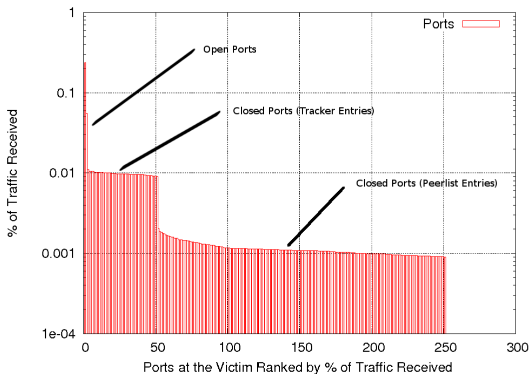

Figure 5: % of traffic received in all attacked ports

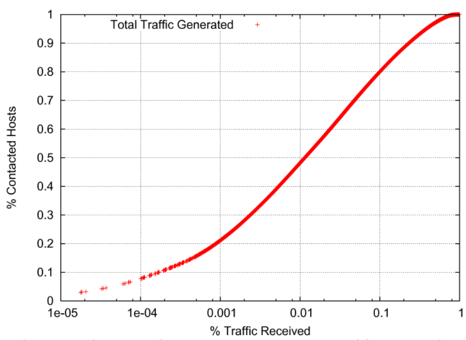

Figure 6: % of sources vs. % traffic received

attack traffic. We can see that not all hosts contribute similar amount 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 numbers; this number will be even larger in 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.

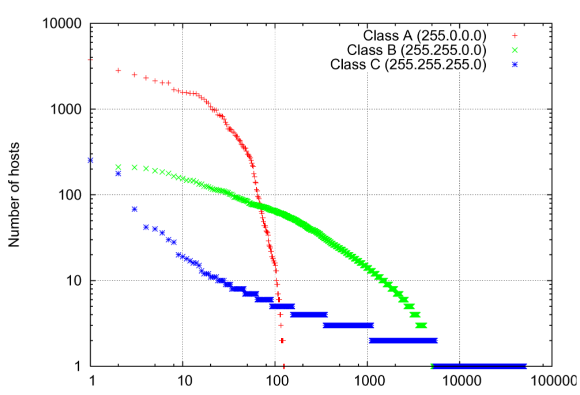

Figure 7: Distribution of source IP on subnets in decreasing order

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. 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.

5 Fixes and Solutions

There are measures that the BitTorrent system could take to avoid being misused as a platform for launching DDoS attacks. Web sites hosting torrents could 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. There are also closed communities for torrent exchange, that require subscription and restrict participation to members; these are clearly less vulnerable to the attacks we explored here. In all cases, we would have to trade-off openness and scalability (which are the main features that make P2P systems attractive at 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

BitTorrent has received a lot of attention [14] [13] [15] [16]. Some researchers have also looked at attacks against BitTorrent itself [17]. However, the use of BitTorrent as a platform for launching DDoS attacks against any host in 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. Our attack used only 25 torrents and a type 2 attack 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 clients. As a proof-of concept, we demonstrated the feasibility of such attacks through real-life experiments. Although we purposely kept our experiments small and simple, the resulting attacks are large enough to deny service to small organizations and home users. Part of future work is to study large scale BitTorrent-based DDoS attacks. We propose modifications to BitTorrent 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. As a side contribution, this paper provided a characterization of the attacks observed in our experiments, including statistics on the volume, duration, spread of attack traffic. Apart from demonstrating the spread and potential severity of the attacks, these data can be used as input to various studies that need attack models to design and evaluate defense approaches.

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] Ka Cheung Sia, “DDoS Vulnerability Analysis of BitTorrent Protocol”, *UCLA TR*, 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.