P2P File-Sharing in Hell: Exploiting BitTorrent Vulnerabilities to Launch Distributed Reflective DoS Attacks

Florian Adamsky
City University London
florian.adamsky.1@city.ac.uk

Syed Ali Khayam
PLUMgrid Inc.
akhayam@plumgrid.com

Rudolf Jäger
THM Friedberg
rudolf.jaeger@iem.thm.de

Muttukrishnan Rajarajan
City University London
r.muttukrishnan@city.ac.uk

Abstract

In this paper, we demonstrate that the BitTorrent protocol family is vulnerable to distributed reflective denial-of-service (DRDoS) attacks. Specifically, we show that an attacker can exploit BitTorrent protocols (Micro Transport Protocol (uTP) [32], Distributed Hash Table (DHT) [30], Message Stream Encryption (MSE) [8]) and BitTorrent Sync (BTSync) [6] to reflect and amplify traffic from peers. We validate the efficiency, robustness and evadability of the exposed BitTorrent vulnerabilities in a P2P lab testbed. We further substantiate the lab results by crawling more than 2.1 million IP addresses over Mainline DHT (MLDHT) and analyzing more than 10,000 BitTorrent handshakes. Our experiments reveal that an attacker is able to exploit BitTorrent peers to amplify the traffic up to a factor of 50 times and in case of BTSync up to 120 times. Additionally, we observe that the most popular BitTorrent clients are the most vulnerable ones.

1 Introduction

DDoS attacks continue to become increasingly devastating, despite widespread adoption of mechanisms to circumvent IP spoofing[1]. In 2013, CloudFlare registered a DDoS bandwidth record by an attack which generated nearly 300 Gbps traffic [36]. A year later, a new record was established by a DDoS attack that generated 400 Gbps [37]. Both these record-setting attacks belonged to a category of DoS attacks where the attacker does not send traffic directly to the victim; traffic is instead sent to *reflectors* (with spoofed source IP of the victim) which in turn flood the victim with responses. Such a DRDoS attack becomes particularly potent if the reflectors send higher volumes of traffic to the victim than what they received from the attacker—i.e. the reflectors act as *amplifiers*.

The impact of a DRDoS attack is proportional to the adoption of the protocol that it is exploiting, as wide adoption makes it easier to find and scale-out the amplifier population. The two attacks mentioned above were particularly devastating because they exploited DNS and NTP, both of which are widely-used protocols in the Internet today.

In this paper, we show that BitTorrent, one of the most popular P2P file sharing protocols[2], can also be exploited to launch DRDoS attacks. BitTorrent and BTSync make use of UDP protocols. Since these protocols do not include mechanisms to prevent IP source address spoofing, an attacker can use peer-discovery techniques like trackers, DHT or Peer Exchange (PEX) [7] to collect millions of possible amplifiers.

We use the following three criteria to understand the impact of the vulnerabilities exposed in this work:

1. *Efficiency*: defined in terms of Bandwidth Amplification Factor (BAF) [39] and ease of amplifier identification;
2. *Robustness*: defined in terms of attack resilience under amplifier churn; and
3. *Evadability*: in terms of difficulty of attack circumvention (at amplifiers and victims) and ease of evasion.

We evaluate the detected vulnerabilities on the above criteria. Experiments are first performed on a custom testbed with 33 peers. We further substantiate our findings by crawling 2.1 million IP addresses over MLDHT and analyzing more than 10,000 BitTorrent handshakes.

Our experiments demonstrate that BitTorrent has a bandwidth amplification factor (BAF) of 50 times and

[1] According to the Spoofer project [9], more than 70 % of the public networks implement BCP 38 [18] to circumvent IP source address spoofing.

[2] According to some recent measurements, BitTorrent comprises 3.35 % of the worldwide bandwidth [35].

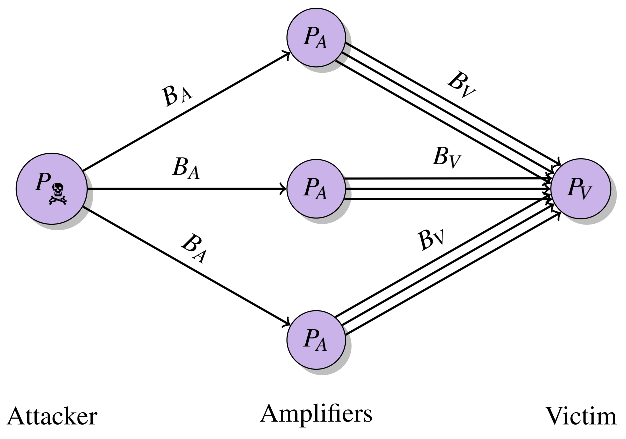

Figure 1: Schematic diagram of the threat model of a DRDoS attack.

in case of BTSync up to 120 times. Moreover, we observe that the most widely-used BitTorrent clients like uTorrent, Mainline and Vuze are also the most vulnerable ones. We also show that a possible attack is quite robust under amplifier churn as the BitTorrent protocol is widely-adopted and new amplifiers can be discovered quite quickly using standard peer discovery mechanisms. Finally, we show that due to its use of dynamic port ranges and encryption during handshake (in terms of MSE), a DRDoS attack that exploits BitTorrent cannot be detected using a standard firewall, and would instead require Deep Packet Inspection (DPI) to be detected.

2 Background

In this section, we provide a brief overview of DRDoS attacks and the BitTorrent protocol family. While discussing BitTorrent, we highlight the vulnerabilities that can be exploited.

2.1 Distributed Reflective Denial-of-Service (DRDoS) Attacks

An attacker which initiates a DRDoS does not send the traffic directly to the victim; instead he/she sends it to *amplifiers* which reflect the traffic to the victim. The attacker does this by exploiting network protocols which are vulnerable to IP spoofing. A DRDoS attack results in a distributed attack which can be initiated by one or multiple attacker nodes. Figure 1 outlines the threat model of a DRDoS attack.

The attacker $P_{☠}$ in Figure 1 needs to identify amplifiers before initiating the attack. This step is dependent on the protocol which the attacker wants to exploit. Internet-wide scanning tools like ZMap [17] can help to identify possible amplifiers. The speed and ease of identifying new amplifier is fundamentally important to all the criteria (attack efficiency, robustness under amplifier churn, and evadability at amplifiers and victims) that we used as a efficacy benchmark throughout this paper.

After the attacker has identified amplifiers, $P_{☠}$ initiates the attack by sending small packets B_A to the amplifiers P_A. Instead of using its own socket address, the attacker spoofs the address in the packet B_A from the victim P_V. The amplifiers respond to the victim P_V with a larger packet B_V. This type of attack has several advantages:

- the attacker hides his own identity, since the attacks uses IP spoofing (evadability advantage);
- it can be initiated by a single computer, but results in a distributed attack (efficiency advantage); and
- the amplifiers send a larger packet to the victim and therefore increase the impact of the attack (efficiency advantage).

The ratio of the smaller and larger packet is known as BAF [39]:

$$BAF = \frac{|B_v|}{|B_a|}, \tag{1}$$

where the payload to the victim is denoted as $|B_v|$ and the amplified payload from the victim as $|B_a|$. For instance, a BAF of 5 times means, that an attacker with 1 Gbps upload capacity can send 5 Gbps of traffic to the victim. Similar to BAF, a Packet Amplification Factor (PAF) is defined as the ratio of the number of packets sent from the amplifier to the victim and the number of packets sent from the attacker to the amplifier.

2.2 BitTorrent Protocol Family

In this section, we first introduce the BitTorrent terminology which we used throughout this paper. We then briefly discuss the different protocols.

Node: A physical or virtual machine with an IP stack.

Peer: A node that runs a BitTorrent client.

Swarm: All the peers sharing a torrent.

Torrent: A file that contains metadata about the BitTorrent swarm and the distributed files.

Info-hash: SHA-1 hash (160 bit) from the .torrent file which identifies a swarm.

Peer-id: Unique ID which identifies a single peer which is chosen at random.

Seeder: A peer that has downloaded the complete content and shares it with other peers.

Leecher: A peer which is downloading content of the torrent.

BitTorrent is the most commonly-used P2P protocol of the world today[3]. The novelty of this protocol is, that it provides solutions for the *free riding problem* [11] and the *last piece problem* [23]. To overcome the first problem, BitTorrent uses an incentive mechanism called the *choking algorithm* [13] which results in a tit-for-tat-ish way of sharing. BitTorrent solves the second problem by introducing the *rarest piece first* algorithm [28].

Similar to all P2P systems, BitTorrent also has to overcome the *bootstrapping problem*: how can a new peer join the network when there is no central contact point? The original BitTorrent specification introduces the concept of a *tracker*, which is a server that registers all participating peers. Before a newly-joined peer can participate, it requests the tracker for contact information of other peers.

Over time, a number of extensions to the BitTorrent protocol have been proposed to avoid a central (tracker) contract point; see, for instance, DHT [30], PEX [7] and Local Peer Discovery (LPD) [33]. If the new peer has a number of peers, it starts to connect to them and sends a special BitTorrent handshake.

Originally, TCP was the default protocol for handshake and all subsequent communications with peers. TCP, however, has some disadvantages when used in a P2P environment, as it distributes the available bandwidth evenly across all connections. Since a P2P application uses multiple connections, it always gets more bandwidth than other applications. As a consequence, BitTorrent, which is usually meant to run in the background, ends up interfering with the foreground traffic (like web surfing or emails). This is why BitTorrent invented a new transport protocol called uTP.

2.3 Micro Transport Protocol (uTP)

BitTorrent Inc. announced in December 2008 that uTorrent will switch its default transport protocol from TCP to uTP [21]. This protocol sits on top of UDP and implements a novel congestion control called Low Extra Delay Background Transport (LEDBAT) [24]. This congestion control detects foreground traffic by using one way delay measurements. If the difference between the measurements increases, the sender automatically throttles back.

[3] BitTorrent's bandwidth usage is geographically disparate [35]; e.g. in Europe, it comprises 31.8 % of all upstream traffic and 12.1 % of downstream traffic during peak hours [42].

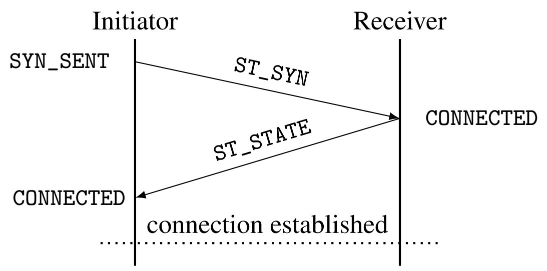

Figure 2: Two-way handshake to initiate a connection between two uTP nodes. The text on the outer edge reflects the state of the protocol.

uTP [32] adopts a few ideas from TCP. It controls the flow with a sliding window, verifies data integrity with sequence numbers and initiates a connection with a handshake. Unlike TCP, uTP uses a two-way handshake instead of a three-way handshake. Figure 2 depicts the message flow to establish a connection between two uTP nodes.

It can be seen that the initiator sends a `ST_SYN` packet to the receiver to initiate a connection. This is similar to the `SYN` packet in TCP. The receiver acknowledges the `ST_SYN` packet with a `ST_STATE` packet. After receiving that packet, the bidirectional connection is successfully established.

Most of the present BitTorrent clients now implement uTP as the default transport option. In the next section we will show how an attacker can exploit uTP to start an amplification attack.

3 Amplification Vulnerabilities in uTP

In this section, we reveal amplification vulnerabilities introduced by three popular BitTorrent protocols: BitTorrent [14], BTSync [6], and uTP [21].

3.1 Two-way handshake of uTP

As described briefly in Section 2.3, uTP establishes a connection with a two-way handshake. This allows an attacker to establish a connection with an amplifier using a spoofed IP address, as the receiver does not check whether the initiator has received the acknowledgment. We first tested this idea with the uTP test program `ucat` which is attached to `libutp`[4]. This program is the counterpart of `netcat` but uses uTP instead.

We setup a receiver that provides an arbitrary binary file when responding to `ST_DATA`. The attacker sends

[4] `https://github.com/bittorrent/libutp`

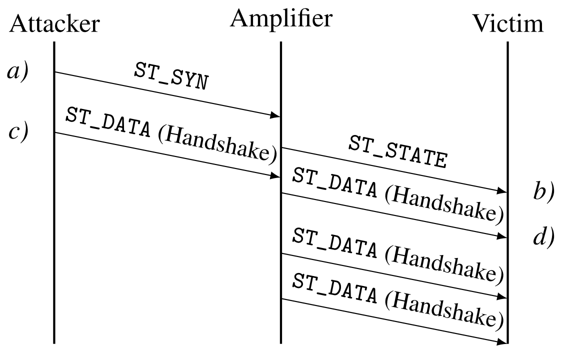

Figure 3: In uTP the initiator sends a ST_SYN packet to the receiver. The receiver acknowledgs this with a ST_STATE packet. After that, the connection is established (two-way handshake).

a forged ST_DATA to this receiver with the spoofed IP address of the victim. The receiver believes that the source IP address in that packet is valid and sends the first ST_DATA packet to the spoofed address. Since the forged address has not expected that packet, it does not send an acknowledgment back. The receiver runs into timeout and retransmits the lost ST_DATA packet. If 4 consecutive transmissions have timed out, uTP kills the connection. In this experiment, the receiver has sent 5 packets with a payload size of 1402 bytes, which results in 7030 sent bytes. According to Equation (1), this results in a BAF of 351.5 times, since the initiator only sends a ST_SYN packet with a size of 20 bytes.

uTP does not send more packets because it implements a slow-start mechanism. The max_window starts with 1382 bytes and increases for every acknowledgment it receives. When it is not receiving acknowledgments, it remains at that value. To the best of our knowledge, uTP is only used by BitTorrent so far. TOR project, however, thought about replacing TCP with uTP [29], but came to the conclusion that it is not trivial to replace it.

Let us now highlight how an attacker can exploit the two-way handshake of uTP with BitTorrent.

3.2 Exploiting BitTorrent Handshake via uTP

After the connection is established BitTorrent requires a handshake as its first message. It contains reserved bytes for extensions, info-hash and the peer-id. If a client receives a handshake with an info-hash that it does not participate, the client drops the connection immediately. An attacker can use the BitTorrent handshake to initiate an amplification attack based on the two-way handshake of uTP. Figure 3 outlines such an attack scenario.

The attacker initiates a connection with a spoofed ST_SYN packet to the amplifier in *a)*. The amplifier responds in *b)* with an acknowledgment via a ST_STATE packet to the victim. This packet does not contain useful information, because uTP only supports the two-way handshake. In step *c)*, the attacker sends an ST_DATA packet with a BitTorrent handshake in the payload.

This handshake needs an info-hash (20 bytes) of an active torrent of the amplifier. The handshake has a minimum size of 88 bytes. If the amplifier participates in that torrent, it will respond in *d)* with its own handshake. The handshake in *d)* is bigger than the packet which the attacker has sent, since the clients put additional messages in the uTP packet.

In our tests, nearly all clients either sent either a BITFIELD or multiple HAVE messages within the first uTP data packet. If and how many HAVE messages are sent with the handshake, depends on the client implementation. The size of the BITFIELD message depends on the size of the shared files. Since the handshake from *d)* does not get acknowledged, the amplifier thinks the packet is lost and retransmits the handshake again. In our tests, all clients retransmitted the handshake from 3–4 times until the connection gets terminated.

BitTorrent provides Libtorrent Extension Protocol (LTEP) to add new extensions without interfering with the default protocol. If an attacker signals that it supports the LTEP, that is specified in BEP 10 [34], it can further amplify the attack without increasing the size of the handshake in *c)*; i.e. efficiency of the attack is only enhanced as LTEP it is a simple bit that is set in the extension byte. With that extension, the peers exchange peer information with each other. All tested clients send these messages together with the handshake. This is because one uTP packet can contain multiple BitTorrent messages. We refer to this vulnerability as BitTorrent Handshake (BTH).

The BAF of BTH is as follows:

$$BAF_{BTH}(p,n) = \frac{20+20+p\times(n+1)}{20+88}, \qquad (2)$$

where p denotes the payload size, n denotes the number of retransmissions, and all numbers are in bytes. The number of retransmissions n gets incremented by one, since the first regular packet does not belong to the retransmissions. The 20 bytes on the denominator belongs to the ST_SYN packet and the 88 bytes belongs to the ST_DATA packet that contains the BitTorrent handshake. The 40 bytes on the numerator are the acknowledgements of two sent packets.

In the next subsections, we evaluate the BAF_{BTH} for the five most commonly-used BitTorrent clients [41], namely uTorrent (48 % market share), Vuze (22 %), BitTorrent mainline client (13 %), Transmission (7 %) and LibTorrent (1 %).

3.2.1 Mainline and uTorrent clients

We have tested uTorrent 3.4.2 (built 35702) and mainline BitTorrent 7.9.2 (built 35144) on Windows 7. Since mainline BitTorrent version 6.0 is uTorrent with a rebranded GUI, we handle both clients together. In our tests, both clients sent not only the BitTorrent handshake, but also one `BITFIELD`, multiple `HAVE` messages and one `PORT` message. The number of `HAVE` messages that the client sends depends on the protocol state of the client. If the client is in leech state, it sends the number of pieces that it has already downloaded, but not more than 24. In seed state, the client always sends 24 `HAVE` messages, thus resulting in a BAF of 27.5 times.

If the attacker sets the LTEP bit in the handshake and the amplifier also supports it, the amplifier sends an additional extension message handshake in the first packet. This handshake is a bencoded dictionary that contains meta information about the peer (e.g. version of client) and also a list of extensions that the peer supports. We observed that in both clients the LTEP handshake is larger compared to other clients. This is because they put more meta information in that packet. Both clients also support more extensions which are not public, like `ut_holepunch` and `ut_comment`. This additional handshake increases the BAF up to 39.6 times.

3.2.2 Vuze

Vuze is the most commonly-used BitTorrent client. We have tested Vuze 5.4.0.0/4 (formerly Azureus) on Windows 7. Without signaling any extensions, Vuze responds only with the BitTorrent handshake and a complete `BITFIELD` message. Vuze sends the uTP packet four times. This results in a BAF of 13.9 times. Vuze does not only support LTEP, but it has also designed its own extension protocol called Azureus Message Protocol (AMP) [3].

If an attacker signals that it supports LTEP and the amplifier also supports it, the amplifier adds the additional extension handshake. Compared with uTorrent, the handshake is smaller, because Vuze does not support the proprietary extensions from uTorrent. The BAF increases through LTEP up to 18.7 times. An attacker can increase the BAF if it signals that it supports the AMP.

Vuze uses AMP to transmit BitTorrent messages, Azureus messages and other communications like chat messages. If the AMP bit is set in the client handshake, a Vuze amplifier adds the following messages to the packet: `AZ_HANDSHAKE`, `AZ_PEER_EXCHANGE`, `AZ_REQUEST_HINT`, `AZ_STAT_REQ` and `AZ_HAVE`. This increases the handshake up to 1165 bytes, which increases the BAF to 54.3 times. This value can be even higher, since the `BITFIELD` message depends on the size of the shared files. In our analysis we used the Ubuntu 14.10 image that has a size of 1.2 GByte.

3.2.3 Transmission and LibTorrent

We tested with Transmission 2.84 (built 14307) on Ubuntu 14.04.1. Transmission supports both LTEP and AMP. However, Transmission does not add any other BitTorrent message in the first uTP data packet than the handshake. It does not matter which extension is activated, Transmission only sends 88 bytes in a BitTorrent handshake and resends a lost packet three times. According to this, an attacker can only achieve a BAF of 4.0 if the amplifier uses the Transmission client.

Libtorrent is a BitTorrent library written in C++ which is used by over 25 different BitTorrent clients. We have tested libtorrent 1.0.2 on Ubuntu 12.04. Like Transmission, libtorrent does not add any other BitTorrent message to the first uTP data packet, except the handshake. Where libtorrent is different, compared to Transmission is the number of retransmissions. Libtorrent resends a lost packet six times, which increases the BAF up to 5.2 times. If a peer wants to obfuscate its BitTorrent traffic, it starts with MSE handshake instead of a BitTorrent handshake.

3.3 Exploiting Message Stream Encryption (MSE) Handshake

The aim of MSE [8] is to obfuscate BitTorrent traffic to avoid shaping, rather than to encrypt the traffic securely. In addition to that MSE encrypts the traffic and provides confidentiality. The first objective of MSE, however, was to obfuscate the traffic to avoid traffic shaping by ISPs. Brumley and Valkonen showed in their paper [12] that MSE has a number of serious weaknesses. It is implemented by most of the BitTorrent clients like uTorrent, BitTorrent mainline, Vuze, Transmission, libtorrent, BitComet, etc.

The protocol starts with a Diffie-Hellman key exchange (DH), where each peer generates a 768 bit public key. To avoid having fixed length packets, each peer generates random data r with a length of 0–512 bytes and adds it to the public key. After the key exchange, the packets are RC4 encrypted. The transport protocol of these messages are based on uTP. One advantage of this method is that an attacker does not have to know a valid info-hash from the amplifier.

An attacker can send a spoofed MSE handshake that includes 768 bit public key without random data. A client which supports MSE responds with its public key and random data. Hence, the BAF for a client with MSE is.

$$BAF_{MSE}(r,n) = \frac{(116+r)\times(n+1)}{116}, \quad (3)$$

where r is the length of the random data with interval 0 bytes $\leq r \leq 512$ bytes and n the number of retransmissions $n = \{4,5,6\}$. According to Equation (3), BAF_{MSE} ranges from 4–32.5 times.

The results of this section demonstrates that an attacker who exploits MSE can significantly enhance the efficiency of the BTH attack. Furthermore, note that the encrypted payload of B_A and B_V packets generated using MSE have a high entropy and are therefore hard to detect and to block with Stateful Packet Inspection (SPI) or DPI firewalls. Statistical measurements, however, show good results to detect MSE [22, 25], but are not widespread used. Hence, use of MSE also helps in making the attack difficult to detect and circumvent, hence contributing to the evadability of the attack. To avoid a central *tracker*, BitTorrent included the DHT protocol to the BitTorrent protocol family.

3.4 Exploiting DHT Messages

The DHT implementation in BitTorrent clients is divided into two protocols: MLDHT [30] and Vuze DHT (VDHT) [5]. The MLDHT is by far the biggest overlay network with users of around 15–27 million [43] per day. Both protocols are not compatible with each other. DHT is used to find peers that share the same torrent. The MLDHT implementation is discussed in the next subsection.

3.4.1 Mainline DHT

The MLDHT supports the following queries: `ping`, `find_node`, `get_peers` and `announce_peer`. The `ping` query is the most basic one and tries to find out if a peer is available via DHT. The request contains the command `ping` and peer-id which consists of a 20 byte string. Together with the RPC protocol, a `ping` request has a size of 56 bytes. The response only contains the peer-id of the responding peer, therefore the response packet has a size of 47 bytes. The `ping` query does not amplify the bandwidth.

The `find_node` query tries to find the k closest peers for a specific peer-id. The request contains both the peer-id of the requesting and sought peer. Together with the RPC overhead, the payload of a request packet has a size of 95 bytes. A peer responds with a list of k peers, whereby BitTorrent Enhancement Proposal (BEP) 05 [30] sets $k = 8$. According to this, the response varies from 283–332 bytes, depending on the implementation. In our tests we found two DHT peers[5] which set

[5] `router.utorrent.com` and `router.bittorrent.com`

Table 1: Amplification factors of the different MLDHT queries.

Description	BAF
`ping`	0.8
`find_node` with $K = 8$	3.1
`get_peers` with 100 peers (IPv4)	11.9
`get_peers` with 100 peers (IPv6)	24.5
`get_peers` with scrapes	13.4

$k = 16$. These peers are the bootstrapping peers for most BitTorrent clients. This results in a BAF of 5.2 times.

`get_peer` query is much more interesting, as it returns a list of peers for a given info-hash. If the peer does not have peers for that info-hash, it will return a list of k peers, similar to the `find_node` query. This means, only peers which are involved in a swarm are able to send a list of participating peers back. BEP 05 does not say anything about a limit for the peers. In our tests, we found peers which send a list of 100 peers back which results in a BAF of 11.9 times. If a peer would only return IPv6 addresses, this would increase the BAF up to 24.5 times. With an additional extension, we can increase the payload of the response even further.

BEP 33 [4] describes an extension to DHT called *DHT scrapes*. Scrapes are statistics of a swarm like the number of seeders, leechers and complete downloads. It is a method to assess the state of a swarm which is based on bloom filters. To request scrapes a peer has to send a `get_peer` request that contains the dictionary entry *scrape*. If the responding peer has database entries for a particular info-hash, it returns the statistics in the `get_peer` response. With that request, we were able to increase this value up to 13.4 times. The BAF of all queries from the MLDHT is summarized in Table 1.

3.4.2 Vuze DHT

The Vuze `ping` query without any special flags is similar to the results from the mainline DHT. The value of BAF is 0.8 times. Vuze, however, supports the Internet coordinate system *Vivaldi* [15, 40] which aims to estimate the round trip time of other peers without testing it. If the protocol version is $>=$ 10, then the amplifier adds Vivaldi network coordinates to the `ping` packet. This increases the `ping` reply packet up to an BAF of 14.9 times.

3.5 Exploiting BitTorrent Sync

BTSync is a proprietary protocol that makes use of uTP to synchronize files in a P2P way. This protocol currently has 1 million users [38] which makes it an interesting target for amplification attacks. We investigated three BTSync messages which can be exploited: tracker request, BTSync handshake and a ping message. For all messages, an attacker needs a valid BTSync secret. There are, however, a number of public websites like [1, 2] where BTSync users publish their secrets. An attacker can use these secrets to run an amplification attack. The first message is the tracker request.

By default, BTSync will contact a tracker to request peers for a specific secret. The tracker is run by BitTorrent Inc. and uses the domain `t.usyncapp.com`. A DNS request resolves this domain to four IP addresses which are hosted by Amazon's EC2 cloud. The tracker request starts with a uTP `ST_SYN` packet followed by a `ST_DATA` packet that contains a bencoded payload which includes the peer-id, secret, local address and local port. The tracker responses with a list of peers. The response is bigger than the request. It depends, however, highly on the number of peers the trackers sends. The next message is the BTSync handshake.

The BTSync handshake is similar compared to BitTorrent. The `ST_DATA` packet contains a bencoded packet with the secret and a 16 bytes `nonce` value which can be random. The peer responses with a 160 bytes public key and a 16 bytes salt value. This yields to a BAF of 10.8 times.

The ping is a normal UDP packet that starts with the string `BSYNC` followed by a zero byte and a bencoded dictionary. The dictionary contains the command `ping`, 20 bytes long secret and 20 bytes long peer-id. The UDP payload of this packet is altogether with the overhead of the dictionary 76 bytes. The other side also responds with a ping message. But not just one — it sends 117 ping messages and 12 uTP `ST_SYN` packets to the requester. This yields to a BAF of `120.2` and a PAF of 129. This happens with BTSync versions 1.4 and 2.0.105.

3.6 Discussion

As a summary, we have listed all the BAF values in Table 2. Our protocol analysis shows that BitTorrent is highly vulnerable to DRDoS attacks. An attacker is able to amplify the traffic beginning from 4–54.3 times. If an attacker knows the peer-id from the amplifier, he/she is able to predict the BitTorrent client and start a target-oriented attack. This is possible through the peer-id convention which is defined in BEP 20 [20]. Even if a client does not support uTP, an attacker can still exploit DHT or MSE with a lower BAF value. This means that nearly every client can be exploited, which makes such an attack very efficient and robust against peer churn. The next section will focus on the experimental evaluation of these exploits.

Table 2: Amplification Factors of the different BitTorrent clients with a BitTorrent handshake with uTP.

Description	BAF	PAF
ucat	351.5	6
uTorrent w/o extensions	27.6	3.5
Mainline w/o extensions	27.8	3.5
uTorrent with LTEP	39.6	3
Mainline with LTEP	39.6	3
Vuze w/o extensions	13.9	2
Vuze with LTEP	18.7	2
Vuze with AMP	54.3	3.5
Transmission w/o extensions	4.0	3.5
Transmission with LTEP	4.0	3.5
Transmission with AMP	4.0	3.5
Libtorrent w/o extensions	5.2	4
Libtorrent with LTEP	5.2	4

4 Experimental Evaluation

In this section, we show that the attacks presented in this paper are efficient, robust and difficult to circumvent.

4.1 Efficiency

We perform experiments in our testbed system consisting of 33 nodes: 1 attacker, 1 victim and 31 amplifiers. All nodes are desktop machines which are running Ubuntu 12.04.02 LTS. We installed uTorrent server V3.3 on all amplifiers with a private torrent of 1 GByte in seed mode.

The attacker in our experiment runs a scapy[6] script that sends forged uTP packets to the amplifier. It first sends 31 uTP `ST_SYN` packets then it waits for 1 second and sends 31 uTP `ST_DATA` packets which contain a BitTorrent handshake. The script waits then for 120 seconds, since otherwise the amplifiers would just terminate the connection with a `ST_FIN` packet. The attacker script does not need to save the state for each peer, which makes it quite efficient. An I/O graph can be seen in Figure 4.

It can be seen that after the attacker sends the forged packet, the amplifiers sends the traffic to the victim. The first peak is the highest, since both, the acknowledgement for the first SYN and the handshake arrive together at the victim. The following peaks are the retransmissions of the handshake and acknowledgement. During our experiments we noticed that uTorrent for Linux behaves differently compared to the Windows client. The attack had an amplification factor of 14.6 times.

[6]`http://www.secdev.org/projects/scapy/1`

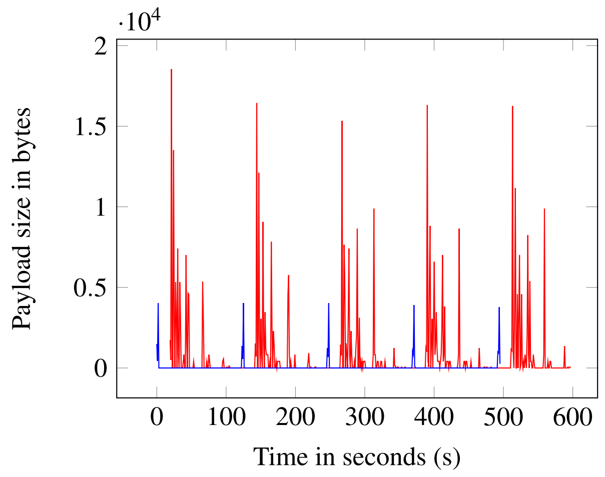

Figure 4: Amplification attack with 1 attacker, 31 amplifiers and 1 victim. The blue line shows the payload size that the attacker sends to the amplifier. The red line shows the payload size that the amplifier sends to the victim.

4.2 Robustness

To evaluate the robustness of our detected vulnerabilities against amplifier churn in a real-world attack, we wrote two BitTorrent crawlers. The first one searched the MLDHT network and the second one used the results from the MLDHT network to start a BitTorrent handshake via uTP. We tracked 9.6 million possible amplifiers where 2.1 million peers were responding to our DHT requests. First we describe our crawler architecture and then we present our results for the DHT network and the handshake requests.

4.2.1 Architecture

We programmed our crawler in the programming language Elixir, which is a functional and concurrent language built on top of the Erlang Virtual Machine (EVM). Our crawler is open source and can be found on GitHub[7].

At first we filled a database table “torrents” with the complete magnet database from Piratebay from 13th February 2012. This database comprised 1.6 million info-hashes. Each DHT harvester process takes an info-hash and sends a `find_node` request to a bootstrapping node. This node returns 20 peers, which the harvester saves in our database. At that time, the requester takes peers from our databases and sends `find_node` requests to it. Additionally, we save meta informations from the responses e.g. payload size, version of the BitTorrent client, number of nodes and so on. The harvester stops when it has requested 1000 peers.

[7]`https://github.com/cit/bt_crawler`

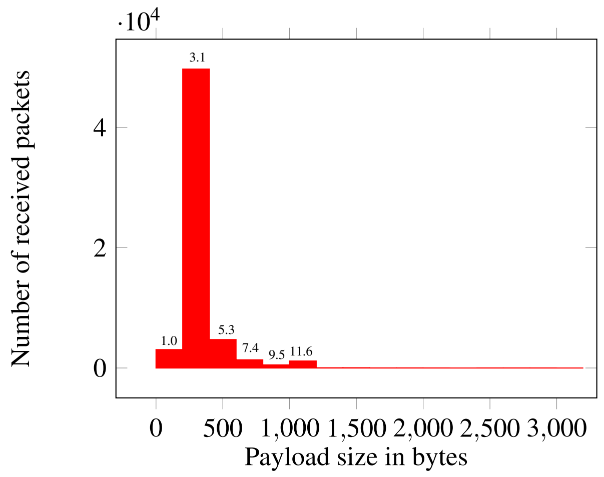

Figure 5: Histogram of the payload size from the DHT responses which are caused by `get_peers` requests. The numbers on top of the bars are the average BAF values.

The peer requester uses the results from the DHT harvester and sends a `ST_SYN` packet to that peer. If the peer sends us a `ST_STATE` packet, we respond with a BitTorrent handshake that has all extensions enabled. Then this process waits for a reply and saves all responses to our database.

4.2.2 Mainline DHT Network

We collected overall 9.6 million peers via MLDHT beginning from 1st January 2015 until 1st February 2015. From those peers, 2.1 million responded to our `get_peers` request, which corresponds to 21.9 %. From these responding peers 67.8 thousand peers included participating peers, which corresponds to 3.2 %. The rest were only returning *k* neighbors. Figure 5 shows a distribution of the payload size of all responses.g

The mean of all received packets is 665.6 bytes which results in a BAF of 7 times. The max value is 3344 bytes with a BAF of 35.2 times and the min value is 82 bytes results in a BAF of 0.9 times. From the above data it is clear that MLDHT can be exploited, but most of the responses do not produce a high BAF value. Therefore, we used the gathered peer information from the MLDHT network to request a BitTorrent handshake to test the efficacy.

4.2.3 uTP distribution

In this section we analyze the responses from the uTP requests. For every peer that participates in a torrent, we send a uTP `ST_SYN` packet. If we get a `ST_STATE` between 10 seconds we send a BitTorrent handshake to that peer. Overall, we collected 10,417 handshakes via

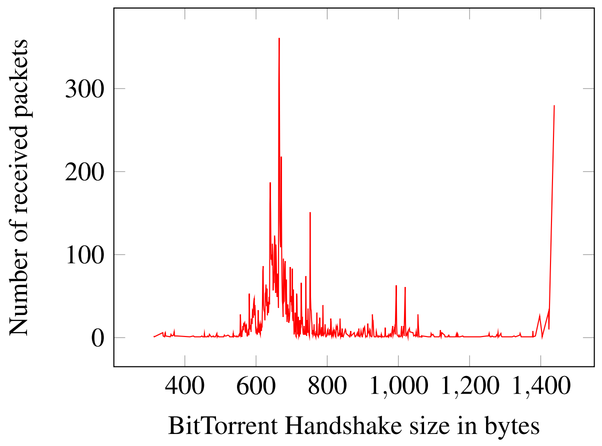

Figure 6: Histogram of the BitTorrent handshake size from the uTP responses.

uTP. The first question that we answer is the following: What is the distribution of the payload size of the first data packet? Since this is crucial for the impact an amplification attack which can be seen in Equation 2. Figure 6 shows a histogram of the payload size of all received BitTorrent handshakes.

The histogram shows clearly that there are three peaks in the distribution of the payload size. The first peak is the highest value of 665 bytes. This peak comprises 15.7 % of all handshakes and would result with a 5 times retransmission of BAF of 31.2 times. The next peak is at around 1000 bytes with a frequency of 2.2 %. This would result a BAF of 46.7 times. The last peak is the highest payload size of 1438 bytes. It comprises 3.9 % of all handshakes and result in a BAF of 66.9 times. The mean of all received packets is 810.8 bytes, the max value is 1438 bytes and the min value is 313 bytes. We calculated BAF values above with a number of retransmissions of 4 packets (total of 5 packets), since this is the protocol default.

We analyzed all LTEP message that were in the first uTP data packet. A LTEP message contains a dictionary entry with the exact client version. Table 3 shows the evaluation of these messages.

It can be seen that the latest uTorrent and BitTorrent version tops the list in Table 3. Only 0.1 % were unknown, since they were not transmitting an LTEP message. Since we only crawled the MLDHT network, Vuze is not in our list. In the next subsection, we will discuss how difficult it would be to circumvent an DRDoS attack that exploits BitTorrent.

4.3 Evadability

DNS and NTP use well-known ports for their service. As a consequence, the reflected traffic can easily be blocked

Table 3: Client software and version number of the inspected LTEP messages.

Version	Count	Percentage
uTorrent 3.4.2	8616	82.7
BitTorrent 7.9.2	1641	15.8
uTorrent 3.4.1	107	1.0
uTorrent 3.4	27	0.3
BitTorrent 7.9.1	14	0.1
Unknown	9	0.1
BitTorrent 7.9	2	0.0

Table 4: Comparison of amplification vulnerabilities and with which firewall technology it can be defended

	DNS'13	NTP'14	BTH	MLDHT	VDHT	BTSync	MSE
SPI firewall	X	X					
DPI firewall			X	X	X	X	

by a SPI firewall. This is not possible with BitTorrent, since it uses dynamic ports. The payload of the BitTorrent handshake contains the character 19 followed by the string “BitTorrent protocol”. In case of BTSync the handshake contains the string “BTSYNC”. In both cases, the victim needs DPI hardware to block an attack from these protocols.

The situation looks different when an attacker exploits MSE. The payload of a MSE handshake is completely random. Table 4 shows comparison of the different protocols and with which firewall technology it can be defended. In the next section, we will discuss the countermeasures of the proposed attacks.

5 Countermeasures

In this section we will discuss countermeasures against the presented vulnerabilities. The root cause of these attacks is the IP source address spoofing. It has been seen that the anti-spoofing filtering techniques like Ingress address filtering and unicast reverse path are effective against IP spoofing. In practice these techniques have limits, especially in an ISP infrastructure. The Spoofer project measures the susceptibility of IP spoofing around the world since 2005 until now [9]. According to their latest measurements from 2015, 26.1 % of all autonomous systems allow spoofing and 15.5 % allowed partial spoofing. We think a working countermeasure must follow two parallel ways: global ISP coordination to prevent IP spoofing and protocol defense mechanism to avoid protocol exploitation.

5.1 Handshakes over uTP

The presented attacks are only possible, since uTP makes use of the two-way handshake. Therefor it is possible for an attacker to send a spoofed BitTorrent or MSE handshake via uTP to an amplifier. We highly recommend that uTP switches to the three-way handshake, like TCP. It would prevent the presented attacks, since both nodes do not send data until the last acknowledgment arrives, like some TCP devices do [27]. The disadvantage of this approach is that it is a significant change in the protocol. An alternative that does not prevent the attack, but reduce the impact, is to limit the messages in the first uTP packet to one, like the clients in Section 3.2.3. This results in a maximum BAF of 4–5 times.

5.2 DHT

UDP tracker, MLDHT an VDHT already have a countermeasure against index poisoning attacks [31]. It prevents an attacker to add other peers except from themself to the DHT network with the `announce_peer` or `STORE` call. It does this, by requiring a token that a peer has to request from a previous `get_peers` or `FIND_NODE` query. This token is valid for 10 minutes and prevents spoofing attacks but only for the `announce_peer` or `STORE` query. To prevent amplification attacks it would be necessary to request the token in the `ping` query, since it is the one with the smallest BAF value. All other queries need to require this token. This would prevent any spoofing attacks against DHT. The disadvantage of this mechanism is that it always requires two DHT queries to request new peers which slows down the bootstrapping time.

6 Related Work

In this Section we will discuss related work that has influenced and inspired our research.

6.1 Amplification Attacks

Rossow [39] did the first broad investigation of UDP based protocols. They found 14 protocols that are vulnerable, including MLDHT, for which we reproduced the same results. We, however, did a more thorough investigation of the BitTorrent family, which includes uTP, MSE, BTSync, MLDHT and Vuze DHT with different queries and different clients.

Kührer et al. [26] analyzed the amplifier magnitude for DNS, SNMP, Simple Service Discovery Protocol (SSDP), Character Generator Protocol (CharGen), Quote of the Day (QOTD), NTP and NetBIOS with an Internet-wide scan. Additionally, they developed a remote spoofer test to check if a network allows IP spoofing. They found more than 2,000 networks which were lacking of egress filtering. In a follow-up work, Kührer et al. [27] investigated how vulnerable TCP is. Despite its three-way handshake, there are TCP/IP implementations that do not strictly follow the standard. This means, the implementation sends data before the three-way handshake is complete.

6.2 P2P DDoS attacks

A number of papers focused on BitTorrent tracker as their target to run DDoS attacks. Harrington [19] exploited the fact that peers trust the tracker. They modified a tracker which instead of distributing a list of participating peers, it distributes a list of victims. Every peer that requests that tracker tries to connect to the victims. An attacker, however, can inject the victim as a tracker itself, like presented by Defrawy [16]. Hereby, the victims get all the tracker requests from peers which results in a DDoS. We focused instead how vulnerable the BitTorrent protocol family is to IP source address spoofing.

In a previous research work [10], we did a security analysis on uTP. LEDBAT, the congestion control in uTP, only works with correct feedback from the receiver. A misbehaving receiver which is not interested in data integrity can increase the bandwidth consumption up to five times. This can cause congestion collapse which results in a DOS attack.

7 Conclusion

BitTorrent and BTSync are vulnerable to DRDoS attacks. We demonstrated that these attacks are *efficient*. With peer-discovery techniques like trackers, DHT or PEX, an attacker can collect millions of amplifiers. An attacker only needs a valid info-hash or secret to exploit the vulnerabilities. In that case, we have shown that the most used BitTorrent clients, uTorrent, Mainline and Vuze, are highly vulnerable and can be amplified up to a factor of 50 times. With a single BTSync ping message, an attacker can amplify the traffic up to 120 times. An easier amplifier target is a MSE handshake, since an attacker does not need an info-hash. The amplification factor of this attack ranges from 4–32.5 times. We showed that a possible attack is *robust* against amplifier churn. To validate the robustness of the attack, we wrote a BitTorrent crawler which ran for one month and collected more than 2.1 million IP addresses and analyzed more than 10,000 BitTorrent handshakes. We showed that an attack is quite *difficult to circumvent*, as the found vulnerabilities can only be defended with a DPI firewall. In case of a MSE handshake, it is even harder to detect the attack, since the packet contains a high entropy payload with a public key and random data.

References

[1] Public Directory of BitTorrent Sync Keys. http://btsynckeys.com/.

[2] Reddit: BitTorrent Sync Secrets. http://www.reddit.com/r/btsecrets/.

[3] Azureus messaging protocol - VuzeWiki. https://wiki.vuze.com/w/Azureus_messaging_protocol, Oct. 2010.

[4] BEP 33: DHT Scrapes. Tech. rep., BitTorrent Inc., Jan. 2010.

[5] Distributed hash table. https://wiki.vuze.com/w/Distributed_hash_table, Oct. 2012.

[6] BitTorrent Sync. https://www.getsync.com/, Apr. 2013.

[7] BitTorrentPeerExchangeConventions - Theory.org Wiki. https://wiki.theory.org/BitTorrentPeerExchangeConventions, Apr. 2014.

[8] Message Stream Encryption Format Specification Version 1.0. http://wiki.vuze.com/w/Message_Stream_Encryption, May 2014.

[9] Spoofer Project: State of IP Spoofing. http://spoofer.cmand.org/summary.php, Jan. 2015.

[10] ADAMSKY, F., KHAYAM, S. A., JÄGER, R., AND RAJARAJAN, M. Security Analysis of the Micro Transport Protocol with a Misbehaving Receiver. In *Proceedings of the 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (Oct. 2012), IEEE, pp. 143–150.

[11] ADAR, E., AND HUBERMAN, B. A. Free Riding on Gnutella. *First Monday* (2000).

[12] BRUMLEY, B. B., AND VALKONEN, J. Attacks on Message Stream Encryption. In *Proceedings of the 13th Nordic Workshop on Secure IT Systems* (Copenhagen, Denmark, Oct. 2008), pp. 163–173.

[13] COHEN, B. Incentives build robustness in BitTorrent. In *Proceedings of 1st Workshop on Economics of Peer-to-Peer Systems* (2003), pp. 1–5.

[14] COHEN, B. BEP 03: The BitTorrent Protocol Specification. Tech. rep., BitTorrent Inc., Nov. 2013.

[15] DABEK, F., COX, R., KAASHOEK, F., AND MORRIS, R. Vivaldi: a decentralized network coordinate system. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (2004), ACM Press, pp. 15–26.

[16] DEFRAWY, K. E., GJOKA, M., AND MARKOPOULOU, A. BotTorrent: misusing BitTorrent to launch DDoS attacks. In *Proceedings of the 3rd USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet* (June 2007), USENIX Association, pp. 1–6.

[17] DURUMERIC, Z., WUSTROW, E., AND HALDERMAN, A. J. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)* (Washington, Aug. 2013), pp. 605–620.

[18] ERGUSON, P., AND SENIE, D. BCP 38: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. Tech. rep., Internet Engineering Task Force (IETF), May 2000.

[19] HARRINGTON, J., KUWANOE, C., AND ZOU, C. C. A BitTorrent-driven distributed. In *Proceedings of the 3th International Conference on Security and Privacy in Communications Networks* (2007), IEEE, pp. 261–268.

[20] HARRISON, D. BEP 20: Peer ID Conventions. Tech. rep., BitTorrent Inc., Feb. 2008.

[21] HAZEL, G. Announcements: uTorrent 1.9 alpha 15380. https://goo.gl/4ThWLY, Dec. 2008.

[22] HJELMVIK, E., AND JOHN, W. Statistical protocol identification with SPID: Preliminary results. In *Proceedings of the Swedish National Computer Networking Workshop* (2009).

[23] KOSTÍC, D., BRAUD, R., KILLIAN, C., VANDEKIEFT, E., ANDERSON, J. W., SNOEREN, A. C., AND VAHDAT, A. Maintaining high bandwidth under dynamic network conditions. In *Proceedings of the annual conference on USENIX Annual Technical Conference* (2005), pp. 193–208.

[24] KUEHLEWIND, M., HAZEL, G., SHALUNOV, S., AND IYENGAR, J. RFC 6817: Low Extra Delay Background Transport (LEDBAT). http://tools.ietf.org/html/rfc6817, Dec. 2012.

[25] KÖHNEN, C., ÜBERALL, C., ADAMSKY, F., RAKOCEVIC, V., RAJARAJAN, M., AND JÄGER, R. Enhancements to Statistical Protocol IDentification (SPID) for Self-Organised QoS in LANs. In *Proceedings of the 19th International Conference on Computer Communications and Networks (ICCCN)* (Aug. 2010), IEEE, pp. 1–6.

[26] KÜHRER, M., HUPPERICH, T., ROSSOW, C., AND HOLZ, T. Exit from Hell? Reducing the Impact of Amplification DDoS Attacks. In *23rd USENIX Security Symposium (USENIX Security 14)* (San Diego, CA, Aug. 2014).

[27] KÜHRER, M., HUPPERICH, T., ROSSOW, C., AND HOLZ, T. Hell of a Handshake: Abusing TCP for Reflective Amplification DDoS Attacks. In *Proceedings of the 8th USENIX Workshop on Offensive Technologies* (San Diego, CA, Aug. 2014), USENIX Association.

[28] LEGOUT, A., URVOY-KELLER, G., AND MICHIARDI, P. Rarest first and choke algorithms are enough. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement* (2006), ACM Press, pp. 203–216.

[29] LOESING, K., MURDOCH, S. J., AND JANSEN, R. Evaluation of a libutp-based Tor Datagram Implementation. Tech. rep., Oct. 2013.

[30] LOEWENSTERN, A., AND NORBERG, A. BEP 05: DHT Protocol. Tech. rep., BitTorrent Inc., Mar. 2013.

[31] NAOUMOV, N., AND ROSS, K. Exploiting P2p systems for DDoS attacks. In *Proceedings of the International Workshop on Peer-to-Peer Information Management* (2006), ACM Press, pp. 47–53.

[32] NORBERG, A. BEP 29: uTorrent transport protocol. Tech. rep., BitTorrent Inc., June 2009.

[33] NORBERG, A. uTorrent Forum: Local Peer Discovery Documentation. http://goo.gl/jAgTlC, 2009.

[34] NORBERG, A., STRIGEUS, L., AND HAZEL, G. BEP 10: Extension Protocol. Tech. rep., BitTorrent Inc., Feb. 2008.

[35] PALOALTO NETWORKS. Application Usage Threat Report. http://researchcenter.paloaltonetworks.com/app-usage-risk-report-visualization/#sthash.Lme8Q76M.dpbs, 2013.

[36] PRINCE, M. Deep Inside a DNS Amplification DDoS Attack. http://blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack/, Oct. 2012.

[37] PRINCE, M. Technical Details Behind a 400gbps NTP Amplification DDoS Attack. https://goo.gl/l1vPTN, Feb. 2014.

[38] PROTALINSKI, E. BitTorrent Sync Passes 1 Million Users, Gets iPad Support and API. http://goo.gl/p46p9y, May 2013.

[39] ROSSOW, C. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 14)* (2014), Internet Society, pp. 1–5.

[40] STEINER, M., AND BIERSACK, E. W. Where Is My Peer? Evaluation of the Vivaldi Network Coordinate System in Azureus. *NETWORKING 2009* (2009).

[41] VAN DER SAR, E. uTorrent Keeps BitTorrent Lead, BitComet Fades Away. `https://goo.gl/EImuS`, Sept. 2011.

[42] VAN DER SAR, E. BitTorrent Traffic Increases 40 `https://goo.gl/d4cGA`, July 2012.

[43] WANG, L., AND KANGASHARJU, J. Measuring large-scale distributed systems: case of BitTorrent Mainline DHT. In *Proceedings of the 13th IEEE International Conference on Peer-to-Peer Computing* (Sept. 2013), IEEE, pp. 1–10.