IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

Hijacking the Vuze BitTorrent network: all your hop are belong to us

ISSN 1751-8709
Received on 11th July 2014
Accepted on 24th October 2014
doi: 10.1049/iet-ifs.2014.0337
www.ietdl.org

Eric Chan-Tin[1] ✉, *Victor Heorhiadi*[2], *Nicholas Hopper*[3], *Yongdae Kim*[4]

[1]*Computer Science Department, Oklahoma State University, Stillwater, OK, USA*
[2]*Department of Computer Science, University of North Carolina Chapel Hill, Chapel Hill, NC, USA*
[3]*Computer Science & Engineering Department, University of Minnesota, Minneapolis, MN, USA*
[4]*Department of Electrical Engineering, KAIST, Daejeon, Republic of Korea*
✉ *E-mail: chantin@cs.okstate.edu*

Abstract: Vuze is a popular file-sharing client. When looking for content, Vuze selects from its list of neighbours, a set of 20 nodes to be contacted; the selection is performed such that the neighbours closest to the content in terms of Vuze ID are contacted first. To improve efficiency of its searches, Vuze implements a network coordinate system: from the set of 20 to-be-contacted nodes, queries are sent to the closest nodes in terms of network distance, which is calculated by the difference in network coordinates. However, network coordinate systems are inherently insecure and a malicious peer can lie about its coordinate to appear closest to every peer in the network. This allows the malicious peer to bias next-hop choices for victim peers such that queries will be sent to the attacker, thus hijacking every search query. In our experiments, almost 20% of the search queries are hijacked; the cost of performing this attack is minimal – less than $112/month.

1 Introduction

Network coordinate systems [1, 2] assign coordinates to every node in the network, which allows an accurate prediction of network latency between any pair of nodes. Network coordinates have been used in various applications, such as in online game match-making [3], finding closest nodes to download content from in a file-sharing network [4], reducing inter-ISP communications [5], detecting Sybil [6] attackers [7], improving the Tor [8] router selection algorithm [9], reducing state in Internet routers [10] and conducting Byzantine leader elections [11]. Network coordinate systems have been shown to be reasonably accurate in estimating network latencies [1, 12].

Vuze [4] is a popular, open-source, BitTorrent [13] client, with over two million concurrent users. It includes an implementation of Vivaldi [1], a popular and widely-cited network coordinate system. Vivaldi is turned on by default and is used in the 'distributed' portion of Vuze. When searching for content in Vuze, a user contacts both central servers and its distributed hash table (DHT) for peer-to-peer downloads. The Vuze DHT is based on the Kademlia DHT [14]. Vuze uses the Vivaldi network coordinates in deciding which peers to contact next when searching for content. A searching peer will first contact peers that are closer to itself, in terms of network distance, to obtain other peers that are closer, in ID space, to the content location. A complete description of Vuze, its DHT and how network coordinates are used in routing decisions is given in Section 2.

Vuze is an example of an application using a network coordinate system to improve its performance. The main **contribution** of this paper is to show how such an application can be attacked. Instead of attacking the actual application, the network coordinate system is targeted so that the eventual outcome of the 'indirect' attack is similar to directly attacking the application. However, the cost of targeting the network coordinate system is much cheaper than targeting the application. More specifically, the Vuze DHT routing uses network coordinates to contact closer (in terms of network coordinate distance) next-hop peers in an attempt to locate the content faster. Vuze allows five parallel but not independent queries for each search. If a malicious peer is contacted on the first hop, it can advertise that it knows of other (malicious) peers which hold the content being searched for. Thus, if a malicious peer is contacted on the first hop, this search is considered to be hijacked as the malicious peer can return other malicious peers close to the target, and the attacker can eventually return bogus results. The bogus results cause a user to waste time performing searches. Moreover, research applications [15, 16] relying on the correctness of the Vuze DHT will suffer from such malfunction. Although attacks [6, 17–20] against distributed hash tables (DHT) [14, 21, 22] to try to hijack searches have been proposed before, this paper shows that the network coordinate system can be exploited such that that searches are hijacked more efficiently. Only 32 nodes are required for our attack to be successful, regardless of the total number of nodes in the Vuze network. Although [20] is effective with a fixed number of malicious peers, it requires 12.5 MB/s of download bandwidth whereas our attack requires only 700 KB/s of download bandwidth, more than one order of magnitude cheaper. An estimate of the cost to run the complete attack is less than $112 per month.

Although it has been shown that existing network coordinate systems are insecure [23–25], all the attacks were on the actual network coordinate systems in an attempt to disrupt the network latency estimations. The proposed attack is on disrupting an application because it is using an insecure network coordinate system.

Our attack on the network coordinate system is to appear closer to all the peers in the network than anybody else. Since a Vuze node picks closer (in terms of network distance, estimated from the network coordinates) peers to send its search queries, if an adversary is very close to every peer, then it will most likely receive a majority of search queries. The goal of the attacker is to be among the first peers contacted during a search query. Once the attacker becomes the first hop, this search is considered to be hijacked. The attacker's plan is thus to (1) lie about its network coordinates such that it appears very close to every peer on the network, (2) remain online so that it becomes the first hop on every search query and (3) return other malicious peers whenever it is queried. A detailed description of our attack is in Section 3.

An attacker, controlling 10% of the peers in the network and trying to directly attack the DHT routing of Vuze, can capture at most 2% of all first-hop queries. We show that an attacker

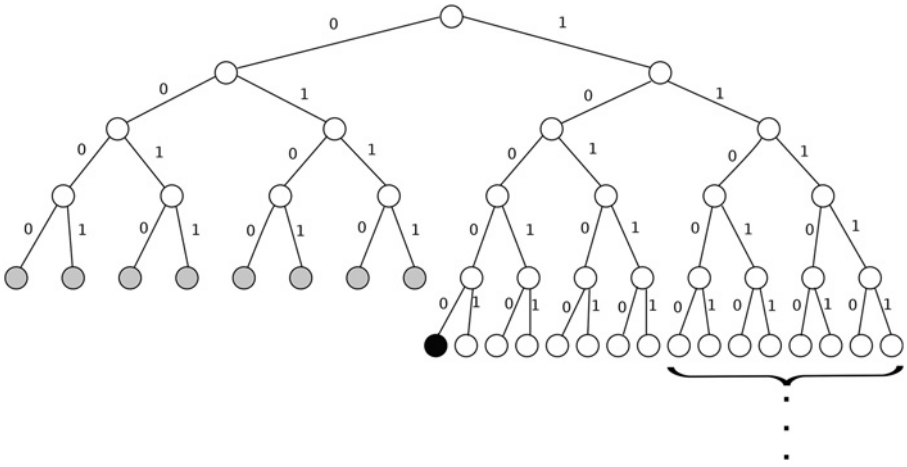

Fig. 1 *Vuze routing table*

Numbers indicate the ID bit match. Grey nodes indicate buckets important to the attack described in Section 3. Black node indicates the bucket where nodes with only first-bit ID match would reside. Right side of routing table keeps expanding if nodes with more prefix bit matches are found

performing a straightforward implementation of our attack can capture almost 20% of queries. However, based on a heuristic analysis, we expect our attack to capture a higher fraction of queries. The difference is attributed to the instability of the network coordinates in Vuze. In Section 4.5, we show how to improve the attack to match the analytical expectations. We expect to obtain over 50% hijack rate if the experiments were run for a longer period of time. Section 4 provides more information about our experimental results.

2 Background

2.1 Network coordinate systems

Every peer in a network coordinate system [1, 2, 26] computes its network coordinates, in such a way that the difference in network coordinates between two peers approximates the actual network latency between these two peers. Vivaldi [1], one of the first decentralised peer-to-peer network coordinate systems, has been implemented in many applications [4, 27]. The implementation of Vivaldi in Vuze [4] is similar to the original Vivaldi paper [1].

2.2 Vuze

Vuze is a popular BitTorrent client with over two million concurrent users, which supports additional functionality, such as *distributed tracking* of torrent files. This is achieved through the use of a *distributed database*, which in turn is an implementation of the Kademlia distributed hash table (DHT) [14]. In the original BitTorrent specifications, in order to start a download, the user needs to obtain a torrent file that contains all the necessary information for the download process. This includes: (i) IP address of one or more trackers, to obtain information about the swarm and other peers (ii) information about files, such as number and size of pieces (iii) info-hash, a hash of the file contents to verify that the download was successful. Distributed tracking, on the other hand, allows Vuze clients to start a download without a torrent file. This is a usability feature as well as a backup option to locate extra peers in case the tracker is unreachable. To participate in distributed tracking, Vuze clients register torrents in the DHT by storing their IP and port under a key. The key is a 160-bit identifier obtained by computing the SHA-1 hash of the torrent info-hash.

2.2.1 Routing table: Vuze uses a modified version of the Kademlia DHT, which differs from the original in several ways, some of which include Vivaldi coordinate extension, caching along paths and encrypted data transfer. The reader is referred to [4] for more details on how the routing table works. Each Vuze client has an 160-bit unique ID, which is simply a SHA-1 hash of the IP:port pair. The Vuze ID is the main identifier for any node and is used to store nodes in the routing table. The routing table is a tree structure that consists of buckets, each bucket corresponds to a length of Vuze ID prefix matches and contains up to 20 nodes (see Fig. 1). 'Closeness' (in terms of Vuze ID) of any two nodes in the network or in the routing table is determined by XOR *distance*. XOR distance is calculated by performing a bitwise exclusive OR operation on two given Vuze IDs, and treating the result as an integer.

A Vuze peer continuously discovers new nodes and updates its state by sending various types of messages to known peers. The Vuze message types we are interested in are: *ping, findNode, and findValue.* The *findNode* and *findValue* message types are used by Vuze to maintain its routing table and for finding content (Vuze ID key) in the DHT. Upon receiving a reply to any of these messages, the node is added to the appropriate bucket in the routing table and marked alive. If the bucket is full, the node is added to the replacement list. The replacement list is another extension to the routing table that allows Vuze to quickly replace dead nodes in the bucket. Each bucket has a separate replacement list containing up to 5 alive nodes. Whenever a node in the bucket fails to respond to a message several consecutive times, it is marked as dead and another node from the replacement list, chosen randomly, is put in its place. Nodes discovered as a result of replies to *findNode* and *findValue* messages are marked as known, but not alive, and a *ping* message will be sent to them in order to establish whether they are alive.

2.2.2 Routing: Lookups in Vuze are parallel and iterative; they work similar to the lookup algorithm in [28]. Here are the exact steps of performing a DHT lookup for target key T in Vuze

1. Choose 20 nodes closest to T in terms of XOR distance in the Vuze ID space from the appropriate bucket in the routing table and add them to the processing list.
2. Sort first 20 nodes in the processing list by XOR distance in the Vuze ID space to key T.
3. Sort first 10 nodes in the processing list by Vivaldi distance in the network coordinate space. This is done to improve the search lookups, as replies are received faster, because of the smaller network distance which suggests a smaller round-trip time.
4. Send queries to the top 5 nodes.
5. When reply is received, add discovered nodes (usually 10 per reply) to processing list, and perform sorting operations as described in steps 2 and 3

IET Inf. Secur., 2015, Vol. 9, Iss. 4, pp. 203–208

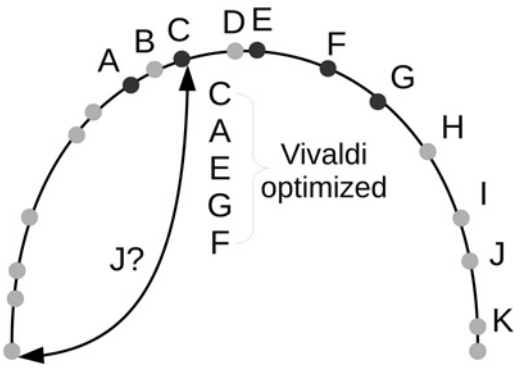

Fig. 2 *Vuze lookup process*

First 10 contacts are sorted by Vivaldi distance in the network coordinate space to the querier

6. If there are fewer than 5 outstanding queries, send another query.
7. Go to step 5, unless target is reached, no new nodes were discovered, or a timeout occurs.

Fig. 2 shows how peers are contacted. Regular Kademlia lookup would contact node G, as it is closer in terms of Vuze ID distance to node J. However, since network coordinates are used, node C is contacted first, as C is closer in terms of network coordinate distance to the querier than node G is.

3 Attack on Vuze

3.1 Overview

Our attack is a practical application of the *Closest node attack* described in [25]. The objective of the attack is to capture the DHT lookups performed by other Vuze clients. With successful distribution of colluding attacker nodes in the Vuze ID space, it is possible to capture a majority of lookups, giving us complete control of the network. To achieve this, we exploit the lookup mechanism in Vuze, in particular the ordering of nodes according to the network coordinate distance when choosing the next hop. In our attack, the attacker claims to be the closest node to the victim in terms of network coordinate distance, which causes the victim to send a query for each search lookup to the attacker. It is important to note that our attack is significantly different from previous attacks on DHTs, such as Sybil attacks [6] or Eclipse attacks [17]. This is because of the fact that we do not directly target the routing layer of the DHT, but rather target the network coordinate system, which indirectly influences the routing of lookups. Although the same outcome is obtained, the means and cost of achieving that outcome is very different.

3.2 Execution

Let us describe a step by step execution of the attack. First, it is important for the attacker to become a member of the victim's routing table. In order to do so, the attacker A sends a *ping* message to the victim V to indicate that A is a node in the network. This achieves two important goals: V adds A to its routing table (or replacement list) and tells A its current network coordinates C_V in the *pingReply* message. We know we were successful in penetrating V's routing table when V sends a *ping* message to the attacker. This happens shortly after V adds A to the routing table in order to verify that A is still alive. If the attacker did not receive such message, he simply needs to continue sending *ping* messages to the victim, since the attacker might have become a member of the replacement list and needs to remind V that he is still alive in the network. This leads us to the second step of the attack: after receiving a *ping* message, A replies with falsified coordinates $C_A = C_V + \delta$, where δ is a small value, piggybacked on top of *pingReply* message to V, making it appear as if A was indeed close to V. This lie allows the attacker node A to become the closest peer to V. Now the attacker must continue to perpetuate the lie. In order to do so, the attacker periodically sends any one of the messages discussed in Section 2.2.1 to obtain an up-to-date knowledge of the victim's network coordinate position. Similarly, when A receives any such messages from V, he must respond with newly forged coordinates $C_A = C_V + \delta$, where C_V is the last known position of V. To hijack the search queries, when the attacker receives a query, it will reply with other malicious peers closer to the target key T. The attacker might not be able to return T as the Vuze ID is a hash of the IP address and port number. The attacker can return bogus results, and other applications [15, 16] relying on the correctness of the Vuze DHT will also suffer.

To summarise, below is what the attacker needs to do in order to hijack search queries for the Vuze network:

1. Send ping messages to each victim to get into their routing table.
2. Once in a victim's routing table, keep sending ping messages to get the latest network coordinate of that victim. This allows the attacker to lie in a more effective manner so that it will always be the closest node to each victim.
3. Remain active and participate in the Vuze network.
4. When receiving a query, return other malicious peers closer to the target key T.

3.3 Analysis

Our attack allows the attacker to capture the lookups in Vuze DHT at a low cost. The execution of the attack does not require a large number of Sybil identities. The only requirement is that attacker IDs are uniformly distributed across the 160-bit ID space, to cover all the buckets in the victim's routing table. This is easy to achieve, since changing the TCP port of the attacker node will result in a new Vuze ID. Our goal is to place the attacker nodes into the 8 buckets marked in Fig. 1, since 50% of the lookups performed are for keys that have no first-bit prefix match with the ID of the searcher. Such placement will allow the attacker to become the first hop for 50% of the lookups, giving us control of a large portion of the network. Note that the other 4 parallel queries have little effect on this attack, since the attacker can always provide more attacker nodes close to target key T in his reply, rendering other queries' responses unappealing. Owing to the way the Vuze search lookup works (see Section 2.2.2), the attacker actually needs to have two entries in each top-level bucket, since each bucket contains 20 Vuze nodes and Vuze sorts the first 10 nodes by Vivaldi distance. For example, for bucket 0000, the attacker needs to create two Vuze IDs 00001… and 00000…. This ensures that the attacker can hijack all the search queries in that bucket. With 8 top-level buckets, the attacker needs to create 16 Vuze IDs. However, it needs to double that number to also hijack the other half of the Vuze network (the victims with Vuze IDs with different first bit: 1…). A total of 32 attacker nodes with carefully crafted Vuze IDs is required to hijack over 50% of the search queries for the whole Vuze network.

4 Experimental results

4.1 Experimental setup

We downloaded the source code from the Vuze [4] website. We only made minor modifications to the code to make sure that our experiments are as close to the regular users' version as possible. Thus, our experiments are applicable to the real Vuze network. The modifications include extra logging and extra code for our attack. All our experiments were performed on PlanetLab [29]. Although all the Vuze clients were connected to the real Vuze network, our malicious nodes only attacked the victim PlanetLab nodes. To simulate search lookups, every 30 minutes, each victim Vuze node performs a search lookup for a randomly generated

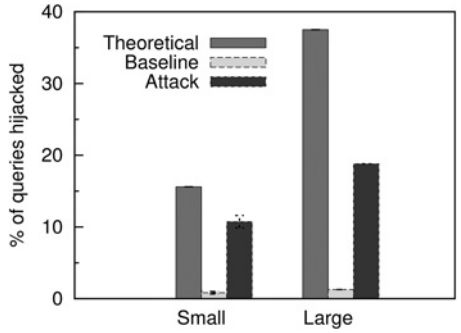

Fig. 3 *Percentage of hijacked search queries for both the large and small experiments obtained by analytical measurement, baseline, and our attack*
Standard deviation is also shown

target key. The hijacking succeeds if one of the five first-hop queries for each key is sent to an attacker node.

4.2 Analytical estimates

The success of our attack depends on the malicious nodes being added to the victims' routing tables. This is not trivial as a victim's routing table, especially the top-level buckets, are usually full. The malicious peer has to be added to the replacement list of each bucket first, and then when an entry in the bucket is considered dead (does not respond to messages), then a random entry in the replacement list is chosen to replace the dead entry. Since this is completely random, the only way to get into victims' routing tables is to keep sending *ping* messages to them and hoping to be randomly added at some point. This takes time, but the attacker can create 32 malicious peers participating in the Vuze network for months and eventually, all 32 nodes will be added to every victim's routing table. As a tradeoff for time, the attacker can also create more malicious nodes at the beginning, to increase the chances of getting added to the routing table.

To determine the expected percentage of hijacked search lookups for a Vuze client V, we need to know the number of attackers added to V's top-level routing table. If the number is, say, 4, meaning that the top-level buckets contain four malicious peers, then the attacker will be able to hijack $(4/16) \times 100 = 25\%$ of top-level bucket queries.

4.3 Experimental baseline

To determine the efficiency of our proposed attack, we ran an experiment on PlanetLab with no manipulation of network coordinates. This would be an attack similar to the Sybil attack [6], where many malicious peers keep sending *ping* messages to the victims in an attempt to be added to their routing table. We consider this as our *baseline*. The only difference between this experiment and our attack experiment is that in the attack experiment, malicious peers lie about their coordinates when responding to requests from the victims.

4.4 Results

We ran both the baseline and attack experiments 3 times on PlanetLab. Each experiment was run for at least five days. We also had two sizes of experiments: a 'large' one which consisted of rougly 1,000 nodes, and a 'small' one consisting of roughly 500 nodes. All the nodes were connected to the real Vuze network but only the PlanetLab nodes were the victims. For each experiment, about 10% of the network was malicious, and the rest of the network were victims.

Fig. 3 shows the results of our experiments. For the small experiment, the baseline was able to capture 0.83% of search queries whereas our attack was able to hijack 10.7% of search queries. This is lower than our theoretical analysis (see Section 4.2 and Section 3.3) of 15.6% of search queries hijacked. Our theoretical analysis is very conservative as it assumes that every top-level bucket is full and the Vuze IDs in each bucket evenly distributed. For the large experiment, the baseline hijacked 1.27% of search queries and our attack hijacked 18.8% of search queries. Again, our attack result is lower than our analytical analysis of 37.5% of captured search queries. Our theoretical analysis depends on the number of attackers that were able to be added to victims' routing tables. On average, 2.5 out of the 160 routing table entries for each victim pointed to a malicious node for the small

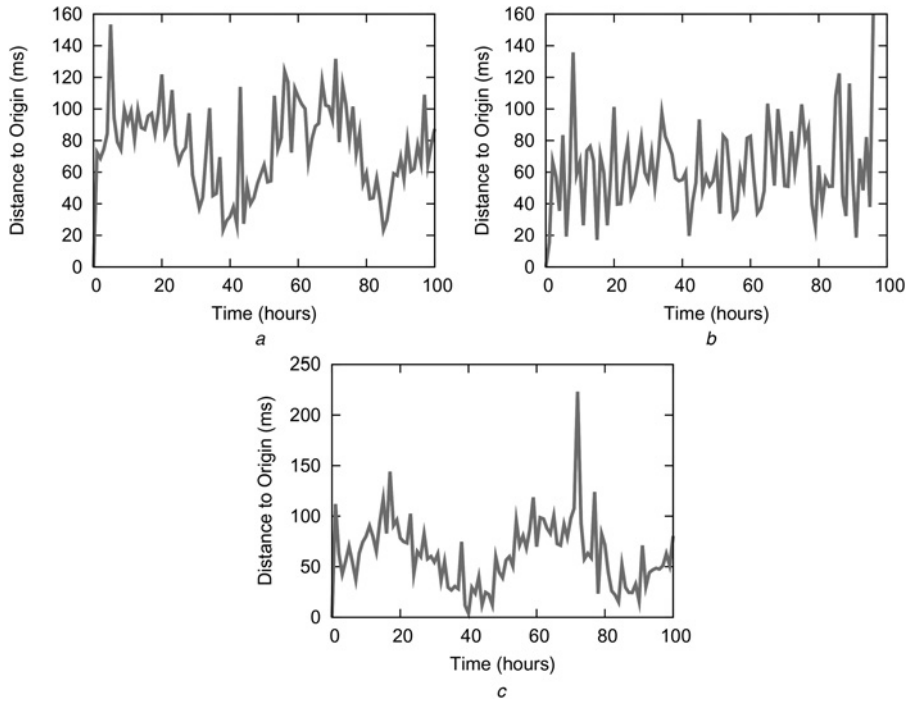

Fig. 4 *Coordinates of three PlanetLab Vuze nodes over time*
Graphs show that the coordinates vary over time, making it harder for the attacker to be the closest node in terms of Vivaldi distance all the time

IET Inf. Secur., 2015, Vol. 9, Iss. 4, pp. 203–208

experiment. This number increased to 6 for the large experiment, which explains the higher percentage of hijacked search queries.

The success of our attack depends on (1) the ability to get into victim's top-level buckets, and (2) lying to victims about the network coordinates such that the attacker is the closest peer to each victim. We reran the small experiment but with 20% of attackers instead of 10%. As expected, the average number of malicious entries in the top-level routing table of victims rose to 4.3, which allows the attacker to hijack 16.5% of search queries, when compared with a theoretical hijack rate of $(4.3/16) \times 100 = 27\%$.

4.5 Better attack results

The straightforward implemention of our attack is less effective than the expected theoretical values. This is because our attacker is not the closest peer to each victim; thus the first-hop search queries are sent to other peers. The reason for this can be depicted in Fig. 4. The figure shows the coordinates of three victim peers over time. Since it is hard to picture a 3-dimensional coordinate with height, the figure shows the distance to the origin for each peer. The figure shows that for all three peers, their coordinates change drastically over time. Thus, the coordinate reported by a malicious node at time t to become the closest node to a victim V, is no longer the best coordinate at time t' ($t' \neq t$). The attacker needs to keep lying about its coordinate. The frequency of possible lies depends on how often the victim contacts the attacker. Since this is beyond our control, there are gaps when the attacker node is no longer the closest peer to the victim. This explains why our attack is less effective than expected.

In an attempt to improve the effectiveness of our attack, we implemented a 'smarter' version of our attack. Instead of the attacker nodes reporting themselves to be the closest node to the victims all the time, they try to position the victim nodes such that they will be the closest nodes for a long period of time. When an attacker node receives a *ping* message from a victim node, it compares the first bit of its Vuze ID to the first bit of the victim's Vuze ID. If the first bits are different, then that means the attacker node is in the top-level buckets of that victim's routing table. Then the attacker replies with a *pingReply* message with its coordinate being close to the victim's coordinate. This is similar to the original attack. However if the first bits are the same, then the attacker reports a coordinate far away from the victim in an attempt to isolate the victim node. In that way, when an attacker node in the victim's top-level routing buckets is contacted next, that attacker node will be the closest node to the victim for a long time. This is a version of the repulsion/isolation attack [24]. The repulsion/isolation attack attempts to isolates each victim node to a network coordinate, far from all other nodes. We ran 3 experiments for three days each; Fig. 5 shows the averaged result. The percentage of hijacked queries is 11.5% for our attack, compared with 10.25% for the theoretical result (average of 1.64 attacker nodes in top-level buckets of victims). This variant of the original attack is thus more effective, as the success rate exceeds the expected success rate. We note that our expected success rate is very conservative as it assumes the top-level routing table is full and all the Vuze IDs are evenly distributed.

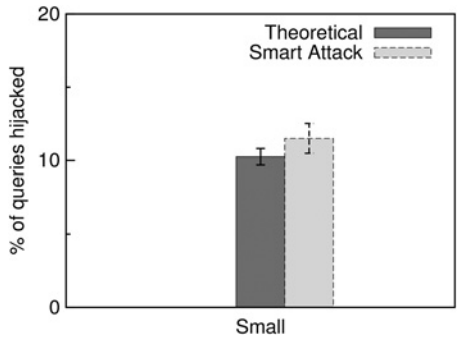

Fig. 5 *Percentage of hijacked search queries obtained by analytical measurement and our 'smart' attack*

Standard deviation is also shown

4.6 Cost of attack

We emphasise that our attack is very efficient. Each attacker only needs to send one *ping* message every 2 hours to try to get into each victim's routing table. Each ping request is 42 bytes and each ping reply is 80 bytes. With a two million nodes Vuze network, that is about 12 KB per second uplink and about 22 KB per second downlink. Since we need 32 attackers, that translates to 371 KB per second of uploaded bandwidth and 711 KB/s of download bandwidth. This is much cheaper than the 100 Mb/s (12.5 MB/s) required for [20]. The cost to maintain the routing table is also fairly cheap. Each Vuze attacker instance uses around 55 MB of memory and <3% of one core of a quad-core 2.67 Ghz Intel Xeon W3550 processor. If our attacker were to use Amazon EC2 [30] to conduct the attack, this would cost less than $112 per month. Despite its low cost, our attack has serious implications to those using Vuze for file sharing. As we mentioned above, Vuze uses DHT lookups to locate a suitable torrent file to start a download. However, if a user is a victim of our attack, his or her lookups can be redirected to another malicious node, which will serve a bogus torrent file causing the user to perform a futile download. In addition, any research applications [15, 16] relying on correctness of the Vuze DHT might suffer from such malfunction.

The attack described is conservative; the attacker can perform additional work to increase the success rate of the attack, such as running more attacker nodes.

5 Conclusion

Vuze is a popular BitTorrent file-sharing client. It uses a network coordinate system to improve the efficiency of its search lookups. However, the network coordinate system implemented is insecure and can be easily attacked. By becoming the closest peer, in terms of network coordinate distance, to every victim node in the Vuze network, an attacker can theoretically hijack every single search query. Experimentally, we were able to hijack almost 20% of all search queries. The reason for our lower success rate, when compared with the theoretical hijack rate, is because of the instability of the victims' network coordinates. This difference is fixed by using a 'smarter' version of our attack. We expect over 50% hijack rate, given enough time for the malicious nodes to be added to the victims' routing tables. Our attack is directly applicable to the real Vuze network and can be deployed any time. Moreover, our attack is very cheap to launch, costing less than $112 per month.

To mitigate our attack, Vuze should not be using an insecure network coordinate system in an attempt to improve its performance, as the network coordinate system actually makes Vuze more vulnerable to easier and cheaper types of attacks. However, network coordinate systems are useful and can decrease the overall latency of the distributed system. Alternatively, Vuze could implement a secure network coordinate system, such as [31, 32]. Vuze could also modify its lookup algorithm so that one of the five parallel queries does not use Vivaldi distance. Although a fix is not hard to implement, the goal of this paper is to show that a cross-layer attack that is more efficient than previously known attacks is possible in Vuze.

6 References

1 Dabek, F., Cox, R., Kaashoek, F., Morris, R.: 'Vivaldi: a decentralized network coordinate system'. SIGCOMM '04: Proc. of the 2004 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, New York, NY, USA, 2004, pp. 15–26

2 Eugene Ng, T.S., Zhang, H.: 'Predicting internet network distance with coordinates-based approaches'. IEEE INFOCOM, 2001, pp. 170–179
3 Agarwal, S., Lorch, J.R.: 'Matchmaking for online games and other latency-sensitive P2P systems'. SIGCOMM 'mt09: Proc. of the ACM SIGCOMM 2009 Conf. on Data Communication, ACM, New York, NY, USA, 2009, pp. 315–326
4 Vuze. http://azureus.sourceforge.net
5 Choffnes, D.R., Bustamante, F.E.: 'Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems', *SIGCOMM Comput. Commun. Rev.*, 2008, **38**, (4), pp. 363–374
6 Douceur, J.R.: 'The sybil attack'. IPTPS 'mt01: Revised Papers from the First Int. Workshop on Peer-to-Peer Systems, Springer-Verlag, London, UK, 2002, pp. 251–260
7 Bazzi, R.A., Konjevod, G.: 'On the establishment of distinct identities in overlay networks'. PODC'05: Proc. of the 24th Annual ACM Symp. on Principles of Distributed Computing, ACM, New York, NY, USA, 2005, pp. 312–320
8 Tor. http://www.torproject.org
9 Sherr, M., Blaze, M., Loo, B.T.: 'Scalable link-based relay selection for anonymous routing'. PETS 'mt09: Proc. of the Ninth Int. Symp. on Privacy Enhancing Technologies, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 73–93
10 Abraham, I., Malkhi, D.: 'Compact routing on euclidian metrics'. PODC '04: Proc. of the 23rd Annual ACM Symp. on Principles of Distributed Computing, ACM, New York, NY, USA, 2004, pp. 141–149
11 Cowling, J., Ports, D., Liskov, B., Popa, R.A., Gaikwad, A.: 'Census: location-aware membership management for large-scale distributed systems'. Proc. of USENIX Annual Technical Conf., 2009
12 Ledlie, J., Gardner, P., Seltzer, M.: 'Network coordinates in the wild'. Proc. of USENIX Symp. on Networked Systems Design and Implementation (NSDI)07, 2007
13 BitTorrent. http://bittorrent.com
14 Maymounkov, P., Maźieres, D.: 'Kademlia: A peer-to-peer information system based on the xor metric'. IPTPS, 2001
15 Geambasu, R., Kohno, T., Levy, A., Levy, H.M.: 'Vanish: Increasing data privacy with self-destructing data'. Proc. of the 18th USENIX Security Symp., 2009
16 Geambasu, R., Levy, A., Kohno, T., Krishnamurthy, A., Levy, H.M.: 'Comet: An active distributed key/value store'. Proc. of OSDI, 2010
17 Singh, A., Castro, M., Druschel, P., Rowstron, A.: 'Defending against eclipse attacks on overlay networks'. EW11, 2004
18 Sit, E., Morris, R.: 'Security considerations for peer-to-peer distributed hash tables'. IPTPS, 2002
19 Castro, M., Druschel, P., Ganesh, A., Rowstron, A., Wallach, D.S.: 'Secure routing for structured peer-to-peer overlay networks'. OSDI, 2002
20 Wang, P., Tyra, J., Chan-Tin, E., *et al.*: 'Attacking the kad network'. Proc. of the Fourth Int. Conf. on Security and Privacy in Communication Netowrks, SecureComm '08, ACM, New York, NY, USA, 2008, pp. 23:1–23:10
21 Stoica, I., Morris, R., Liben-Nowell, D., *et al.*: 'Chord: A scalable peer-to-peer lookup service for internet applications'. ACM Special Interest Group on Data Communication (SIGCOMM), 2001
22 Rowstron, A.I.T., Druschel, P.: 'Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems'. Middleware'mt01: Proc. of the IFIP/ACM Int. Conf. on Distributed Systems Platforms Heidelberg, Springer-Verlag, London, UK, 2001, pp. 329–350
23 Kaafar, M.A., Mathy, L., Turletti, T., Dabbous, W.: 'Real attacks on virtual networks: Vivaldi out of tune'. LSAD 'mt06: Proc. of the 2006 SIGCOMM Workshop on Large-scale Attack Defense, ACM, New York, NY, USA, 2006, pp. 139–146
24 Zage, D.J., Nita-Rotaru, C.: 'On the accuracy of decentralized virtual coordinate systems in adversarial networks'. CCS'mt07: Proc. of the 14th ACM Conf. on Computer and Communications Security, ACM, New York, NY, USA, 2007, pp. 214–224
25 Chan-Tin, E., Feldman, D., Kim, Y., Hopper, N.: 'The frog-boiling attack: limitations of anomaly detection for secure network coordinates'. SecureComm, 2009
26 Eugene Ng, T.S., Zhang, H.: 'A network positioning system for the internet'. ATEC'mt04: Proc. of the Annual Conf. on USENIX Annual Technical Conf., USENIX Association, Berkeley, CA, USA, 2004, p. 11
27 Ledlie, J., Pietzuch, P., Seltzer, M.: 'Stable and accurate network coordinates'. ICDCS'mt06: Proc. of the 26th IEEE Int. Conf. on Distributed Computing Systems, IEEE Computer Society, Washington, DC, USA, 2006, p. 74
28 Kaune, S., Lauinger, T., Kovacevic, A., Pussep, K.: 'Embracing the peer next door: Proximity in kademlia'. Proc. of the 2008 Eighth Int. Conf. on Peer-to-Peer Computing, P2P 'mt08, IEEE Computer Society, 2008, pp. 343–350
29 PlanetLab. http://planet-lab.org
30 Amazon EC2. http://aws.amazon.com/
31 Chan-Tin, E., Hopper, N.: 'Accurate and provably secure latency estimation with treeple'. Network and Distributed System Security (NDSS) Symp., 2011
32 Chan-Tin, E., Hopper, N.: 'KoNKS: Konsensus-style network koordinate system' (ACM ASIACCS, 2012)