BitTorrent's Mainline DHT Security Assessment

Juan Pablo Timpanaro, Thibault Cholez, Isabelle Chrisment*, Olivier Festor
INRIA Nancy-Grand Est, France
*LORIA - ESIAL, Henri Poincaré University, Nancy 1, France
Email: {juanpablo.timpanaro, thibault.cholez,olivier.festor}@inria.fr
Email: {isabelle.chrisment}@loria.fr

***Abstract*—BitTorrent is a widely deployed P2P file sharing protocol, extensively used to distribute digital content and software updates, among others. Recent actions against torrent and tracker repositories have fostered the move towards a fully distributed solution based on a distributed hash table to support both torrent search and tracker implementation. In this paper we present a security study of the main decentralized tracker in BitTorrent, commonly known as the Mainline DHT. We show that the lack of security in Mainline DHT allows very efficient attacks that can easily impact the operation of the whole network. We also provide a peer-ID distribution analysis of the network, so as to adapt previous protection schemes to the Mainline DHT. The mechanisms are assessed through large scale experiments on the real DHT-based BitTorrent tracker.**

***Index Terms*—BitTorrent, Distributed Tracker, Mainline DHT, Security Assessment, Protection Mechanisms.**

I. Introduction

BitTorrent [3] is a peer-to-peer protocol developed by Bram Cohen. A recent study [5] shows that between 43% and 70% of all Internet traffic is generated from BitTorrent clients, which makes it the most popular peer-to-peer protocol. However, there have been several legal issues and complaints from music and movie companies, putting in jeopardy the continuity of its success. Moreover, in some countries, there have already been legal actions to successfully shut-down major trackers site, such as The Pirate Bay or Mininova. Without a central server to retrieve the peers participating in the download of a given torrent, there is no possibility to join the network. However, an alternative tracking approach has already been implemented, the decentralized tracking. Every peer in the network acts as a small tracker, allowing a fully-decentralized architecture, in which there is no central component to attack. Notwithstanding, decentralized tracking in BitTorrent has major security problems. In BitTorrent, there are two implementations of decentralized tracking, both based on the Kademlia DHT [10]. The Azureus DHT came first, and it is only used for this client. In second place, the Mainline client introduced its DHT after around a month later and it was adopted by several clients.

Being today the largest decentralized tracker in BitTorrent, we restrict the focus in this study to the Mainline DHT.

In this work, we make the following contributions as a way to protect the alternative tracking in BitTorrent, the Mainline decentralized tracking network:

- We show the major security problems in the Mainline DHT Network.
- We propose a distributed architecture used to launch a series of evaluation on the real network.
- We adapt and analyse a set of protections mechanisms proposed for the KAD DHT network [2], in order to fulfill the security problems.
- Along with the work done by Wolchok et al. [15], we complete the security view of BitTorrent's decentralized trackers.

Even though these evaluations could have been done from a single computer, very basic rules to limit the number of peers per IP could easily mitigate a one-computer attack. Therefore, we chose a distributed approach.

The document is organized as follows. Section II presents a set of works in the area, regarding the BitTorrent protocol and a set of attacks, including monitoring. Section III details the distributed architecture proposed as well as the targeting component of BitTorrent, its decentralized tracker. Section IV introduces how we exploit some vulnerabilities in this alternative tracker and section V presents some mechanisms to avoid these attacks. Section VI concludes the paper.

II. Related works

Regarding decentralized tracking, Crosby et al. [4] present a complete study about the two decentralized trackers in BitTorrent. They examine a variety of aspects, such as latency, and detect problems mainly in the routing algorithms, and proposed a better maintenance of the routing table to avoid dead nodes. However they do not address any security problems.

Monitoring the BitTorrent network has been investigated in several ways. [9] proposes a simple, but yet effective way of spying BitTorrent users, through exploiting the tracker's infrastructure. Piatek et al. [11] show how exploiting this infrastructure properly, can lead to implicate arbitrary network endpoints in illegal content sharing. Saganos et al. [12] analyse a set of top torrents in order to blacklist BitTorrent's monitors.

Both BitTorrent components, the tracker and the swarm itself, have been the core study in many research works. This is not the case for the decentralized tracking. Jetter et al. [6] propose a self-registration mechanism, as a way to avoid a Sybil attack in the BitTorrent DHT. They limit the number of peers per IP, so as to avoid an attacker to launch several peers from a single machine. However, their solution does not maintain backward compatibility, and using a distributed architecture will bypass this protection. On the other hand, Wolchok et al. [15] conduct a monitoring study

on the Azureus DHT. They clearly show how the Azureus DHT can be crawled thanks to a Sybil attack, so as to rebuild from scratch a BitTorrent search engine as well as to monitor pirate's behaviour. To our knowledge, we present the first security study in the Mainline DHT.

This paper completes the work of Wolchok et al. by analysing Mainline DHT and its security characteristics. We extend our previous analysis in [2] to the Mainline DHT network in order to determinate if the security mechanisms can be applied.

III. BitTorrent Architecture

A. Overview of BitTorrent Architecture

Considering BitTorrent and its architecture, the following components can be described:

- Tracker: Entity responsible for helping peers to find each other by using a central tracker or a DHT service.
- Peers: Depending on if they have the entire file or only a part of it, can be respectively called "Seeders" or "Leechers".
- Swarm: The group of peers sharing the file. It is composed of Seeders and Leechers.
- Torrent File: Contains Metadata describing the file to share.

The torrent file contains mainly two parts. The first one is normally a list of trackers, which will have indexed the torrent file. The second part, the info part, describes the file to be shared and it contains a list of parts composing the file, along with a piece-hash for later verification.

There are two steps when trying to download a file.

In the first, a user wants to download a given file, which has a torrent file associated. Once this user retrieves the torrent file, normally from a website that distributes them, it loads it into a BitTorrent client. Secondly, the BitTorrent client will contact the trackers or the distributed tracker, and retrieve a list of peers already sharing the file. Finally, the client will start contacting every peer to join the swarm. Contacting the tracker will make the peer to be added to the list of peers sharing the torrent.

The second step is slightly more complex. Every peer in the swarm can download from whoever it can. However, in order to decide which peers to upload, a rewarding mechanism is used. This mechanism is based on a Tit-for-Tat scheme [3], and it allows fair trading inside the swarm. For example, peer A will select peer B to upload data, if peer B has shared with peer A. This selection is made locally by a peer and it is based on the upload rate of every peer to the local peer.

B. BitTorrent DHT's

The BitTorrent protocol has acquired some new features over time. Many of them are still under consideration before being added officially to the protocol. Among these protocol extensions, we can find:

- Distributed Tracker
- Magnet Links

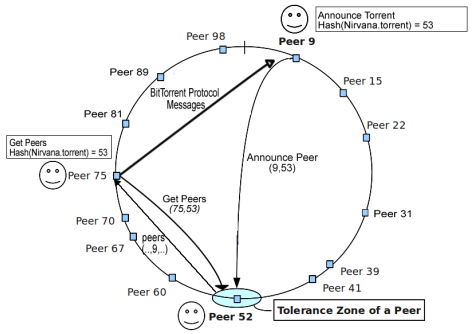

Fig. 1: BitTorrent DHT

- Connection Obfuscation

We will focus on one extension, the Distributed Tracker. This extension aims to replace the way a peer retrieves the list of peers sharing a file. Instead of using a central tracker, a peer can use a decentralized service, where every peer is responsible to index a group of torrents.

There are two protocols to operate a distributed tracker in BitTorrent. The distributed tracker of Azureus, which is only used by the Azureus client, now called Vuze, and the one of the Mainline client. Both are based on the Kademlia Protocol, but are incompatible between them. As we mentioned in section I we will focus our attention on the distributed tracker of the Mainline client, commonly known as Mainline DHT.

In decentralized tracking, every peer is in charge of indexing a group of torrents, mainly, those torrents that are close to the peer. The concept of closeness, as stated in Kademlia, is based on the XOR distance between a peer ID and the torrent ID. While the peer ID is chosen randomly, the torrent ID is obtained by hashing the info part of the torrent file.

In Figure 1 we can observe the basic procedure to announce a torrent and then retrieve the list of peers sharing it.

Let's assume **Peer 9** is sharing the **Nirvana** file. It announces the torrent through an *Announce* message, specifying that it is sharing the torrent. In this case **Peer 52** is responsible to index this torrent and save the entry, because its ID is the closest. Then, **Peer 75** wants to download the **Nirvana** file, so it sends a *GetPeer* message. Again, **Peer 52** receives this message and answers with a list of the peers already sharing that torrent, which in this case contains **Peer 9**. The *GetPeer* message will also add **Peer 75** to the list of peers sharing in the torrent.

This procedure is similar when contacting a central tracker, but it allows to distribute the load among all the peers in the DHT, since a central tracker receives a request for every torrent it indexes, whilst a peer only indexes a small group of torrents.

IV. Exploiting DHT vulnerabilities

We present in this section an architecture together with a set of experiments made to demonstrate the feasibility of large scale attacks against the Mainline DHT.

A. Distributed Architecture

In order to deploy a variety of experimental attacks in the decentralized tracker of BitTorrent, the distributed architecture

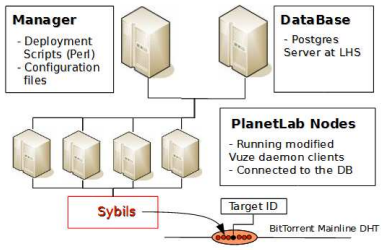

Fig. 2: Attack Architecture

depicted in Figure 2 is proposed. This architecture extends our previous work in the KAD network [2].

We used a group of PlanetLab nodes, along with a Postgres database server located in the LHS(High Security Lab located at INRIA Nancy). This server contains all the architecture configuration and maintains the data retrieved during the experimental evaluations.

We developed a modified version of a plug-in for the Vuze client, which allow us to connect to the Mainline DHT. This modified plug-in has the property to load the configuration from the database server and upload any data it gets.

The modified client does not have the capability to download any file, to share pieces of it or even to join any swarm. The client will respond to every message in the DHT level, but it will not reply to any request for handshake and, as a result, any upload request.

We take advantage of one well-known flaw in DHT's [14]: the free choice of ID's. Letting a DHT user to freely choose its ID leads the user to place itself in a determined space in the DHT. In our case, each modified client will carefully choose its ID so as to position itself close enough to a given torrent, which will allow us to receive most of the DHT protocol messages. Such configuration resembles the well-known Sybil attack, already perform in KAD by [13]. In our case the fake peers, so called the Sybils, are the modified Vuze clients on PlanetLab nodes.

B. Experiment in the Real Network

We configure the architecture with a set of Sybils, which will share between 110 and 140 bits in common with the ID of the targeted torrent. 110 bits in common is more than enough to be in the tolerance zone of any target in the DHT. The tolerance zone represents the torrents that a peer might index, which are those torrents whose IDs have at least 8 bits in common with the peer ID [4]. To improve the efficiency of our architecture, each Sybil knows and advertises the others, instead of discovering them through the regular DHT protocol. With this basic set-up, we successfully performed the following attacks:

1) **Spy Attack:** By logging all the requests each Sybil receives along with their information, like client IPs[1] and client ports we can estimate the amount of users downloading a given torrent with the DHT as a tracking option. Since this alternative tracking is activated in most of the clients, spying becomes more critical. Moreover, we can determinate,

[1]IP addresses are anonymized upon reception

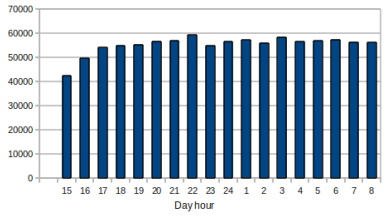

Fig. 3: Number of *GetPeer* requests per hour

for instance, when a client changes its IP address. This measurement can be done in a passive way, in opposition to, for example, a crawler.

We are also able to retrieve valuable data regarding the users behaviour, as well as the traffic in the DHT for a given torrent.

As an example, we deployed our architecture using 18 Sybils for almost 20 hours. As a target, we chose a popular TV series, Fringe (S03E01), which at that moment was the last available and highly popular episode. The purpose of the experiment was to observed all the *GetPeer* requests, as to monitor which peers search for this given torrent.

During the experiment, we received over 1 million of request for this torrent, in which over 91000 different IPs and over 61000 different ports were retrieved. Curiously, from all the requests, 9.2% were outside the tolerance zone of the Sybils and around a 34% of these requests were generated from the same group of IPs. This behaviour might indicate modified clients running particular experiments in the DHT.

Figure 3 shows the number of requests received per hour. During this particular period of time, there are well distributed over the duration of the experiment, probably due to the popularity of the torrent. This experiment shows how easy it is to monitor a given torrent, and retrieve those peers downloading it.

2) **Pollution and Eclipse:** During the experiments, we chose a popular film (Iron Man 2) and we deployed the same architecture configuration as defined before varying the number of Sybils. Figure 4 shows the amount of responses from Sybils that a normal client receives when searching for the given torrent. With 20 Sybils, a normal client will receive over 90% of Sybils. Increasing the number of Sybils allow us to increase the percentage of pollution.

Polluting a torrent means that when a normal client searches for that torrent, it will receive our Sybils clients instead. From this point, a wide set of attacks can be carried on in the Swarm. Attacking BitTorrent swarms from the DHT will require a longer analysis and it exceeds the scope of this work.

On the other hand, we can achieve an Eclipse attack. Each Sybil, when receiving a *GetPeer* message, does not respond instead of sending a valid answer. A normal client will not be able to retrieve any peers, as long as all the Sybils have the same behaviour.

During our evaluation, we were able to fully pollute a given torrent, but intermittently. The main reason is that, despite the Sybils are over 100 bits close to the target, the routing algorithm is not always capable of finding the closest peers, and regularly, normal peers are returned in the search response.

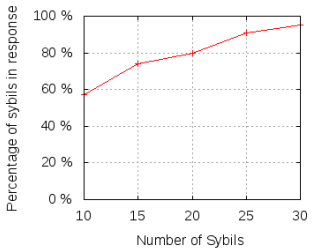

Fig. 4: Pollution over an Iron Man torrent

3) Geo-Localized Isolation: A simpler rule when answering a *GetPeer* request can lead to what we call a "Geo-Localized Isolation Attack". This attack aims to Eclipse a given torrent, but only to those peers in a given geographical region. With an external tool that gives us the geographical location of an IP, we can decide in real-time which requests to answer and which not.

Considering, as one example, that different copyright laws can apply for different countries, we can use this Localized Isolation to eclipse a given content for a particular country. As an alternative, instead of denying the content, we can simply spy a particular group of users, and retrieve statistical data. Clearly there are many situations in which this attack might become useful.

V. Protection Mechanisms

A. Considering the KAD Protection Mechanisms

As it has been stated before, the Mainline DHT is based on the Kademlia Protocol. The KAD network is also based on this protocol and it is one of the most deployed P2P networks currently active. However, KAD contains a number of security mechanisms that makes the network resilient to most well-known attacks. In our previous work [2] we conducted an evaluation of protection schemes in KAD, concluding that these protections are highly effective against attacks from a single machine, but can be bypassed if the attacker has a group of public IPs from different sub-networks, which already made the attack harder. Later on in [1] we proposed a distribution analysis scheme so as to detect distributed attacks.

Based on our previous work on the KAD network, we will conduct a measurement in the Mainline DHT network, and study its distribution of Ids. This study will show if a distribution analysis will be adequate to develop an Ids distribution protection.

B. IDs Distribution in BitTorrent Mainline DHT

An analysis distribution is applicable if safe lookups (when the peers involved in a search are not malicious, then a lookup is consider safe) in the network follow a theoretical distribution so as we can discriminate attacks.

Firstly, it is necessary to determinate what is a normal distribution of IDs in the Mainline DHT network. The theoretical distribution can be simply obtained by considering that if the ID of a peer is really randomly chosen, the distance between two different IDs, in term of number of common bits, also called prefix length, is based only on the number of peers in the network. Equation (1) describes the mean number of peers sharing a given prefix x, considering a network with N peers.

$$F(x) = \frac{N}{2^x} \tag{1}$$

Aware of the difficulty to accurately compute the number of peers in the network [4], we used the population estimator of the Mainline DHT plug-in for Vuze to get a rough estimate. Figure 5 shows the oscillation of the population during a one-day experiment, in which we are able to compute 4.2 million users in average.

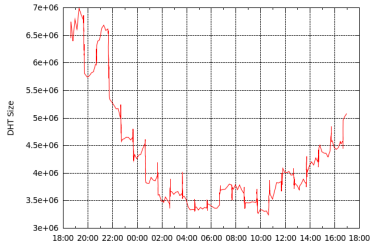

Fig. 5: DHT Size Estimation

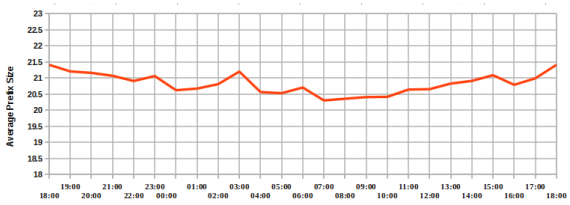

Fig. 6: Average Prefix Size for the best 8 peers found

The theoretical distribution can give us a precise idea of how the peers'ID are distributed among the DHT space, however a measurement in the real network is still needed, since the real distribution of prefixes among neighbours surrounding a target and found during a search might not follow the theoretical distribution, due to the efficiency of the routing algorithm of the Mainline DHT or dead nodes. Therefore, we conducted a measurement experiment aiming to assess this real distribution.

C. Distribution Experiment Setup

We conducted a 24-hours experiment in the network, in which we searched for random IDs and kept the 8 best contacts found during this search by the Mainline DHT plug-in for Vuze. This procedure was repeated every hour, starting at 18:00Hs (CEST), and searching for 35 different random keys each time. Using random keys allow us to avoid attacks on well-known keys, such as popular movies or TV series.

During this period of time, 861 keys were searched. Figure 6 shows the average prefix size of the 8 best contacts found during the experiment. The fact that IDs are randomly chosen give us the following relationship: if the number of peers in the network is divided by two, the average prefix size between two neighbors decreases in one bit. As it can be observed, the variation of population in the Mainline DHT network, seen in figure 5 impacts directly in the average prefix size, seen in figure 6. The network size varies from 3.2 millions to 6.4 millions whilst the average prefix size varies from 20.5

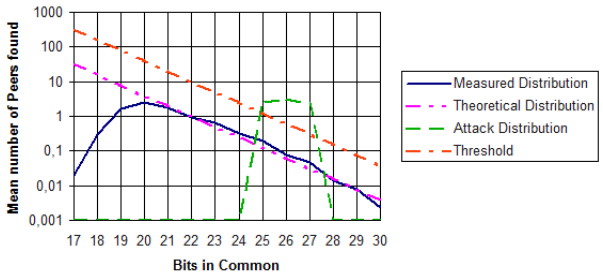

Fig. 7: Theoretical Distribution Vs. Measured Distribution

bits to 21.5 bits. This behaviour can be seen around 06:30Hs, where the population is 3.3 millions and the average prefix size decreases to 20.5.

After considering all the best contacts found and their distances to the target, we can compute the mean number of peers found for a given prefix. Figure 7 puts together both the theoretical with N = 4.2 million and the measured distribution of IDs. As it can be noticed, after the 20 bit prefix, the measured distribution follows perfectly the theoretical distribution, which indicates that the routing algorithm, in this case, is precise enough on average to find the closest peers possible. Figure 7 shows an example of an attack distribution which could be measured if an attacker introduces a group of Sybils close to a key (sharing 26-27-28 bits prefixes).

D. Protection against attacks through Distribution Analysis

In order to avoid those peers which are dangerously close to the key we are searching for, we need to take into account the distribution of peers' IDs and compare it with the theoretical distribution. If the distance between the two distributions is higher than a threshold, we should consider that the peers retrieved might be attackers. However, before computing the distance to detect stealth attacks, we can already avoid peers sharing more than 30 bits in common with the given key, since peers with these prefixes are highly unlikely to be found as being honest. The detection threshold is obtained from computing the distance between safe distributions and attack distributions. The attack distributions needs to contain a wide set of examples, from naive attacks (all peers sharing 30 bits with the target) to more smart distributions (barely above the normal distribution).

When a set of peers is considered as suspicious, we cannot precisely identified the bad nodes, so we remove all peers that increase the most the divergence, and replace them with new peers further from the target. Therefore, each attackers will not receive requests from the participants.

We prove the applicability of a distribution analysis in the Mainline DHT network, based on our previous work on the KAD network. However, a fixed theoretical distribution might not be accurate enough, since the network size varies considerably, contrary to the KAD network. A periodically calculation of the DHT's size and a re-computation of the theoretical distribution is a good approach to avoid a high false positive rate.

VI. Conclusion

In this paper we showed that the move towards a DHT-based tracking service in BitTorrent is not alone sufficient to provide bulletproof security to the service and privacy to the users. We have demonstrated through real deployment that efficient attacks can be performed on the Mainline DHT using the distributed architecture presented in the paper. In its current form with no build-in security mechanism, the DHT-based approach is even open to more vulnerabilities than the centralized tracker approach initially built into BitTorrent. We have shown that with few nodes, one can highly pollute or even eclipse a given content on the Mainline DHT.

While raising awareness is one of the main outcome of this paper, we are convinced that the solution is clearly to integrate new security mechanisms in the DHT. Firstly by implementing the set of protections already included in KAD which limit the number of peers per IP. Secondly by computing the distribution of IDs to identify malicious nodes in the network while keeping backward compatibility regarding random ID assignment. Future work will focus on extending the measurements and defining new metrics to identify malicious behaviour. These measurements and metrics will be compared to other DHTs that already implement a subset of the proposed security measures.

References

[1] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Efficient DHT attack mitigation through peers' ID distribution. In *HotP2P 2010*.
[2] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Evaluation of Sybil Attacks Protection Schemes in KAD. In *AIMS 2009*.
[3] Bram Cohen. Incentives build robustness in bittorrent. Technical report, bittorrent.org, 2003.
[4] Scott A. Crosby and Dan S. Wallach. An analysis of bittorrents two kademlia-based dht's. Technical report, Rice University, 2007.
[5] Ipoque. Internet study 2008/2009. http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009.
[6] Oliver Jetter, Jochen Dinger, and Hannes Hartenstein. Quantitative analysis of the sybil attack and effective sybil resistance in peer-to-peer systems. In *ICC 2010*.
[7] Jie Kong, Wandong Cai, and Lei Wang. The evaluation of index poisoning in bittorrent. *ICCSN 2010*.
[8] Jie Kong, Wandong Cai, Lei Wang, and Qiushi Zhao. A study of pollution on bittorrent. In *ICCAE 2010*, volume 3, 26-28 2010.
[9] Stevens Le-Blond, Arnaud Legout, Fabrice Le Fessant, Walid Dabbous, and Mohamed Ali Kâafar. Spying the world from your laptop – identifying and profiling content providers and big downloaders in bittorrent. *CoRR*, abs/1004.0930, 2010.
[10] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS 2001*.
[11] Michael Piatek, Tadayoshi Kohno, and Arvind Krishnamurthy. Challenges and directions for monitoring p2p file sharing networks-or: why my printer received a dmca takedown notice. In *HOTSEC 2008*.
[12] Georgos Siganos, Josep M. Pujol, and Pablo Rodriguez. Monitoring the bittorrent monitors: A bird's eye view. In *PAM 2009*.
[13] Moritz Steiner, Taoufik En-najjary, and Ernst W. Biersack. Exploiting kad: Possible uses and misuses. *ACM SIGCOMM CCR 2007*, 37.
[14] Guido Urdaneta, Guillaume Pierre, and Maarten van Steen. A survey of DHT security techniques. *ACM Computing Surveys 2009*.
[15] Scott Wolchok and J. Alex Halderman. Crawling bittorrent dhts for fun and profit. In *To appear in Proc. WOOT 2010*.