THE INDEX POISONING ATTACK in P2P FILE SHARING SYSTEMS

ABASTRACT

In P2P file-sharing users search index to find locations of desired files. ***Index poisoning***, for particular file, attacks index with bogus that may include bogus IP addresses, bogus port number, bogus file identifiers.
Remember there are different types of networks p2p, structured and unstructured: index poisoning attack both without any distinction. In our analysis we consider unstructured file-sharing system as FastTrack and DHT based file sharing system.
This scientific paper is aimed at those who are already familiar with the terminology and P2P technology. In any case we recall concepts and structures fundamental.

Keywords: peer-to-peer, overlay network, distributed hash table, poisoning attack, pollution and poisoning level.

I. INTRODUCTION.

Normally, the main difference between the concepts of P2P network and traditional server-client network is that the file downloading is not provided by central server. P2P technology, in these years, is used not only for file-sharing as the first time, but also for instance message communication and distributed computer. For example think about Skype o IP-TV.
P2P network can be divided into four categories:

- Centralized P2P (Napster)
- Pure P2P (Gnutella 0.4, Freenet)
- Hybrid P2P (Gnutella 0.6, JXTA)
- DHT (Distributed Hash Table) (BitTorrent)

As “traditional” Internet, P2P networks can be attacked in several ways, such as Dos attacks, DDOS attacks, poisoning attacks, pollution attacks. Our attention is focalized about index poisoning attacks.
Sometimes someone can confuse poisoning with pollution attack.
In Pollution attack the attackers corrupts the targeted content, rendering it unusable. The polluted files is equal to unpolluted files and one user can not distinguish them. So the user download infect file into his file-sharing folders, from which then later download polluted files contributing to its spread.
Index poisoning attacks is done by inserting massive numbers of bogus records into the index. When a users attempt download a file with a false “coordinates” the system of course fails his search. The file-sharing system typically continues to search for the non existent file. After few attempts the search is abandoned. The “advantage” to use poisoning attack is that requires less bandwidth and not a lot of resource. Thousand of bogus records per title, all with different identifiers, can be deposited into the index from single attack node.
In this paper we investigate poisoning attack in two file sharing systems, Overnet and Fasttrack. We develop a novel and efficient methodology for estimating index poisoning levels and index pollution level. We note, FastTrack and Overnet, are vulnerable about this attacks. In the end of the article we suggest some solutions against pollution und poisoning.

II. Background Session

In this section we remember the structure of DHT and FASTTRACK. So we can understand better in which way pollution and poisoning attack these network. P2P systems can be classified to unstructured P2P and structured P2P. In unstructured P2P system, files can be stored in any peer, that is, the file storage has no certain structure. In contrast, structured P2P system maintains a link between file contents and IP addresses of the peers using Distributed Hash Table (DHT) [5]. Therefore, the whole P2P system holds a certain structure. Unstructured P2P system includes Centralized P2P, pure P2P and Hybrid P2P. Normally, structured P2P means DHT-base P2P.

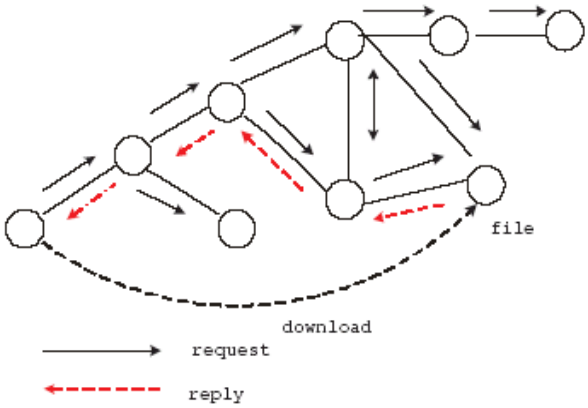


Figure 2: Flooding query of pure P2P

A node requests the desired content by broadcasting the queries to all its neighbours. Then its neighbours continue flooding the requests to the further nodes until the Time to Live (TTL) is exceed. When the peer which holds the needed file gets the request, it responses the query then a download session will be established between initial peer and destination peer. The drawback of pure P2P is that it generate a huge amount of signaling traffic by flooding. And also the routing mechanism is based on best effort, therefore, a query node may not get the reply even the destination is available in the network.

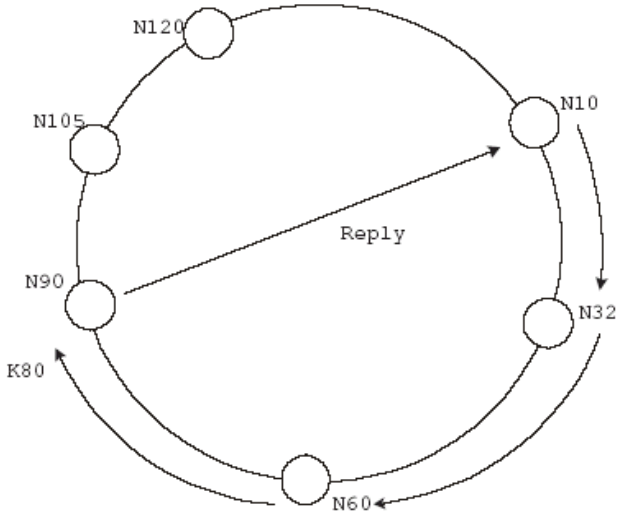
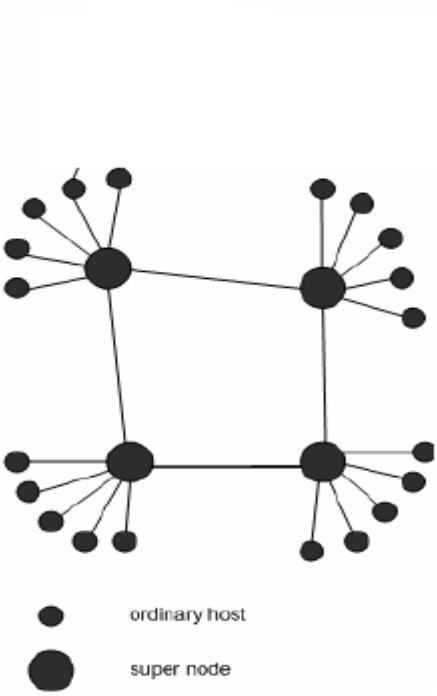


Figure 3: DHT circular address space and basic look up

DHT-based P2P is a structured P2P system. In DHT-based network, nodes are organized in a structured overlay. The Figure 3 illustrates a linear circular address space and the basic look up of Chord. In the picture, each DHT node maintains a range of address space and the address spaces can be consider as a circle. Each file in DHT system is assigned an unique ID in a certain address space. The ID consists of two parts: key and value. Key is the hash value of file name, value is the hash value of file content. Each node in DHT manages a range of key values that start from itself to the successor node that contains the smaller key value. For example, the node 10 manages the key values in the interval 10 to 31.

FastTrack is decentralized and unstructured. FastTrack has two classes of nodes, Ordinary Nodes (ONs) and SuperNodes (SNs). SNs have more responsibility and are typically more powerful than ONs respect to availability, Internet connection bandwidth and processing power.



- **Index poisoning in FastTrack**

When an ON launches the FastTrack application, the ON establishes a TCP connection with a SN, thereby becoming a child of that SN.
Importantly, when a ON disconnects, its SN removes it from its index.
For FastTrack, the index poisoning attack is to insert bogus records into the indexes of the SNs.
For FastTrack, the index poisoning attack requires ongoing TCP connections to each targeted SN. FastTrack has roughly 20,000 super-nodes in operation [3]. The more SNs that an attacker index poisons, the more successful the attack.

- **Index poisoning in Overnet (DHT)**

The attacker _rst determines keywords from the title and hashes the keywords. The attacker then poisons with one of two approaches:

1) The attacker generates a random identi_er, not derived from the hash of some _le. The attacker publishes <key, value> pairs, where key is the hash of a keyword and value is the random identifier. When a user searches for the keyword, the user's client receives the bogus identifier. If the user selects the bogus identi_er, the DHT displays .looking. and searches indefinitely for the identifier.
2) The attacker first publishes <key, value> pairs, where the key is a keyword hash and value is a version identifier. The attacker then sends a second publish message for <key, location> where the key is the same version identifier inserted in the previous step, but the location is bogus (for example, non-present IP address). Overnet uses UDP for messaging, including for the publish messages, the attacking node does not need to maintain any TCP connections to index pollute a particular title. DHTs, in general, are highly vulnerable to the index poisoning attack for targeted titles, unless appropriate counter measures are taken.

III. Methodology

The methodology that I present in this paper develop itself through three steps:

1) Over the measurement period, track the advertised copies. For each advertised copy, we record the version identifier along with the node that published the message. We refer to this step as **harvesting**. From the harvested data, create a list of the advertised versions, and for each advertised version a list of the distinct advertised copies. This step needs to be customized for each file-sharing system.
2) Determine from the harvested data which advertised versions are poisoned, which are polluted, and which are clean. This step is generic to many P2P file sharing systems and involves a crucial sub-step in determining the sources that are responsible for poisoning and polluting the system.
3) Determine the poison and pollution levels, for both versions and copies using the equation:

Let V be the set of all versions advertised over this period for the title T. Each advertised version is either a poisoned version (that is, non-existent), a polluted version (existent but intentionally corrupted) or a clean version. Denote by V1, V2 and V3 for the sets of poisoned, polluted, and clean advertised versions. We denote the **version poison level** to be the fraction of poisoned versions advertised during the investigation interval, that is, $L_1^V = |\mathcal{V}_1|/|\mathcal{V}|$.
Similarly, the **version pollution level** and **version clean level** are define to:

$$L_2^V = |\mathcal{V}_2|/|\mathcal{V}| \text{ and } L_3^V = |\mathcal{V}_3|/|\mathcal{V}|,$$

We now define the copy poison level and copy pollution level, which we more simply refer to as the **poison level** and **pollution level**. For each version $v \in \mathcal{V}$, let $\mathcal{C}_v$ be the set of all distinct copies advertised during the investigation interval for the version v. Then the poison level L1, pollution level L2, and clean level L3 are defined as:

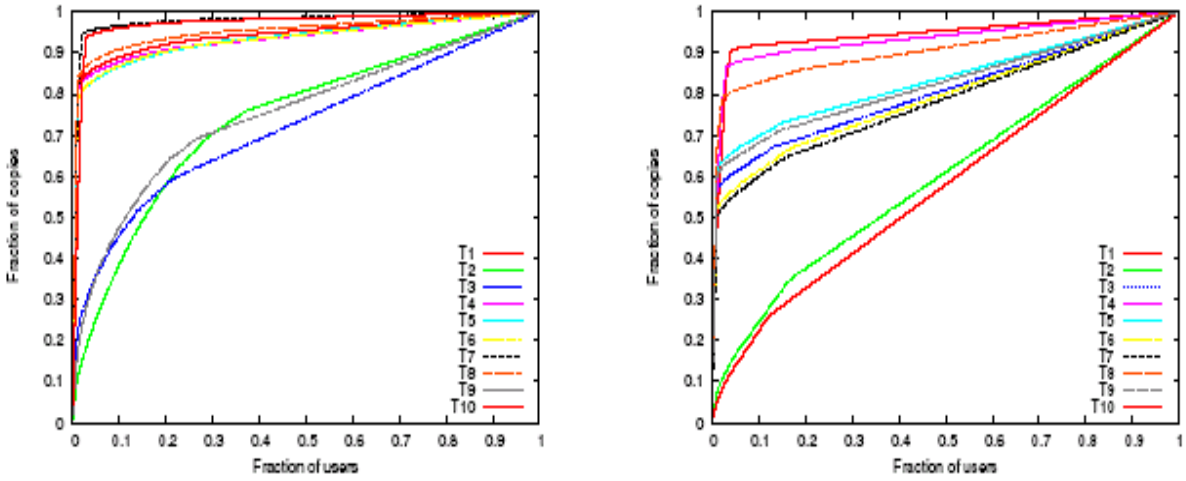
$$L_i = \frac{\sum_{v \in \mathcal{V}_i} |\mathcal{C}_v|}{\sum_{v \in \mathcal{V}} |\mathcal{C}_v|}, \qquad i = 1, 2, 3$$

- Harvesting in FastTrack:

We distinguished between distinct copy advertisement by using the tuple (IP Address, Service port number). For the investigated titles, we selected 10 songs from the iTunes top-100 list in April of 2005 [6]. We harvested the copy information for these 10 titles for a 1-hour period in April 2005.

- Harvesting in Overnet

In Overnet we harvest by inserting our own Overnet nodes into the DHT. For each title, we extract one of the keywords from the title's name and hash it. For each hashed keyword, we insert our own Overnet node with node ID configured to be the 128-bit value of the hashed keyword. In this manner, users that advertise a copy for the title will send a publish message to our inserted node. The inserted node thus collects advertisements for the title over the measurement period. Each advertisement is encapsulated in a UDP datagram and includes the the name of the title, the identifier (content hash) for the version, as well as other information such as duration, size, etc. The UDP datagram header provides the source IP address and the source (messaging) port number.
For the investigated titles, we selected 10 songs from the iTunes top-100 list in June of 2005 [6]. These selected titles are distinct from those selected for the FastTrack experiments because of the dynamic nature of popular music charts. We harvested the copy information for these 10 titles for a 1.5- hour period in June 2005. We chose the time period to reflect the rate that Overnet clients use to refresh their shared files.



(a) FastTrack (b) Overnet

Fig. 3

BASIC IDEA

The basic idea of our paper is:

the large majority of users advertise at most a few versions of the title and a relatively small number of users advertise a large number of versions of the title.

As we can see in Fig.3, for FastTrack and for Overnet too, the result of our measurement support this idea. For FastTrack, we see that many curves quickly rise, with 5% of the users advertise more than 85% of the copies. Similarly, for Overnet, for many of the titles, the large majority of the users advertise few versions whereas a small minority advertise a large number of versions.
Dividing the users in heavy users und light users (*Note* for Wolfgang*: I don´t know if all the formula are necessary*) we indicate set of heavy users as H and set of light users as L . The users in H advertise a relatively large numbers of copies for at least one of investigated titles, we naturally suspect them being the decoyers.
We make another divisions:

- V_x is the set of versions that have been advertised by both the heavy and light users;
- V^*_h is the set of versions that have been advertised by heavy users but not by the light users;
- V^*_l is the set of versions that have been advertised by light users but not by heavy users.

- The sets V_x, V^*_h, , V^*_l form a partition of the set all advertised versions V.

Our idea is that the vast majority of the advertised version in V^*_h are poisoned versions, the vast majority of the advertised in V_x are polluted versions, and the vast majority of the advertised versions in V^*_l are clean versions. I can summarize the situation as:

Hypothesis : Users in H are decoyers and the users in L are regular users.
Thesis : If a version has been advertised by regular user but not by decoyer the version is clean;
If a version has been advertised by decoyer but not by regular the version is poisoned;
If a version has been advertised by both (regular and decoyers) the version is polluted.

As we can see this demonstration have not the mathematic scientific rigour.
It is heuristic and to evaluate we develop the following steps:

1. Query for a title and obtain advertise v
2. For Overnet, we query another time for v. If replay not come back , we can say the version is poisoned.
3. For FastTrack, for our search we obtained one location (IP address and port Number .
   We can try to execute one TCP connection to download the file; if attempt fails and come back a TCP RST, we can say the version is poisoned. If there isn´t problems we download a copy of v.
4. We have now a copy of v and see if it is polluted. Take note.
5. At which set (V_x, V^*_h, , V^*_l) the version belongs to.

(NOTE for Wolfgang: I omitted all the table and results for methodology..
For me it is important the idea. Later number and measurement are only calculus).

IV. Defense against Poisoning.

As we have seen poisoning is a nasty problem for P2P file sharing.
I present in this section two different defense against poisoning: Tarzan and Rating Sources.

- **Tarzan**

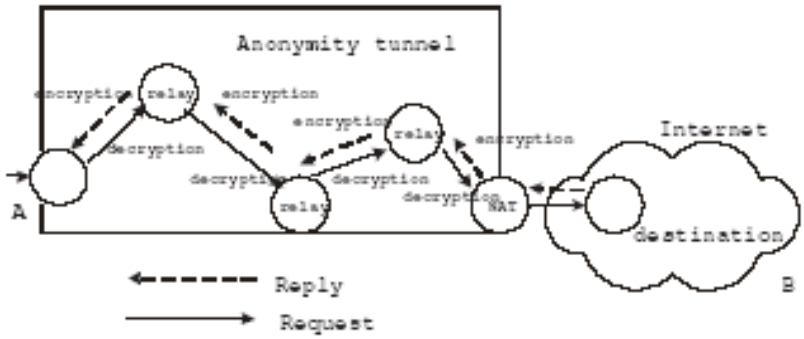


Figure 4: Network overview of Tarzan

The main idea of the anonymous overlay network is to protect the network by concealing the network addresses of the participants from the malicious nodes.
Figure 4 shows the network structure and working principle of Tarzan. Node A is a Tarzan node running a client application [10], in another words, it is the node requires anonymity. It selects a group of available peers to compose the transport path. All these peers called "relay" [10] are selected randomly for security reason. After the nodes selection, the symmetric keys of each relay are distributed. Then, node A encrypts packets several times using the symmetric keys of the relays. All these nodes compose a forwarding path that is considered as a tunnel. Thus, the encrypted packets start from node A and go through the relays. Each relay decrypts one layer of encryption and forwards the packets to the next hop, until the packets reach the last relay which runs a network address translator (NAT). By this way, each relay can only know its previous and succeed nodes but have no idea about the source node. The last node decrypts the last layer encryption and obtains the destination address of the packets. Finally, the packets are sent to the public destination address by the NAT node. The reply path is a reverse tunnel of sending path. The final node encrypts the message using its symmetric key and forwards it to the penultimate relay. Then the relays encrypt and send the message one by one until it gets to Node A. Node A decrypts the layers of encryption and gets the message. By using Tarzan, the anonymity of the nodes is well protected. However, a notable disadvantage´of Tarzan is that it utilize best effort routing principle.

- **Rating Sources and list of reputation.**

The idea is, as we have seen in the previous section, that the Decoyers advertise a large number of version of the same title. Decoyers typically controls narrows blocks of IP addresses and can easily move their nodes from one IP address to another within the block. We assign reputations to narrow ranges of IP addresses. For the IP range, we use /n subnets.

In this scheme, each supernode creates a local reputation for each /24 prefix that is advertising directly to the supernode. The reputation is a function of the number of copies per title being advertised by the prefix; if a prefix advertises a large number of copies of any one title, it will likely have a low reputation. (in this case we use /n with n = 24).
Each SN would also maintains its own global reputation list, which includes all IP prefixes advertised to all the SNs. SNs would exchange with each other their global reputation lists. When a SN receives global lists from other SNs, it combines the received lists with its own local list to create a new global reputation list at that node. Each SN periodically refreshes its local reputation lists, recalculates its global view, and exchanges its global view with other SNs. The lower the reputation, the more likely the prefix is a decoyer.
Each SN could periodically send its global reputation list to its children nodes, and the children nodes could use a peer dependent threshold for blacklisting peers with reputations lower than the threshold.
One problem with such a scheme is that some of the SNs actually may be operated by attackers. But if the number of compromised SNs is small, then a reputation scheme should give meaningful results.

V. Conclusion

P2P file Sharing (Emule , Bit Torrent) and all the services (IP TV, Skype) that use P2P network architectures in the last years strong increased. When someone wants project a P2P must considered poisoning and polluted attack. It is hard understand where and who have generated the attack.
There is one important aspect that I want remember and remark: in a P2P network the infected file is sharing with others peer and there is a propagation. With a single infected index attackers can, for example, crash one peer. Infected one files and forwarding all the traffic against one node: in few second a lot of peer try to establish a TCP connection. This peer crash.
Another example is that the attackers use false IP addresses: a peer search a file but without success. After a few second try again. This steps for a lot of files and a lot a nodes cause a congestion and useless traffic.
We have seen only two methods against poisoning but Tarzan, rating sources and list of reputation, but these methods, with the help of better network design, can cover these problems.