On Combating Content Poisoning in Peer-to-Peer Networks

Mohammed Hawa, *Member, IAENG,* Raed Al-Zubi, Khalid A. Darabkh, and Ghazi Al-Sukkar

***Abstract*—Poisoning attacks on Peer-to-Peer (P2P) networks are common nowadays. Poisoning refers to the condition in which malicious peers share corrupt or infected content in an attempt to destabilize the system and waste network bandwidth. By so doing, regular peers find less value in the P2P network and get discouraged from participating in the system. This problem can be thwarted by introducing mechanisms to verify the integrity of the downloaded content. However, this requires extra resources for such verification process. In this paper, we propose a strategy to minimize the threat of content poisoning, while requiring less verification overhead on the peers participating in the network.**

***Index Terms*—Peer-to-Peer, content poisoning, probabilistic, verification.**

I. Introduction

Peer-to-Peer (P2P) networks have emerged as a popular way to share content across the Internet [1]. Both the number of participating peers in P2P networks and the traffic volume they exchange rose to quite significant levels. The distributed nature of such networks and the lack of a centralized authority were among the main reasons why P2P networks were such a success. However, those same reasons made such networks vulnerable to various malicious attacks, such as content poisoning, free-riding, collusion, Sybil and denial-of-service attacks [2], [3], [4], [5]. In content poisoning, a *malicious* peer injects corrupt or infected content into the P2P network. This might be a video file with missing or white noise content, a corrupt or mislabeled archive file, or an executable file infected with viruses [5], [6], [7], [8].

Several measurement studies indicate that P2P systems are plagued by content poisoning. In [6], the authors studied content poisoning in the FastTrack network, and found that for some popular songs, more than 50% of the shared song versions were poisoned. A similar study investigated the poisoning in both the FastTrack and the Overnet P2P networks, and showed that both were affected by poisoning [7]. A more recent study [8] analyzed a large number of popular files on the KAD Kademlia-based distributed hash table (DHT) network implemented in the eMule client [9], and showed that 66% of its content is poisoned.

Poisoned content degrades the quality of data available on the P2P content sharing network, and deters peers from participating in the system in the future. Without proper countermeasures, *regular* peers might also end up as unwilling participants in distributing such poisoned content. This happens when an unsuspecting peer downloads poisoned content from a malicious peer, and then shares such content with the rest of the P2P system, thus infecting more unsuspecting peers.

In this paper, we propose a probabilistic approach to address content poisoning, in which we balance the trade-offs between quickly identifying poisoned content and thus eliminating the risk of this attack, and the overhead of such undertaking on various peers in the system. We argue that balancing those two issues will provide a more practical solution for real-life P2P systems.

This paper is organized as follows: In Section II we cite some related work. In Section III we elaborate on the verification overhead required to address the problem of content poisoning, and in Section IV we describe our approach to achieve the best possible balance between verification overhead and poisoning-restraint. The simulation setup and evaluation results are explained in Section V, and we conclude our work in Section VI.

II. Related Work

Two forms of poisoning exist in real-life P2P networks [8]: the first form, called content poisoning, consists of sharing files whose content is deliberately damaged or infected. The second form of poisoning, called index poisoning or meta-data poisoning, consists of corrupting the file indices used by the P2P system to advertise fake files which are actually not shared by any peer.

Several researchers have proposed solutions to the problem of content and index poisoning, especially in DHT-based overlay networks. For example, a scheme of maintaining a centralized database of blacklisted peers was proposed in [7]. In addition, various reputation systems, such as peer reputation [10], object reputation [11], and hybrid reputation [12] were also suggested to fight against content poisoning by introducing the concept of distributed voting, which adds an extra overhead on peers as they need to collect votes and process them. Such reputation systems are also vulnerable to malicious votes and other similar attacks.

Other suggested methods to combat poisoning include using a digital signature scheme applied to shared content [13], similar to what is used in various Internet applications [14]. However, this solution requires peers to access a centralized trusted authority to verify the digital signatures for every download, which consumes intensive resources from the centralized authority, it also compromises privacy, and introduces a single point of failure.

Another similar alternative is using a distributed or centralized service to perform the verification on behalf of the peer. This idea is similar to submitting an executable file to the http://virustotal.com/ service to check if the file contains

Manuscript received March 8, 2013. This work was supported in part by the Deanship of Academic Research at The University of Jordan.

M. Hawa, R. Al-Zubi, and G. Al-Sukkar are with the Electrical Engineering Department, The University of Jordan, Amman, 11942, Jordan, e-mail: {hawa, r.alzubi, ghazi.alsukkar}@ju.edu.jo,

K.A. Darabkh is with the Computer Engineering Department, The University of Jordan, Amman, 11942, Jordan, e-mail: k.darabkeh@ju.edu.jo.

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

a virus. Although this provides a more reliable verification compared to running a local copy of an anti-virus software (since such services test the file using multiple methods of analysis), it again consumes intensive resources from the third party, introduces a single point of failure, and adds to the resource consumption of the network bandwidth.

None of the proposed solutions has actually been widely deployed so far because of the associated overhead required for such solutions. To the best of our knowledge, no previous work looked at the issue of minimizing the required overhead in combating poisoned content to make the system more practical.

III. Content Verification

The various proposed solutions to the problem of content poisoning allow peers to identify and eliminate poisoned content. This, however, requires performing an integrity verification on the requested content before (or after) downloading such content. Whatever the method of verification we adopt, for this integrity verification to be reliable (i.e., accurate and with minimal false-positives), it will require sizable processing overhead on the peer's part.

Hence, we ask the following question: does every peer really need to invoke this expensive verification process every single time the peer downloads a file? Is there a way to reduce such overhead, while still minimizing the outbreak of poisoned content across the network. Since we do not live in an ideal world, and since solutions that require extreme overhead are difficult to implement and even more difficult to scale well in real-life practical systems, we propose a *probabilistic* technique for reducing this verification overhead while still minimizing any possible outbreak of poisoned content.

In addition, our algorithm does not disturb the decentralized nature of the P2P system, as each participating peer manages the potential risks involved in the content downloading and sharing by himself without communicating with other peers in the network. This is done based on the experience and knowledge acquired by that peer during his participation in the P2P network.

The choice of the exact method used for verification is irrelevant to this work. We just assume that the cost of verification is expensive, and we attempt to minimize this overhead. We will quantify the performance tradeoffs introduced by our proposed technique using simulation.

IV. Poisoning Restraint

Peers in a P2P network share the files they have created, and the files they download from others. We assume that each peer downloading a file has the ability (if it so desires) to perform integrity verification on the content of that file before (or after) the download process. If the peer decides to perform this integrity check and finds the file to be infected, it immediately deletes that file from its shared folder. On the other hand, if it finds the file to be genuine, or if it decided to skip the verification step, it shares the file with the rest of the network.

We propose two techniques to restraint poisoning, while limiting the number of content verifications done by each peer. In both techniques the decision to perform integrity verification or not is done on a random basis using a predetermined *check probability* P_c. We adopt this choice in order to reduce the number of verifications at various peers in the network. Moreover, each peer in our framework maintains a list of the peers from which it has downloaded content in the past, and maintains a value P_c^i for each such peer i. This probability represents the probability of verifying files downloaded from peer i in the future. Maintaining a list of peers with which the downloader has interacted is done regularly in the popular ED2K P2P network [9], and does not require excessive local storage resources in real-life P2P systems.

```
1   initialize P_c^i to 0.5 for all peers i
2   for file F from peer i
3    with probability P_c^i verify file F
4    {
5     if F is infected
6      Delete F
7     else
8      Keep and share F
9    }
10  otherwise
11   Keep and share F
```

Fig. 1. Pseudo-code for the probabilistic verification algorithm.

We present two overhead-minimizing poisoning-restraint algorithms: In the first one, the value of P_c^i is fixed for all peers in the system, and it set to 0.5. We call this the *probabilistic verification* method (see pseudo-code of Fig. 1). In the second algorithm, the probability P_c^i changes dynamically as the content of files received from peer i are investigated. We call this the *dynamic verification* method.

The idea behind dynamic verification is the following: if the downloading peer starts noticing that peer i is sending infected files, it will increase P_c^i to always verify files from peer i, and hence avoid being infected by its shared content. On the other hand, if the downloading peer starts noticing that peer j is sending genuine files most of the time, it will decrease P_c^j to reduce its verification cost. This is essentially an attempt to identify poisoners versus regular peers in the network based on interactions between the different peers, and hence has the potential to achieve a better tradeoff between the need for verification versus limiting the spread of infected files. The dynamic verification method is illustrated in the pseudo-code of Fig. 2. Notice that we maintain $P_c^i \in [0.1,\ 1.0]$. The reason we avoid dropping P_c^i to 0.0 is to fend against the practical possibility of regular peers turning to poisoners midways through the lifespan of the peer.

V. Evaluation

In this section, we first describe the simulation setup used to test our poisoning-restraint algorithms. We then present our key performance metrics, and discuss the simulation scenarios. Finally, we study the performance of the poisoning-restraint strategies we proposed earlier.

A. Simulation Setup

In the following simulations, we assume a P2P network composed of 2000 peers. A total of 20% of such peers are

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

```
1   initialize P_c^i to 1.0 for all peers i
2   for file F from peer i
3    with probability P_c^i verify file F
4    {
5     if F is infected
6       P_c^i ← 1.0
7       Delete F
8    else
9       P_c^i ← max(P_c^i ÷ 2, 0.1)
10      Keep and share F
11   }
12  otherwise
13   Keep and share F
```

Fig. 2. Pseudo-code for the dynamic verification algorithm.

poisoners, while the remaining 80% are regular peers. Other poisoner ratios, such as 5% and 10% of the total population have been tested, and we found that the results are quite similar. For brevity, we do not report on them here.

At the start of the simulation, each peer in the system creates 10 files and shares them. The 10 files created by each regular peer are genuine files, while the 10 files created by each poisoner peer are infected files. Poisoners are not interested in downloading content from the P2P network. On the other hand, regular peers have good faith, and do not know which files are good or bad. Hence, such regular peers are interested in all files in the network, and they download such files at random instants of time.

Once a regular peer downloads a file, it shares such file with the rest of the P2P network, unless it decided to verify the content of this file and discovers that it is infected, in which case the file is deleted and never downloaded again.

This downloading process continues until all regular peers have download all the files in the network. To normalize our simulation setup we break the time necessary to finish downloading all files into 20 cycles.

B. Performance Metrics

We seek to reach a condition in which we minimize the number of carried out file verifications while also reducing the infection level expected in the P2P network. Hence, we evaluate two performance measures: the first is the *fraction* of all files downloaded that are *verified*, whether kept or found to be infected and deleted. Higher values represent larger verification cost, resulting in a waste of resources at the various peers.

The second quantity we evaluate is the *fraction* of all downloaded files that are *infected* but never detected, and hence shared, because of lack of verification. Larger infection values are undesirable because they destabilize the P2P system and waste the network bandwidth.

C. Simulation Scenarios

To evaluate the performance of our proposed poisoning-restraint methods, we simulate the following four scenarios:

- **Full verification:** In this simple case, we set $P_c^i = 1.0$ for all peers in the simulation. This means that peers will always verify the files they download from others. We expect maximum verification overhead, but also zero infections.
- **No verification:** Here, we set $P_c^i = 0.0$ for all peers. This means that peers will never verify the files they download from others. We should expect the maximum possible infection level, but zero verification overhead.
- **Probabilistic verification:** In this case, we set $P_c^i = 0.5$ for all peers. This means that after downloading a file, the peer will have a fifty-fifty chance of verifying the file, which reduces the verification overhead, but allows infections in the network. This is our proposed probabilistic verification method (see Fig. 1).
- **Dynamic verification:** Here, we use the more elaborate dynamically changing P_c^i values based on the pseudo-code of Fig. 2. This is our dynamic verification method.

It is obvious that the first two scenarios represent limiting cases, and hence, will help us provide upper and lower bounds on the possible performance of poisoning-restraint algorithms. They will also represent a convenient way of validating our simulation code.

The last two scenarios represent the two poisoning-restraint mechanisms proposed in the previous section. We wish to quantify the performance of those proposed methods.

D. Results

Fig. 3 shows the fraction of verifications for the four tested scenarios versus simulation time, and Fig. 4 shows the fraction of infections that occurred in the P2P system versus time. These fractions are calculated based on the total number of downloaded files.

The results for the full verification and no verification scenarios match what we expect, and hence validate our simulations: When 100% of the files are verified by the downloading peers, 0% infections occur in the P2P system. On the other hand, when trying to eliminate the cost of verifying files, the infection rate reaches the maximum possible value, which is equal to the fraction of injected poisoned content (20% in this experiment). This is exactly the same ratio of poisoners in the P2P system.

Our proposed algorithms attempt to reduce the verification overhead by as much as possible while still maintaining a small infection rate in the P2P network. We observe that this is somewhat possible using the probabilistic verification method, but is perfectly achieved by the dynamic verification technique. In the probabilistic approach, we notice that the verification overhead dropped from 100% to 50% (compared to full verification), while the infection rate has dropped from 20% to 10% (compared to no verification). This represents a simple, but reasonable tradeoff between verification cost and infection breakout.

However, the highlight of the performance results come from the dynamic verification algorithm. In such case, the verification overhead is reduced to 45.9% by the end of the simulation. This is achieved while the infection level is kept to 0%. This is because all downloaders in the dynamic verification algorithm start by verifying all files they receive (i.e., $P_c^i = 1.0$), but they gradually drop the probability P_c^i for the remoter peers that have good reputation of delivering genuine files. It is worth mentioning that the verification overhead can be brought down even further if we drop the

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

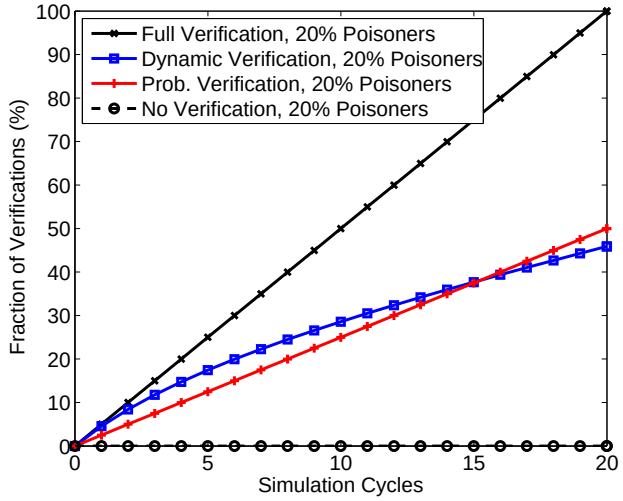

Fig. 3. Fraction of verifications in the case of 20% poisoners sharing infected files.

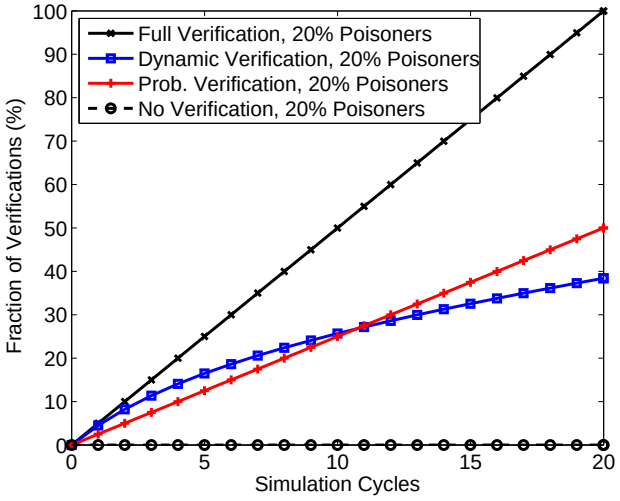

Fig. 5. Fraction of verifications in the case of 20% poisoners sharing both genuine and infected files.

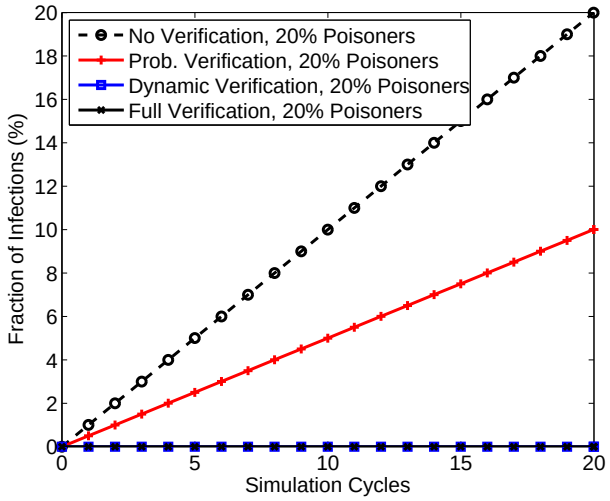

Fig. 4. Fraction of infections in the case of 20% poisoners sharing infected files.

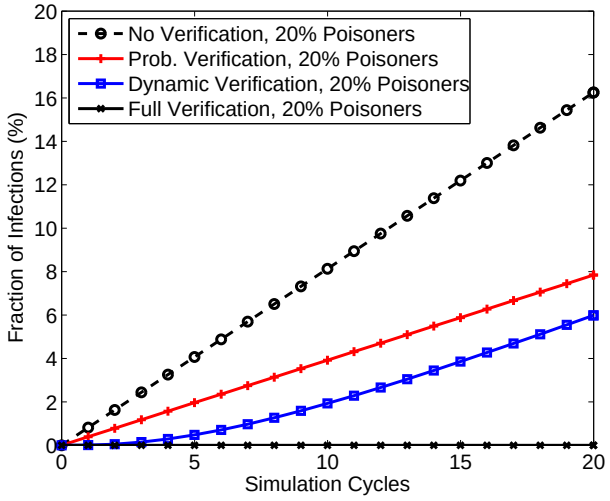

Fig. 6. Fraction of infections in the case of 20% poisoners sharing both genuine and infected files.

value of P_c^i quicker (by dividing by, say, 4.0), and/or if we lower minimum value of P_c^i from 0.1 to 0.0.

However, the reason we do not do this is that a poisoner peer m can start by sending genuine content to reduce its P_c^m to 0.0 at other peers in the system, and then starts sending infected files. We have to admit, though, that this is a remote possibility. For starters, in real-life P2P systems the main concern of poisoners is to destabilize the P2P network with the *minimum* possible effort. Therefore, poisoners who are willing to send genuine content to other peers (and hence dedicate the resources to do such thing) are extremely rare. We see this often in the real-life BitTorrent P2P file sharing network, for example [15]. In addition, since each peer in the network has its own P_c^m for the poisoner peer m, the poisoner has to work extremely hard to deliver genuine content to so many peers to gain good reputation, which again requires extensive resources in a large P2P system.

With that being said, we still would like to quantify how our proposed algorithms work in the case where poisoners send some genuine content along with infected files. Therefore, we repeat our simulation experiments with the same parameters described earlier, except that this time, poisoner peers create and share 2 genuine files and 8 infected files (rather than 10 all infected files). The results of the new simulation runs are shown in Figs. 5 and 6.

Notice that even when the poisoners are acting as *rogue* peers in the system, our dynamic verification method performed quite well. The verification overhead is only 38.4%, and the infection rate remained at 6%, which is less than both the no verification case and the probabilistic case. It is worth mentioning that the dynamic algorithm is quite flexible in its design, since the value of P_c^i can be made to drop slowly to prevent infection (albeit on a slight increase in the cost of verification), or can be made to drop quicker to reduce the verification overhead.

VI. Conclusions

The reason that P2P systems suffer extensively from poisoning attacks is the amount of resources needed to implement integrity verification to combat such attacks. Unless this overhead can be reduced, poisoning-restraint methods cannot find their way into practical P2P networks.

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

In this work, we introduced two methods that balance poisoning-restraint and verification overhead. The first is the probabilistic verification method, which is extremely simple, and does not require any processing or storage of the P_c^i values. It results in reasonable, but not excellent, performance.

The second method, on the other hand, is the dynamic verification method, which adaptively changes the P_c^i values depending on the observed behavior of the peers in the system. This method provides the best balance in performance against poisoners. It is also resistant to poisoners who are willing to put more effort into fooling the P2P system into believing they are regular peers.

REFERENCES

[1] C.-L. Hu and Z.-X. Lu, "Downloading trace study for BitTorrent P2P performance measurement and analysis," *Peer-to-Peer Networking and Applications*, Volume 5, Issue 4, 2012, pp. 384–397.

[2] P.M.R. Anand and V. Bhaskar, "Polluted content prevention in Peer-to-Peer file sharing networks." In: *Proceedings of the Annual IEEE Conference on Engineering Sustainable Solutions*, 2011, pp. 1–4.

[3] C. Selvaraj and S. Anand, "A survey on security issues of reputation management systems for peer-to-peer networks." *Computer Science Review*, Volume 6, Issue 4, 2012, pp. 145–160.

[4] N. Christin, A.S. Weigend, and J. Chuang, "Content availability, pollution and poisoning in file sharing Peer-to-Peer networks." In: *Proceedings of the 6th ACM Conference on Electronic Commerce*, 2005, pp. 68–77.

[5] B. S. Sarjaz and M. Abbaspour, "Securing BitTorrent using a new reputation-based trust management system." *Peer-to-Peer Networking and Applications*, Volume 6, Issue 1, 2013, pp. 86-100.

[6] J. Liang, R. Kumar, Y. Xi, and K.W. Ross, "Pollution in P2P file sharing systems." In: *Proceedings of the 24th IEEE International Conference on Computer Communications (INFOCOM)*, 2005, pp. 1174–1185.

[7] J. Liang, N. Naoumov, and K.W. Ross, "The index poisoning attack in P2P file sharing systems." In: *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, 2006, pp. 1–12.

[8] G. Montassier, T. Cholez, G. Doyen, R. Khatoun, I. Chrisment, and O. Festor, "Content pollution quantification in large P2P networks: a measurement study on KAD." In: *Proceedings of the 11th IEEE International Conference on Peer-to-Peer Computing*, 2011, pp. 30-33.

[9] eMule official website and source code, http://www.emule-project.net.

[10] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks." In: *Proceedings of the 12th international world wide web conference*, 2003, pp. 640-651.

[11] K. Walsh, and E.G. Sirer, "Experience with an object reputation system for Peer-to-Peer file sharing." *In: Proceedings of the 3rd conference on Networked Systems Design and Implementation*, Volume 3, 2006, pp. 1-1.

[12] C. Costa and J. Almeida. "Reputation systems for fighting pollution in peer-to-peer file sharing systems," *In: Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing*, 2007, pp. 53–60.

[13] C. Cid, "Recent developments in cryptographic hash functions: Security implications and future directions." *Information Security Technical Report*, Volume 11, Issue 2, 2006, pp. 100-107.

[14] Mobione, "iOS Application Provisioning Requirements." http://www.genuitec.com/mobile/docs/appledigitalsignature/appledigitalsignature.html, 2013. Retrieved March 2013.

[15] Ernesto, "Anti-Piracy Outfits Launch Attack on BitTorrent Protocol." http://torrentfreak.com/anti-piracy-outfits-launch-attack-on-bittorrent-protocol-120519/, May 2012. Retrieved March 2013.

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)