

Real-World Sybil Attacks in BitTorrent Mainline DHT

Liang Wang
Dept. of Computer Science
University of Helsinki
Helsinki, Finland

Jussi Kangasharju
Helsinki Institute for Information Technology
University of Helsinki
Helsinki, Finland

Abstract—Distributed hash tables (DHT) are a key building block for modern P2P content-distribution system, for example in implementing the distributed tracker of BitTorrent Mainline DHT. DHTs, due to their fully distributed nature, are known to be vulnerable to certain kinds of attacks and different kinds of defenses have been proposed against these attacks. In this paper, we consider two kinds of attacks on a DHT, one already known attack and one new kind of an attack, and show how they can be targeted against Mainline DHT. We complement them by an extensive measurement study using honeypots which shows that both attacks have been going on for a long time in the network and are still happening. We present numbers showing that the number of sybils in the Mainline DHT network is increasing and is currently around 300,000. We analyze the potential threats from these attacks and propose simple countermeasures against them.

I. INTRODUCTION

BitTorrent is the dominant P2P software for file sharing and content distribution. Originally BitTorrent was based on a centralized tracker, where one tracker would host one or more swarms with each swarm corresponding to one file being distributed. More recently, distributed variants of BitTorrent have been developed. In these variants, the tracker functionality has been spread over the peers in the system using a distributed hash table (DHT). Popular DHT-based BitTorrent variants include VUZE and Mainline DHT (MLDHT).

In this paper we focus on MLDHT and its security problems. DHTs have long since been known to have inherent security issues, e.g., [1] and our work in this paper shows how MLDHT has failed to take these issues into account. Effective lack of protection on MLDHT implies that it is very easy for an attacker to monitor large fractions of traffic on the system or even hijack large parts of the system. Due to these vulnerabilities of MLDHT, it turns out that even with very modest resources (a few computers), a determined attacker can effectively make a very large-scale attack. We base these claims on the two forms of attack we present in this paper and on our measurement data, which shows that both attacks are being performed on a large scale in MLDHT. However, we are only able to observe the presence of these attacks, but not their actual impact; our work only shows the potential for damage but we have not observed any actual malice happening on the network.

Specifically, the contributions of this paper are as follows:

- We present two possible routing table attacks on MLDHT system: *horizontal* and *vertical attack*, both based on attacking the routing tables of nodes in the system.
- We analyze the damages to the system caused by such attacks and their potential to violate user privacy.
- Through extensive measurements, we discover that both of these attacks are on-going in current MLDHT and report on their detailed implementation and discuss the implications of this.
- We discuss how existing security solutions could be applied to MLDHT in order to make it less vulnerable to attacks.

This paper is structured as follows. Section II reviews the background on BitTorrent Mainline DHT. In Section III we present the two attacks and discuss the possible damage on the system. In Section IV we discuss the real-world attacks we found and analyze their potential impact. Section V discusses possible defenses against the attacks. We discuss related work in Section VI, and conclude our paper in Section VII.

II. BACKGROUND ON MAINLINE DHT

In the BitTorrent system, to join a swarm, a peer needs to get meta information first. In standard BitTorrent, the meta information can be obtained from the torrent file, which also contains a list of centralized trackers to help a peer get the initial peer set to bootstrap the download.

Partly due to legal issues, but also based on improving the service availability and system robustness, distributed trackers have been developed. BitTorrent has two independent, incompatible distributed tracker implementations, even though both are based on the Kademlia DHT [2]. One is VUZE [3] and the other is MLDHT.

MLDHT implements the minimum functionality of Kademlia. In MLDHT, both peers and content each have a 160-bit string as its ID. Content IDs are also known as infohashes. A peer uses this infohash to obtain the meta information and initial peer set. MLDHT supports four control messages:

- 1) PING: probe a node's availability. If the node fails to respond for some period of time, it will be purged out of the routing table.
- 2) FIND_NODE: given a target ID, this message is used to find the K closest neighbors of the ID.
- 3) GET_PEERS: given an infohash, get the initial peer set.

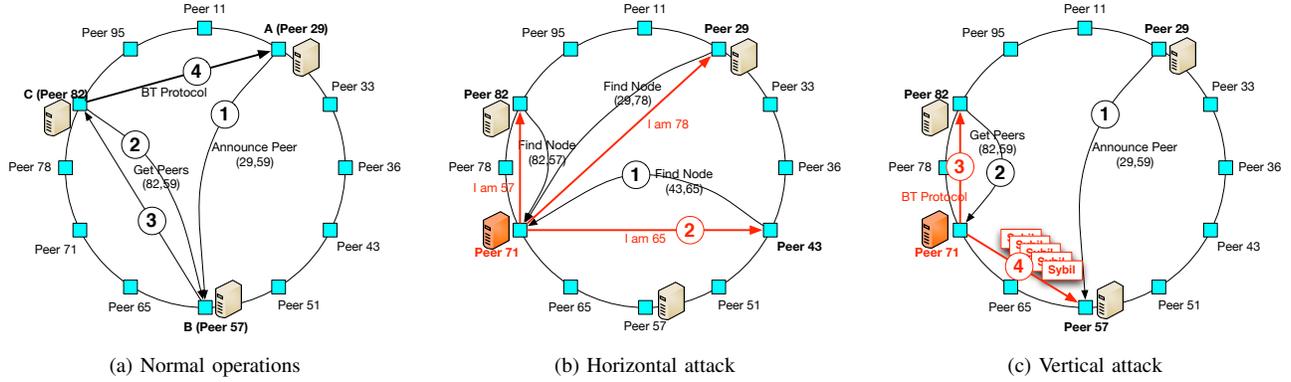


Fig. 1: Illustration of normal operations and two kinds of attacks in MLDHT

4) ANNOUNCE_PEER: a peer announces it belongs to a swarm.

Figure 1a illustrates normal operation in MLDHT. Suppose we have 3 nodes A , B and C . A holds a file with infohash $x = 59$. Assume B is responsible for storing x its peer set. Node C wants to download the file.

First, A should publish the file by storing x at B . A will call GET_PEERS iteratively to get closer and closer to B , and finally reaches it. Then, A will use ANNOUNCE_PEER to tell B he is downloading (sharing) a file with infohash x . B then stores A 's contact information in the corresponding peer set for x . Since A is the publisher, it is the only one in the peer set at the moment.

When A sends the GET_PEERS messages, two possibilities emerge. If the queried node knows this infohash already and stores some peers in the corresponding peer set, it will respond with the peer set. If it does not know the infohash, it will respond with the k closest nodes to the infohash in its routing table. In such a way, A will get closer and closer to B , and finally reach B and the search finishes.

For C to download the file, it should get x first. It will do exactly the same as A did before by using GET_PEERS to approach B . Since B already saved the peer set for x , C can obtain the initial peer set from B . C joins the swarm, sets up connections to the peers in peer set, and gets metadata (torrent-file) from other peers using BitTorrent extension protocols [4], [5]. Then the download process starts.

III. ATTACKS & DAMAGES

Sybil attack is a well-known attack originally introduced by Douceur [6]. The idea is to inject multiple fake identities into the system, and use them as a starting point to perform further attacks. It can also be considered as routing table attack, which tries to pollute users' routing table by inserting sybils into it.

In practice, there are two major strategies to perform routing table attacks, and we name them as *horizontal attack* and *vertical attack* based on their characteristics. In this section, we show how these two attacks work, and how to combine them into a more dangerous attack. To simplify our following discussion, we assume a Kademlia-based system [2], consists of N nodes, each node maintaining a k -bucket routing table.

```

foreach incoming message  $M$  do
  switch  $M$  do
    case PING
      | PONG with random ID
    case FIND_NODE
      | Reply as the owner of the queried ID
    case GET_PEERS
      | Save the infohash and keep silent
    otherwise
      | Drop  $M$ 
    endsw
  endsw
end

```

Algorithm 1: Horizontal Attack

A. Horizontal Attack

Horizontal attack spreads sybils widely across the system. The aim is to pollute as many routing tables as possible. The number of sybils in one routing table is not the concern; the goal is to pollute the maximum number of routing tables. A successful horizontal attack can let the attacker sniff most of the the control messages and therefore hijack the system.

The k -bucket mechanism implies that an attacker can effectively intercept messages if he has at least one sybil among a node's k closest neighbors. It means an attacker only needs to inject $\frac{N}{k-1}$ sybils. However, we already know that the average number of hops in routing a message in the system is $O(\log(N))$. Because k is usually less than 8, $O(\log(N))$ will eventually become bigger than k as the network size grows. Thus the minimum sybils needed to perform a successful horizontal attack is in fact $\frac{N}{\max(k, \log(N))}$.

The next question arisen is how much resources we need to perform the attack. The most straightforward way is running one node instance for each sybil, which obviously requires lots of computation and traffic resources. However, by exploiting the MLDHT protocol, this can be done with very limited amount of physical resources. Figure 1b and Algorithm 1 show how horizontal attack works in MLDHT. (This algorithm depicts an actual horizontal attack on MLDHT which has been active since late 2010 and is still on-going, see Section IV.)

As Algorithm 1 shows, the attacker first needs to promote

```

Wait for target  $tID$ 
for  $i \leftarrow 1$  to  $k$  do
| Create sybil with  $ID \leftarrow tID + i$ 
end

```

Algorithm 2: Vertical Attack

himself to some other nodes to bootstrap the attack. Then he enters into an infinite loop waiting for incoming messages. For different message types, the attacker has to respond differently to make the attack efficient. Due to the routing table refresh mechanism in MLDHT, PING messages have to be answered quickly in order to avoid being purged. Answering FIND_NODE messages is the critical part of the attack. The normal behavior is to reply with k nearest nodes to the queried ID, however, the attacker puts only himself in the response and claim he is the owner of queried ID. This makes the querying node believe that the attacker has that ID.

From the effectiveness and efficiency perspective, the interesting part of this algorithm is after bootstrapping, sybils spread quickly across the system like virus. This is because the sybils can be injected into a victim's routing table either directly by attacker, or indirectly by polluted nodes, i.e., the victims help the attacker to promote sybils in the system.

In MLDHT, this attack can be carried out with very limited physical resources. First, the attacker only has to answer two types of messages, PING and FIND_NODE; he can ignore every other message. Second, the MLDHT protocol does not require the querying node to check the consistency between the ID in PONG and the corresponding entry in routing table; this means that the attacker can answer PINGs with random ID. This further implies the attacker does not need to maintain any state information to carry out his attack.

B. Vertical Attack

Vertical attack attempts to insert as many sybils as possible in one specific routing table. Figure 1c and Algorithm 2 show how the attack works. Although similar to eclipse attack¹, vertical attack is broader because it can target a content ID.

After launching an attack, the attacker waits for an interesting target ID, either node ID or content infohash. Then, the attacker inserts sybils close to the target to "isolate" it from the others. The algorithm also takes advantage of k -bucket mechanism. If the target is a node ID, the attacker not only gets a copy of every $\langle \text{key}, \text{value} \rangle$ stored at target node, but also intercepts all queries. If the target ID is content infohash, the attacker can take full control of the content.

This attack is made easy by MLDHT allowing nodes to pick their own ID. Interestingly, this weakness of DHTs has been known for over a decade [1], but MLDHT is still vulnerable to it. We will return to this issue in Section V.

C. Hybrid

Real-life attacks usually are a mixture of the two and combining them, large-scale attacks against the system can be

¹Eclipse attack [7] is an attack where the attacker tries to corrupt honest nodes' routing tables by filling them with references to malicious nodes.

effectively deployed. First, the attacker should launch a horizontal attack to make himself well-known by as many others as possible. As more and more control messages are directed towards the attacker, he basically takes over the control layer. Then he can launch the vertical attacks on interesting targets. Effectively this leads to the attacker controlling the system.

D. Potential Damage

Using the hybrid attack as a base, we analyze the potential damages from system, content and user privacy perspective respectively. In Section IV we present numbers from real measurements to support the ease of performing these attacks.

1) *System*: Obviously, since the attacker can take control of most of the routing in the system, he is able to introduce delays and lookup failures at will. Previous work [8]–[10] discussed the possibility of performing a distributed denial-of-service attack by exploiting BitTorrent system. Our work shows that if the MLDHT is compromised, such an attack can be easily performed and with potential for severe damages.

As shown in Section IV, these attacks are widely going on in the real world. From a research point of view, these present an interesting challenge. In particular, measurement work which attempts to discover system behavior may be biased because of the presence of sybils in the system. For example if a study attempts to measure session times by contacting nodes and seeing how often they respond, then an on-going horizontal attack would skew the results upwards, since the attacker would always reply, which would be interpreted as a very long session by that ID (actually all the IDs used by the attacker).

We are not aware of any previous study of MLDHT which takes into account the presence of these types of attacks. Given the wide spread of the attacks and their impact, this may put some results from previous studies on MLDHT into question.

2) *Content*: Attacker can also manipulate any content easily if he successfully hijacks the system. As the results in [11] shows, the attacker only needs to insert 20 sybils to make a client receive over 90% of sybils. He can pollute the target content, carry out an eclipse attack, or censor certain content.

3) *User Privacy*: These attacks put a lot of traffic in the hands of the attacker. This means that any time a user requests any content, the likelihood of the attacker knowing this is very high. In other words, privacy on MLDHT is likely to be non-existent and wide-spread monitoring of users is possible. The ability to choose different IDs may help because it hinders the ability of the attacker to correlate actions between sessions, but it does not protect privacy within a session.

L. Blond et al. showed in [12] that user privacy can be compromised by exploiting a popular BitTorrent Portal and continuously monitoring swarms. With the attacks we present, such violations can be done more easily and at larger scale.

IV. ATTACKS IN MLDHT

We have been monitoring MLDHT network evolution since December 2010. The monitoring system keeps taking samples from different zones² in a fixed 30 minutes interval. Each

²An n -bit zone refers to the nodes share common n -bit prefix.

sample contains information such as node’s ID, IP, port and status. The monitoring system selects a random zone for every round of measurement. For further details on the measurement methodology and results, please see [13].

During the course of our measurements, we observed some strange phenomena in MLDHT. A careful analysis revealed that these were examples of horizontal and vertical attacks being performed on the system. We set up multiple different honeypots to get a good picture of their behavior and report our findings in this section. Although we did not observe any overtly malicious behavior, we analyze the potential threats from these activities.

We first describe our honeypot setup and then the observed attacks in detail.

A. Honeypots

We deployed three different honeypots for studying the anomalous behavior. The background in Section II captures the interactions on the DHT level, but there is also the BitTorrent protocol level above that. On the BitTorrent level, nodes would attempt to get the metadata for the infohash and possible attempt to start downloading the file. Our honeypots acted on both the DHT and BitTorrent levels.

B. Detector/Honeypot

To monitor suspicious behavior in the system, we set up a detector, which had two purposes. The first is identifying horizontal attacks and the second is identifying as many suspicious nodes as possible. After discovery, we set up a specific kind of honeypot for the identified suspicious node.

For the first functionality, we periodically use `FIND_NODE` to find some random IDs in the system. Due to the randomness, the ID we look for should not exist with very high probability. However, a horizontal attacker will always claim he is the actual owner of queried ID. By checking who claims the ownership of those non-existent IDs, we can identify attackers.

For the second functionality, we periodically use `GET_PEERS` to search for some non-existent infohashes in the system. Anybody coming to us with those infohashes will be marked as suspicious and stored in the database for post-analysis. This is because the protocol does not specify any reaction to a `GET_PEERS` by an intermediate node.

C. MLDHT Honeypot

Our detector identified several suspicious activities and in order to investigate them closer, we generated 10 infohashes very close to our own honeypot’s ID. Then we used `GET_PEERS` message to search for those infohashes. Any node receiving our message knows then that such an infohash exists, but they are not supposed to do anything about it, according to protocol. In order for this honeypot to work, we need to have those infohashes close to us so that if anyone follows them, they will come to us.

Because nodes are not supposed to react to these messages (except by replying with their k closest nodes to that infohash), any node approaching us is by definition suspicious. In fact,

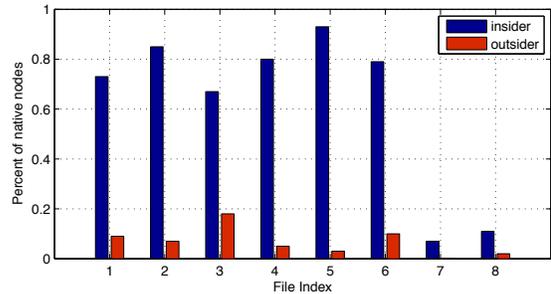


Fig. 2: Evidence for Mr.ISP localizing BitTorrent traffic

a large number of nodes attempted to reach us and used `GET_PEERS` (with DHT *scrape*-extension [14]) to probe us. Some of them even attempted to establish connections on the BitTorrent level (see below for the third honeypot).

We saw that two entities showed up with large shares in this MLDHT honeypot. One of them we will cover in more detail under the third honeypot. The other was using IP addresses belonging to a large, international ISP. In the following, we call this entity “Mr. ISP”. Those nodes used `GET_PEERS` message to get the peer set from us. However, the interesting part is that the node ID they claimed to possess was the ID of the infohash plus one. In other words, they attempt to intercept all query traffic for this infohash, by claiming to be right next to it. This methodology is identical to the methodology for localizing BitTorrent traffic in DHT-based overlays presented in [15], [16]. However, Mr. ISP is much more aggressive than the methodology described in those two papers. *Effectively, this is a vertical attack on the system.*

Based on work presented in [15], [16], we suspect Mr. ISP is doing traffic localization in MLDHT. To verify our hypothesis, we designed the following experiment. We chose 8 torrents, 6 of them from the most popular files on The PirateBay, and 2 unpopular files from a Chinese sharing website. For each file, we set up two instrumented clients: insider and outsider. We let the insider connect to MLDHT via a free proxy running within Mr. ISP’s network, while the outsider runs on our network. From Mr. ISP’s point of view, insider is a client running within its network and outsider is not. Both clients try to join the swarm and get the initial peer set from Mr. ISP, and we study the percent of native nodes (those within the same ISP) in the returned peer set from Mr. ISP. We only requested the peer set from Mr. ISP without joining the distribution of the file.

Figure 2 shows the fraction of native nodes received by the two clients for each of the file. Files 1 through 6 are the popular files and for them, the fraction of native nodes in insider’s peer set is significantly higher than that in outsider’s, peaking at over 90%. This clearly shows that Mr. ISP attempts to localize traffic to its own network for its own clients. For unpopular files, the fraction of native nodes is low because there are very few nodes interested in those files. Still, the inside client gets a little bit more native nodes.

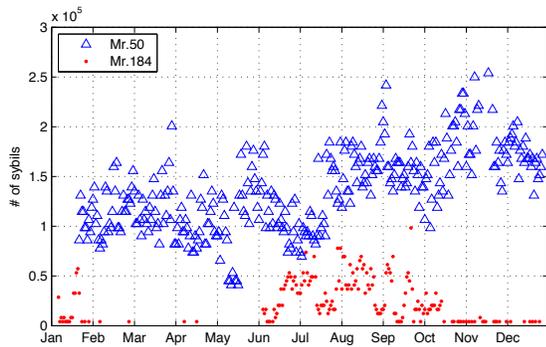


Fig. 3: Mr.50 and Mr.184 density in 12-bit zone in 2011

D. BitTorrent Honeypot

As mentioned, our second honeypot also captured another interesting source of anomalous traffic. This traffic first appeared in our logs in January 2011 and has been going on since then. We set up our honeypots in September 2011, thus we have no record of Mr. ISP’s previous behavior, but because this second entity behaves differently, we were able to trace him throughout our whole study. We call him Mr. 50, because his IP address can be traced to Amazon’s EC2 and it starts with 50. There is also a Mr. 184, likewise coming from Amazon’s EC2, who is far less active than Mr. 50, but appeared around the same time. We suspect they are both controlled by the same person, because both implement the same, identically erroneous version of the MLDHT protocol. We have attempted to see if they collaborate by feeding information to only one of them, but have never managed to observe any sharing of information between them. However, because of the erroneous protocol implementation (described below), we strongly suspect them to be the same person.

Mr. 50 was the only entity to attempt BitTorrent level communications. Mr. 50’s goal seems to be monitoring content on MLDHT and *he performs a horizontal attack*, as presented in Section III-A.

Figure 3 shows the estimate of sybils injected by Mr. 50 and Mr. 184. From mid-January till the end of July, the number of sybils from Mr. 50’s remains around 100,000; the number increases after August. This means that the number of sybils injected by Mr. 50 varied between 80,000 and 290,000. Mr. 184 is less active and we suspect it to be a test node because it appeared before Mr. 50 in January and after Mr. 184’s second appearance in Summer 2011, Mr. 50’s activity rose.

We found out that Mr. 50 implements only part of the DHT level protocol. He only answers PING and FIND_NODE messages. PING is used to check if another node is still online and Mr. 50 answers them immediately. However, he does not answer with the ID we queried for in the previous FIND_NODE message, but uses a random ID instead (in violation of the DHT level protocol). FIND_NODE is used by a node to find another node with a specific ID. Mr. 50 answers a FIND_NODE by claiming he knows the node who possesses the ID he was asked about and gives only his own

ID in the response. The net effect of these two messages is that any node interacting with Mr. 50 will assume him to be a good source for *any* content in the network.

Because of the random and incorrect replies, we conjecture that Mr. 50 does not keep any state, but has simply injected a large number of (logical) sybils in the network. Some nodes might refuse to accept him in their routing tables because of the incorrect replies, but apparently this is not a concern for him and allow him to use far less resources in monitoring.

Another thing worth mentioning is that the delay in answering PING is much shorter than for FIND_NODE. Because his answers to PING are random, answering them quickly is trivial. Interestingly, even though his answers to FIND_NODE should be equally simple (his own ID), the reply takes a much longer time to arrive. We speculate that this is because of logging the FIND_NODE messages for further processing.

Concerning the other two MLDHT messages, GET_PEERS and ANNOUNCE_PEER, Mr. 50 does not answer either of them. Given our hypothesis that he is monitoring content (see more evidence below), this is logical since answering these two messages only causes overhead but does not help in monitoring content since they have no effect on routing tables.

Finally, after getting the infohash we fed him, Mr. 50 tries to set up BitTorrent level communication to get the corresponding metafile. He makes the assumption that our BitTorrent TCP port number is the same as our MLDHT UDP port number. Normally, this information would have been sent with ANNOUNCE_PEER, but since he does not appear to store them, he does not have this information and is forced to guess.

When BitTorrent level communication is established, Mr. 50 requests the metafile using BitTorrent extension protocol [4]. During the handshake, both nodes are supposed to identify their clients and versions, but Mr. 50 simply echoes back whatever we send him. If the initial retrieval fails, Mr. 50 will use one of two additional nodes on Amazon’s EC2 (with IP prefix 50) to retrieve the metafile later. These additional IP addresses appear only in this context.

We lured Mr. 50 into our BitTorrent honeypot by sending a GET_PEERS with infohash of a fake metafile, and we placed ourselves near that infohash. Even though Mr. 50 does not reply to GET_PEERS, his arrival into our honeypot confirms that he is indeed logging that information. Recall that we had also Mr. 184 with identical behavior monitoring the network. We fed different infohashes to them using this method, but never observed any sharing, i.e., an infohash given to Mr. 50 only got requests from Mr. 50, but never from Mr. 184 and vice versa. Given their identical implementations of the DHT protocol, we conjecture them to be under the control of the same person, but have no concrete proof either way.

E. Analysis and Threats

Although we did not see any overtly malicious behavior from Mr. ISP or Mr. 50, their actions do cause concern in terms of their potential for malice or invasion of privacy. The root cause of the problem is the ability of nodes to freely

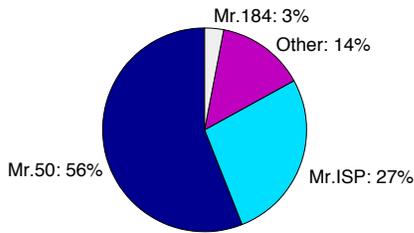


Fig. 4: Distribution of Sybil attacks

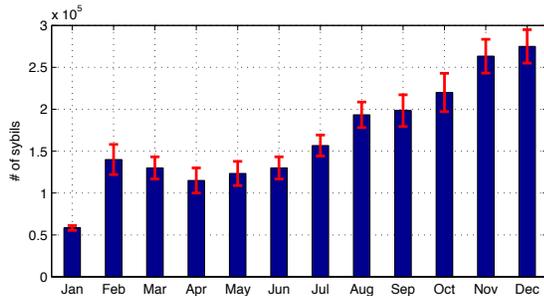


Fig. 5: Estimated # of sybils in MLDHT in 2011

choose their ID. This has long been known to be a problem for DHTs [1] and greatly facilitates large-scale sybil attacks.

In concrete terms, we are certain of the following:

- *Mr. ISP is able to monitor a very large fraction of all traffic on MLDHT.* He seems very resourceful, since he injects an large number of sybils and is able to keep state for all of them. The safe assumption is that any content fetching on MLDHT can be recorded by Mr. ISP.
- *Mr. 50 is extensively monitoring content publishing activities on MLDHT.* Because he does not seem to keep state, it is likely his resources are limited. He only uses a few nodes but because of the lack of state-keeping, can monitor a large part of content publishing on the network.

Considering the potential damage they could cause, we can see the following avenues for exploitation. Mr. ISP, due to his larger amount of resources and aggressive setting up of sybils, could easily hijack the whole MLDHT network or at least very large parts of it. Mr. 50 is in principle able to do the same as well, although if clients were to drop his “incorrect” replies due to his lack of state, his attack would not be successful. (We do not know how the many different clients react to his messages.) Results in [11] show that even a small number of sybils is sufficient to hijack any given content. With more resources, a wider hijack becomes possible.

Figure 4 shows the fraction of sybils attributed to the different attackers in 2011. Figure 5 shows the average total number of sybils in the system in 2011 with the standard deviation. As we can see, the number of sybils in the system is increasing, which represents a worrying trend.

By not answering some protocol messages, Mr. 50 is in fact disrupting the operation of the network. However, because most clients typically send several requests in parallel and only part of them go through Mr. 50, it is likely that his actions at

worst only slow down some downloads a little bit, but do not effectively block downloads.

We must re-iterate that we have not observed any malicious behavior. However, causing considerable damage would be easy for any of the entities behind the attacks we discovered.

V. POSSIBLE SOLUTIONS

Letting a node choose its own ID has long been known to be a vulnerability [1]. Work in [7] provides a good survey of various DHT security techniques, based on which we come to the following possible mitigating actions.

Douceur showed in [6] that the only practical way to guarantee one-to-one mapping between virtual identities and physical entities is letting a logically central, trusted authority to issue all the identities. But this will also increase processing and administrative overheads and put barriers on legitimate nodes joining the network. Castro et al. [17] suggested to charge money for the certificate to prevent attackers from getting enough identities for a sybil attack. They also suggested including IP address into the certificate. However, an IP-based scheme requires special solutions for nodes behind firewalls.

Wang et al. proposed an approach called *net_print* in [18] to build secure DHT. The idea is to adopt physical network characteristics like MAC address and RTT into identifying nodes. The disadvantage is that changes in network conditions may cause the subsequent identity test to fail. In [19], [20], Bazzi et al. proposed *network coordinates* based on a similar idea.

Solutions based social networks have also been proposed in [21]–[23], but they only apply when a social network is feasible in the system.

Work in [11], [24] proposed a *distribution analysis* to detect vertical attacks. The idea is to identify attacks by discovering suspiciously high node densities for an estimated network size. However, the same method can also be used by the attacker to bypass the attack detection, because the attacker can also monitor the system size and make sure the number of injected sybils is below the alarm threshold. Furthermore, distribution analysis cannot protect against a horizontal attack, which increases the node density uniformly across the system.

Even though much previous work has been done to improve DHT security, there is no cure-all solution. However, we can make the attacks practically infeasible by increasing the deployment difficulties (overheads), as suggested in [18], [25].

To fight against horizontal attacks, a simple fix of the protocol would help. A node should explicitly check whether the ID in the PONG message is the same as the one in the routing table. In addition, we can include a challenge in the PING message. For a normal client, this only slightly increases the overheads. However, this fix forces the attacker maintain the state information for each previous contacted nodes, and also increases the computational overhead significantly.

VI. RELATED WORK

There have been a lot of work done in peer-to-peer measurement [26]–[35], and lots of discussions in security issues of

DHT-based system, such as [6]–[11], [21]. For example, [21] studied sybil attack, and [8]–[10] studied the DDoS attack by exploiting P2P system. There have been some small-scale anomalies reported in the previous work like abusing certain IDs in KAD, but no large-scale attacks have been identified in the real world.

Dhungel et al. [36] studied *piece attack* and *connection attack* identified in the traditional BitTorrent architecture. Their work is also based on measurement-based study. Compared with their work, we study the security issues in the distributed tracker architecture and two different types of routing attacks.

Timpanaro et al. proposed a *distribution analysis* mechanism to detect MLDHT attack. Their work focuses on the vertical attack, and as we discuss above, their solution can be bypassed with slight changes to attack mechanism, and is not sufficient to protect the system.

VII. CONCLUSION

In this paper we have considered vulnerabilities in BitTorrent Mainline DHT and have found out that not only is it very vulnerable to simple attacks, but also that attack-like activities are happening on a wide scale. We have identified two routing table attacks, horizontal and vertical attack, and discussed their potential damages. Through an extensive measurement study since December 2010, we have identified that both of these attacks are happening in the real network. We have analyzed their exact behavior through honeypots and have shown the scale of the on-going activities. We must stress that we have no concrete proof of actual malicious activities; our work only shows that the scale of attacks is large enough for this to be a concern. Finally, we have considered existing security solutions and discussed their suitability for protecting Mainline DHT.

REFERENCES

- [1] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *Proc. of IPTPS*, Cambridge, MA, Mar. 2002.
- [2] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *Proc. of IPTPS*, Mar. 2002.
- [3] Vuze, "Distributed hash table," <http://wiki.vuze.com/w/DHT>.
- [4] A. N. Greg Hazel, "Bep 9: Extension for peers to send metadata files," http://www.bittorrent.org/beps/bep_0009.html, 2008.
- [5] G. H. Arvid Norberg, Ludvig Strigeus, "Bep 10: Extension protocol," http://www.bittorrent.org/beps/bep_0010.html, 2008.
- [6] J. R. Douceur, "The Sybil attack," in *Proc. of IPTPS*, Mar. 2002.
- [7] G. Urdaneta, G. Pierre, and M. V. Steen, "A survey of dht security techniques," *ACM Comput. Surv.*, vol. 43, no. 2, pp. 8:1–8:49, Feb. 2011.
- [8] N. Naoumov and K. Ross, "Exploiting p2p systems for ddos attacks," in *Proceedings of InfoScale*, 2006.
- [9] K. El Defrawy, M. Gjoka, and A. Markopoulou, "Bottorrent: misusing bittorrent to launch ddos attacks," in *Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet*, 2007.
- [10] J. Liang, N. Naoumov, and K. W. Ross, "The index poisoning attack in p2p file sharing systems," in *Proc. of IEEE Infocom*, 2006.
- [11] J. Timpanaro, T. Cholez, I. Chrismont, and O. Festor, "Bittorrent's mainline dht security assessment," in *IFIP NTMS*, Feb. 2011.
- [12] S. Le Blond, A. Legout, F. Lefessant, W. Dabbous, and M. A. Kaafar, "Spying the world from your laptop: identifying and profiling content providers and big downloaders in bittorrent," in *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, 2010.
- [13] L. Wang and J. Kangasharju, "Monitoring bittorrent mainline dht," Department of Computer Science, University of Helsinki, Tech. Rep., 2012, available at <http://www.cs.helsinki.fi/u/jakangas/Papers/BT-Report.pdf>.
- [14] "Bep 33: Dht scrapes," http://www.bittorrent.org/beps/bep_0033.html, 2010.
- [15] M. Varvello and M. Steiner, "Traffic localization for dht-based bittorrent networks," in *Proc. of IFIP/TC6 Networking*, Valencia, May 2011.
- [16] M. Steiner and M. Varvello, "Peer-to-peer traffic localization as a service," in *Proc. of INFOCOM (Demo)*, Shanghai, China, April 2011.
- [17] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 299–314, Dec. 2002.
- [18] H. Wang, Y. Zhu, and Y. Hu, "An efficient and secure peer-to-peer overlay network," in *IEEE LCN*, Nov. 2005.
- [19] R. A. Bazzi and G. Konjevod, "On the establishment of distinct identities in overlay networks," in *Proceedings of ACM PODC*, 2005.
- [20] R. Bazzi, Y.-r. Choi, and M. Gouda, "Hop chains: Secure routing and the establishment of distinct identities," in *Principles of Distributed Systems*, 2006.
- [21] G. Danezis, C. Lesniewski-laas, M. F. Kaashoek, and R. Anderson, "Sybil-resistant dht routing," *European Symp Research in Computer Security ESORICS 2005*, pp. 305–318, 2005.
- [22] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *IEEE Symposium on Security and Privacy*, may 2008, pp. 3–17.
- [23] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *Proceedings of ACM SIGCOMM*, 2006.
- [24] T. Cholez, I. Chrismont, and O. Festor, "Efficient dht attack mitigation through peers' id distribution," in *IEEE Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, april 2010.
- [25] N. Borisov, "Computational puzzles as sybil defenses," in *IEEE P2P Conference*, Sep. 2006.
- [26] M. Steiner and E. W. Biersack, "Crawling azureus," Institut Eurecom, France, Tech. Rep. EURECOM+2495, 06 2008.
- [27] S. Wolchok and J. Halderman, "Crawling BitTorrent DHTs for Fun and Profit," in *Proc. 4th USENIX Workshop on Offensive Technologies*, 2010.
- [28] D. Stutzbach and R. Rejaie, "Capturing accurate snapshots of the gnutella network," in *Proceedings of IEEE Infocom*, 2005.
- [29] G. Memon, R. Rejaie, Y. Guo, and D. Stutzbach, "Large-scale monitoring of DHT traffic," in *Proceedings of the 8th international conference on Peer-to-peer systems*, 2009.
- [30] M. Steiner, E. Biersack, and T. Ennajary, "Actively monitoring peers in KAD," in *Proceedings of IPTPS*, 2007.
- [31] J. Falkner, M. Piatek, J. John, A. Krishnamurthy, and T. Anderson, "Profiling a million user DHT," in *Proceedings of ACM Internet Measurement Conference*, 2007.
- [32] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of ACM SIGCOMM Internet measurement conference*, 2006.
- [33] M. Steiner, T. En-Najjary, and E. Biersack, "Exploiting KAD: possible uses and misuses," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 5, pp. 65–70, 2007.
- [34] —, "Long term study of peer behavior in the KAD DHT," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1371–1384, 2009.
- [35] —, "A global view of kad," *Proceedings of ACM Internet Measurement Conference*, Oct. 2007.
- [36] P. Dhungel, D. Wu, and K. W. Ross, "Measurement and mitigation of bittorrent leecher attacks," *Computer Communications*, vol. 32, no. 17, pp. 1852 – 1861, 2009.