Solutions to Swamp Poisoning Attacks in BitTorrent Networks

K.Y. Wong, K. H. Yeung, and Y. M. Choi

***Abstract* – Swamp poisoning in BitTorrent corrupts files sharing between peers. The worst case causes the swamp unusable as the protocol does not provide sufficient data integrity checking. This paper proposes two solutions in order to resolve this attack.**

***Index Terms:* Peer-to-peer systems, BT networks, distributed file sharing, file sharing attacks**

I. Introduction

In peer-to-peer (P2P) systems, peers are contributing themselves to benefit others [1][2]. Unlike the traditional client-server systems, P2P systems rely on the sharing of the available bandwidth to upload resources, and download resources from other peers.. Thus, throughput between peers can be increased substantially.

BitTorrent is one of the most popular implementation of P2P file sharing networks. It combines the benefits of the client server and P2P mechanism by adding a tracker server into each sharing network [1]. Fig.1. shows one of the roles of tracker server is to identify and forward incoming the peers into specific sharing network, called swamp. The other feature of tracker server is to help accomplish fairness of sharing among peers [3].

During the file sharing process, the BitTorrent protocol can be exploited by malicious users to share some polluted pieces of a file in the swamp, making others fail to download the correct file. It may even end up in the termination of the whole file transfer process. This paper mainly discusses how swamp poisoning attacks the BitTorrent sharing networks and proposes some possible solutions for it.

This paper points out how swamp-poisoning attack corrupts the BitTorrent sharing network and purposes some possible solutions about it.

Manuscript received November 26, 2008. The work described in this paper is supported by Macao Science and Technology Development Fund (Project No. 099/2005/A).

K. Y. Wong is with the Computer Studies Program, Macao Polytechnic Institute (phone: +853-85996440; fax: +853-28719227; e-mail: kywong@ipm.edu.mo).

K. H. Yeung is with the Department of Electronic Engineering, City University of Hong Kong (e-mail: eeayeung@cityu.edu.hk).

Y. M. Choi is with the Computer Studies Program, Macao Polytechnic Institute (e-mail: choiyiuman@gmail.com).

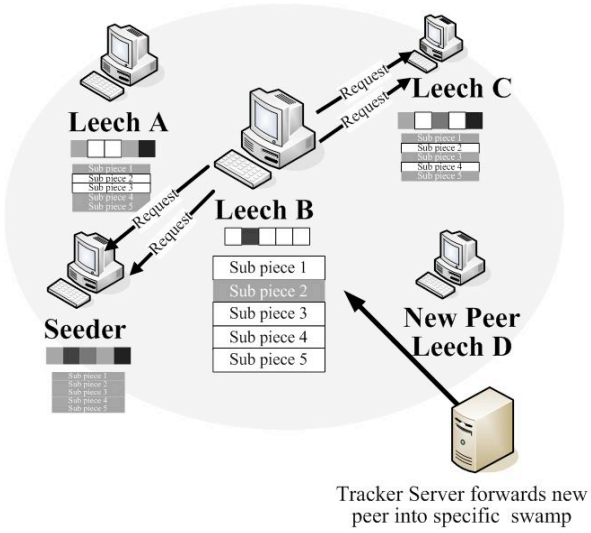

Fig.1. Diagram to illustrate the operation of BitTorrent.

II. Swamp Poisoning Attack

Apart from free-riding, swamp poisoning brings much higher impact to the peer-to-peer network. This kind of attack can be achieved mainly because of the weakness of p2p protocol [3]. On the one hand, based on conventional point-to-point data transferring schema, whole of the file be sent from one end to another end. It is hard for malicious users to manipulate attack such as man-in-the-middle. On the other hand, in P2P sharing swamp, files are divided into numerous of sub-pieces, possible attack can be achieved on sub pieces level.

As peer can stay within or rejoin the swamp after successfully download the whole piece of file [3], mainly design for increase other peers' download bandwidth, this feature provides a reasonable channel for malicious users to resident the sharing swamp with the corrupted sub pieces.

The worst case is some of the BitTorrent application only checks the file size of the file when the peer as a seeder role [3], Fig.2. shows that the torrent file can be opened in editor in order to know the actual size and name of the source file, which provides spaces for malicious users to manipulate the attack.

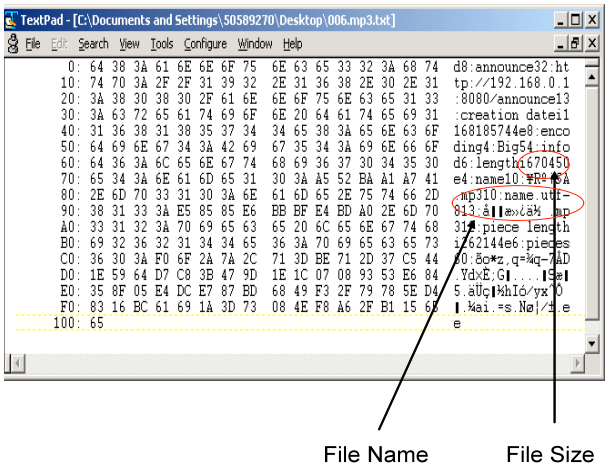

Fig.2. Possible information for malicious users to launch the attack.

Swamp poisoning is easy to be achieved as leech never check any byte of the received sub pieces until all of them have been received, If any sub piece extract cannot match with the SHA-1 hash code, generated by the original source file, the entire file then be discarded and retry for the whole file. As a result the swamp will be corrupted by and spread to other leeches like plague.

As leeches are seeking for their missing pieces from peers, they intend to find a peer with more available pieces to download [4]. This phenomenon increases the chances of malicious users poisoning the swamp as it increases the chance of peers request malicious users to deliver corrupted pieces to them.

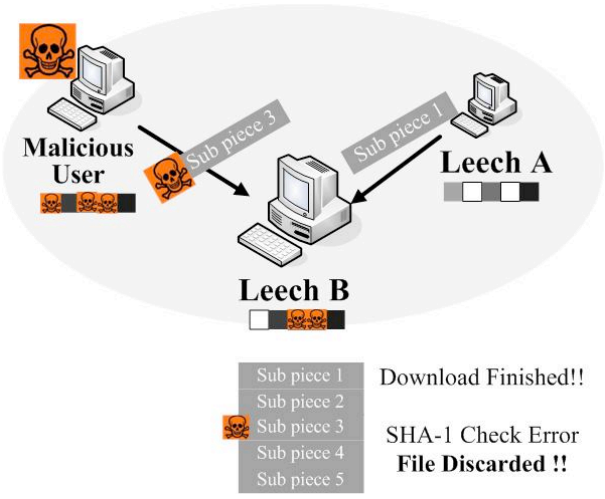

Fig.3. An example of Swamp Poisoning Attack.

Currently numerous of BitTorrent applications recheck the file before seeder do the sharing, however tracker server allow different applications at different version to join the swamp in order to fulfill the design of high compatibility.

Some users suggested blacklisting some of the outdated versions by the tracker server, yet version checking mechanism can easily be pass over by spoofing technique. The solutions going to be discussed are based on the sub piece level checking mechanism.

III. Peer-Selection-Based Solution

The basic concept of peer-selection-based solution is to divide the whole swamp into minor swamps if encountered any error sub piece, coordinated by tracker server.

Whenever any sub piece dropped, the leech updates the tracker server about the incident. The tracker randomly hides half the existing peers; as a result the leech can only requested sub-pieces from the selected peers group. If the leech encounters poisoning again, the tracker server then hide the existing the minor swamp and send the other hidden minor swamp to the leech.

Once all available minor swamps are tried, still the poisoning encountered. Then the tracker server divides each the minor swamp by half, and leaves one-forth of peers for the peer to do the sharing. If the minor swamp does not have sufficient sharing pieces to reform the original file, leech can request the tracker server for another minor swamp.

This mechanism ensures leeches having a higher successful rate of downloading files. It can also narrow down the possible suspect malicious user(s). The location of malicious user can be revealed as most peers are different from the infect swamps.

A. Example of Applying Peer-Selection-Based Solution

Suppose the swamp has 9 peers, as shown in Fig. 4, since the leech B encountered swamp poisoning, the tracker server divides the swamp into 2, minor swamp 1 and minor swamp 2, and each minor swamp contains of 4 peers [A, C, I, H]1 & [D, E, G, F]2.

Let say the minor swamp 1 has been hidden by tracker server thus leech B only request sub pieces from peer D, E, G & F. As all peers within minor swamp 2 does not have the sub piece 1, then B can request tracker server to send the other minor swamp 1 to it.

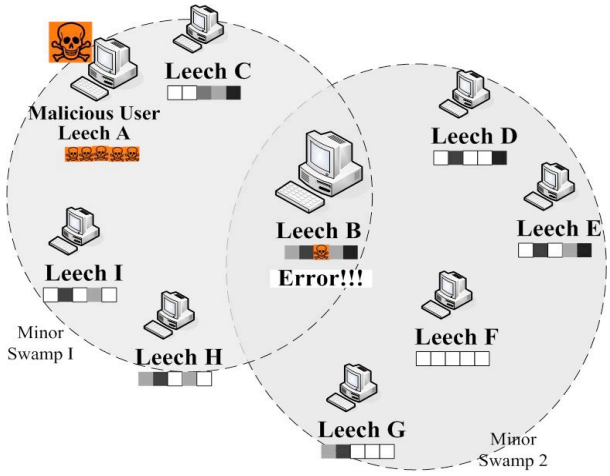

Fig.4. Illustration of peer-selection-based solution.

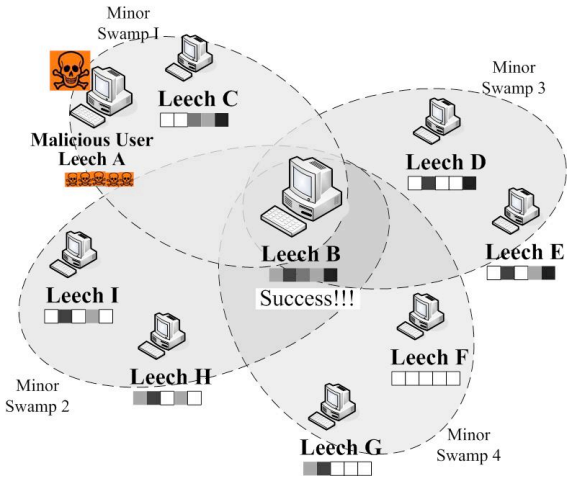

Fig.5. Illustration of peer-selection-based solution.

As shown in Fig. 5, the tracker server then divides the minor swamps into 4, [A, C]1, [I, H]2, [D, E]3 & [G, F]4, and try the sharing minor swamp by minor swamp. By this time B gets the sub piece 1 from peer C and finishes the downloading process.

B. Overhead Calculation of the Solution

Table 1 shows the prediction about the overheads implementing the Peer-Selection- Based solution. It is the prediction about the overhead if this solution being implemented at the existing BitTorrent environment. It obviously shows that the overhead mainly depends on the number of available malicious users within the swamp as number of trials is proportion to number of malicious users.

Table 1. Overhead calculation on Peer-selection-based solution

Parameters (No. of malicious users, No. of normal peers)	**Traffic Overhead** (Max. no. of round trips, Min. no. of round trip)	**Bandwidth Overhead** Network bandwidth used (Kbytes) assuming each piece = 256 kb
(1,7)	(2,1)	Max=256*2=512 Min=256
(4,4)	(8,1)	Max=256*8 = 2048 Min = 256kb
(7,1)	(14,8)	Max = 256*14 = 3584kb Min = 256*8 = 2048kb
(1,3)	(2,1)	Max = 256*2 = 512kb Min = 256kb
(2,2)	(4,1)	Max = 256*4 = 1024kb Min = 256kb
(3,1)	(6,3)	Max = 256*6 = 1536kb Min = 256*3 = 768kb

IV. SUB-PIECES-VERIFICATION-BASED SOLUTION

Apart from dividing the swamp, sub-pieces also provide enough evidence to proof if any sub piece has been modified. As mostly no more than one malicious user poisoning the swamp, which implies most legitimate peers having the correct sub pieces.

Sub-pieces-verification-based solution is designed by this hypothesis and it mainly tracks the malicious users by the source the peers received. Before grouping all sub pieces together, Sub-pieces-verification-base solution verify in sub piece level as the checksum is also generated in sub piece level.

Whenever any sub piece found unmatched with the original hush, leech then temporarily memorize the source peer and ask other peers for the missing sub pieces.

If the sub piece from other peer can pass the checksum, then the leech will update the suspicious remark of the previous source peer to other leeches. If it still cannot pass the checksum, the leech then redo the mentioned steps until successfully obtain the correct sub piece.

A. Example of Implementing Sub-pieces-verification-based solution

Fig.6. shows the steps implementing this solution. For example, firstly leech B collects sub pieces from other peers A, C, & D, at the same time the leech also maintain a peer-sub piece table about whom the sub piece has sent to the leech. If sub pieces 3 & 4 from malicious user D cannot match with the checksum, leech B then discard the corrupted sub pieces and request other peers A & C for sub pieces 3 & 4.

After leech has received the sub pieces from A & C, then leech B updates other peers that malicious user D contains error sub pieces 3 & 4. Whenever other leeches joining the swamp, they will be notified that peer D has the error sub pieces. As a result the possibility of requesting peer D for sub pieces 3 & 4 will be lower than other peers' even it holds all sub pieces.

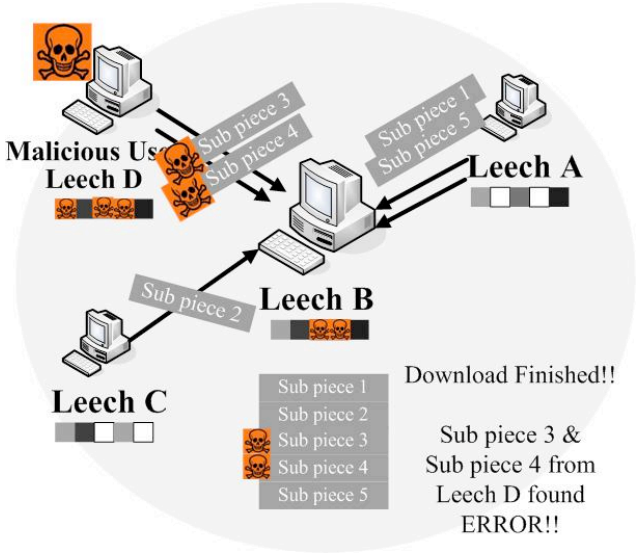

Fig.6. Illustration sub-pieces-verification-based solution

If still sub pieces from peers C & D cannot match with the checksum, then leech B regard the case as connection problem and request other peers for the missing sub pieces at later time.

B. Overhead Calculation of the Solution

Table 2 is the calculation of the overhead prediction about implementing the Sub-Pieces- Verification-Based solution. It is the prediction about the overhead if this solution being implemented at the existing BitTorrent environment. It shows that the overhead mainly depends on the number of normal users. As this solution mainly aim at resolving the case if and only if one malicious user within the swamp, thus the overhead of implement this solution will not bring an impact to the swamp.

Table 2. Overhead calculation on Sub-pieces-verification-based solution

Parameters (No. of malicious users, No. of normal peers)	**Traffic Overhead** (Max. no. of round trips, Min. no. of round trip)	**Bandwidth Overhead** Network bandwidth used (Kbytes) assuming each piece = 256 kb
(1,1)	(2,1)	Max=16*3*2=96 Min =16*3=48
(1,3)	(4,1)	Max=16*3*4=192 Min=16*3=48
(1,7)	(8,1)	Max=16*3*8=384 Min=16*3=48

V. Discussion and Conclusion

The comparison on both advantages and disadvantages between two proposed solutions is shown in Table 3.

As peer-selection-based solution aims at resolving the case with more than one malicious users, it brings an obvious overhead to the swamp since the size of the swamp has been limited as a result the performance of the swamp drops if numerous of the malicious users attack at the same time.

Sub-piece-verification-based solution can perform an outstanding efficiency to the system, however its protection can only be achieved if and only if one malicious user within the swamp.

As version checking and peer blacklisting cannot effectively resolve the attack, sub piece level checking is suggested into to reduce the impact of swamp poisoning. This can both minimize the impact of the poisoning attack also it can guarantee the integrity among sub pieces. Proposed solutions discussed above are suggested solutions based on sub-pieces protecting the concept to against swamp poisoning.

Reference

[1] A. Parker, "The true picture of peer-to-peer file-sharing," Proc. IEEE 10th Intl. Workshop on Web Content Caching and Distribution, Sophia Antipolis, France, 2005.
[2] B. Cohen, "Incentives build robustness in BitTorren" Proc. of the First Workshop on Economics of Peer-to-Peer Systems, 2003
[3] Publications: Bittorrent Protocol Specification v1.0 (2006, September) http://wiki.theory.org/BitTorrentSpecification
[4] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest First and Choke Algorithms Are Enough," Proc. 6th ACM SIGCOMM conference on Internet measurement, pp.203-216, 2006.

Table 3. The strengths and weaknesses of treatment-based solutions

	Advantages	Disadvantages
Peer-selection-based solutions	1. The malicious users may be found. 2. It can be achieved by modifying the application.	1. The efficiency will reduced since all sub-pieces need to be downloaded from fewer peers only. 2. Track suffers from heavy workload at busty traffic
Sub-Piece-verification- based solutions	1. The efficiency will higher than Peer-selection-based solutions since the new arrived sub-pieces will be checked immediately. 2. It can be achieved by modifying the application.	1. It doesn't work for more than one malicious users 2. The size of the torrent file will be greatly increased. 3. The BitTorrent protocol need to be modified