Analyzing Impacts of Software Diversity on Worm Propagation in Peer-to-Peer Networks

Ying Zhou, Zhong-fu Wu, Feng Li, Hao Wang and Zheng-zhou Zhu

College of Computer Science and Technology, Chongqing University Campus A, Chongqing, China

Summary

As Peer-to-peer (P2P) networking technologies become popular, P2P worms which exploit common vulnerabilities of P2P software to achieve fast worm propagation have emerged. It is believed that software diversity can decrease the virulence of worms and the effectiveness of repeated applications of single attacks. In this paper, we introduce software diversity into the prevention of worm propagation in P2P networks. In particular, we propose a P2P-based software diversity model and conduct both theoretical analysis and extensive simulations to test the performance of software diversity in containing P2P worm propagation. The results indicate that our work can provide a new way to mitigate threats of P2P worms.

Key words:

Peer-to-Peer Network, Peer-to-Peer Worm, Software Diversity, Worm Propagation.

Introduction

Peer-to-peer (P2P) overlay networks begin with Napster, Gnutella, and several other related systems, and have become immensely popular in the past few years. Unfortunately, it's believed that P2P systems can be a potential vehicle for the worm to achieve fast propagation. Kazaa P2P network worms, such as Kitro, Lolol, Benjamin, and Roron, have been widespread. P2P networks provide an ideal venue for new types of worms that prey on common vulnerabilities on the peers in a P2P network. These worms identify new victims simply by following P2P neighbor information on infected hosts. They are different from the currently popular random scanning worms and some researchers consider that P2P worm is a kind of topology scanning worm which can spread much faster over P2P networks, since it do not waste time probing unused IP addresses. Furthermore, P2P worms do not generate high rates of failed connections and they can blend into the normal traffic patterns of the P2P network. The lack of abnormal network behavior makes P2P worms a potentially more deadly threat because most existing defense mechanisms against scanning worms are no longer effective.

Some people have done research on the security of P2P networks, such as secure routing [1] and DOS (denial of service) attacks in the Gnutella P2P system [2]. Simultaneously, Much work has been done in worm modeling, analyzing and containing [3], [4], [5]. Several attack techniques for effective worm propagation over different types of network systems including P2P systems are discussed in [6]. Recently, much detailed modeling and analysis on the attack propagation over P2P systems is presented. Yu et al. [7] defined a P2P system based active worm attack model and studied two attack strategies (an off-line and on-line strategy) under the defined model. They observed that a P2P-based attack can significantly worsen attack effects (improve the attack performance). However, they didn't consider the method to prevent worm propagation in P2P networks. Traditional approaches dealing with worm propagation are not suitable for P2P worms.

In this paper, we attempt to introduce software diversity into the prevention of worm propagation in P2P networks. Real data shows that the interval is decreasing between flaw discovery and flaw exploitation. The propagation rate between susceptible hosts regularly sets new speed records. We have to confess that Microsoft's monopoly on the desktop (monoculture of operating system) is answerable for the problems, because it draws a near-monopoly of the attacks, in both type and number of attacks. The aftermath of monoculture i s cascade failure which indulges the worm propagation in networks. The real problem, which is without doubt growing more serious, is that of identical platforms that are riddled with security holes—the same security holes. The distribution of P2P software should not follow the same old disastrous road as that of operating system. Therefore, a desirable method to prevent P2P worm propagation is achieving software diversity in P2P networks. According to the features of worm propagation over P2P networks, we present a P2P software diversity model for worm prevention on P2P networks in this paper. Based on the above P2P model, we study the effectiveness of software diversity on preventing P2P worm propagation. The results of experiments indicate that our work can provide a new way to mitigate threat of P2P worms.

The rest of paper is organized as follows: In Section 2, we present some background related to P2P worms and software diversity. In Section 3, the worm propagation over P2P networks is presented. Based on that, a P2P software diversity model and analysis are specified in Section 4. Finally, conclusion of this paper and future work are given in Section 5.

2. Background

A P2P networked system is a group of Internet nodes that construct their own special-purpose networks on top of the Internet. Such a system performs application level routing on top of IP routing. Researchers have defined structured P2P overlays such as CAN [8], Chord [9], Pastry [10] and Tapestry [11] which give a self-organizing substrate for large-scale P2P applications. Rather than being designed specifically for the purpose of file sharing, these systems serves as a powerful platform for the construction of a variety of decentralized services, including network storage, content distribution, web caching, searching and indexing, and application-level multicast.

Typical worm propagation can be described by epidemic models, such as Simple Epidemic Model [12], Kermack-Mckendrick Model [13], SIS (Susceptible Infectious Susceptible) Model [14], Two-Factor Model [15]. Although these models analyze worm's behavior from different points of view, at least one consentaneous conclusion can be drawn—in the initial phase of worm propagation, exponential increase in infections is observed. Unfortunately, most impact and losses are caused by worms in this period of propagation. Traditional ways based on firewalls, IDS, honey pots and so on, are limited to one host or a small group of hosts, while worm propagation can cover a wide range of networks in short time.

Some researchers assert that security can only be achieved in a real network if a multitude of software packages is utilized. Work has been done to introduce diversity at the system level through a variety of techniques, including both source [16] and instruction set [17] randomization. And some methods artificially introduce diversity to software by virtue of source code modification [18], virtual hardware modification, and compile time randomization techniques. Researchers have begun to examine the problem of distributing diversity from a network-aware perspective that would decrease the rate at which an attacker can progress across the network. To the best of our knowledge, there has been no research devoted exclusively to studying the effect of software diversity on the prevention of P2P worm propagation.

3. Worm propagation over P2P network

P2P network is an overlay network which is composed of numerous peers that are distributed over the whole Internet. The structure of it is elaborately designed for the purpose of network resource sharing. Obviously, worm can use P2P systems as a vehicle for fast propagation as shown in Fig.1.

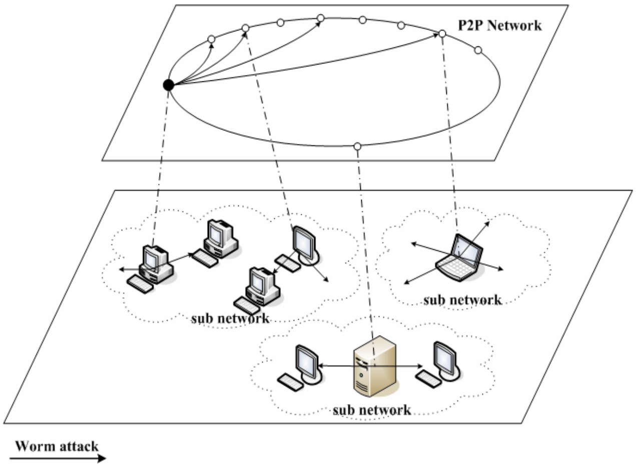

Fig. 1 P2P worm propagation.

As we can see from Fig.1, if a real host belongs to a P2P network, it can attack both its P2P neighbors and hosts in the sub network. It's clear that P2P network accelerates worm propagation over the whole Internet. By containing worm attacks on P2P network, we can achieve our goal of preventing P2P worm propagation. Hence, for simplification purposes, we do not consider the attacks outside the P2P network.

In order to formally analyze P2P worm propagation, we represent the topology of the P2P network by an undirected graph $G=<V,E>$. $\forall v \in V$, v denotes a peer in the P2P network. $\forall e \in E$, e represents the connection between two peers. Peer degree d_i and scan rate r_i are two of the most important parameters for the worm attack. This is a naive P2P worm propagation model. On the based of this model, we present our P2P software diversity model and compare the simulation results to analyze the effect of our method on preventing P2P worm propagation in Section 4. Nevertheless, we believe that our proposed models can be easily extended to more complicated P2P network models. Table I lists all parameters and notations in this simulation.

TABLE 1: Notations used in this simulation

Symbol	*Explanation*
G	Undirected graph representing the P2P network, $G=<V,E>$
V	The total peers in the P2P network
E	The total connections in the P2P network
t	Time in the simulation
d_i	Topology degree of peer i
s_i	Scan rate of peer i
I_t	Infected peers in time t
r_i	Infection ratio
k	the number of software packages
p	Threshold parameter of distributed algorithm

There are various types of P2P systems. Structured P2P systems boast an efficient lookup mechanism by means of DHTs (Distributed Hash Tables). In the structured P2P system, such as CAN, Chord, Pastry and Tapestry, all P2P nodes maintain the same topology degree, which defines the number of neighbors for each P2P node. Contrarily, unstructured P2P systems use mostly broadcast search, like Freenet and Gnutella systems. In this system, the topology degree is a variable for each P2P node. In order to conduct a comprehensive simulation, we run the worm simulation on a random graph network, a small world network, and a power law network.

In this paper we use the power law generator in [19] to generate power law topologies. The node degree of a "power law topology" is heavy-tailed distributed and has the power law ccdf $F(d) \propto d^{-\alpha}$, which is linear on a log-log plot [19]. Except power law topology generators, there is no other network generator available to create a heavy-tailed distributed topology. There are other popular topologies such as random graph topology and small world topology [20]. We study worm propagation on these topologies as well. We generate the small world topology by using the two-dimensional small world model presented in [21].

Fig.2 shows the worm propagation in different P2P topologies. Although the worm attack efficiency is different, we can see that all the worm propagation is fast and most of the peers are infected at last.

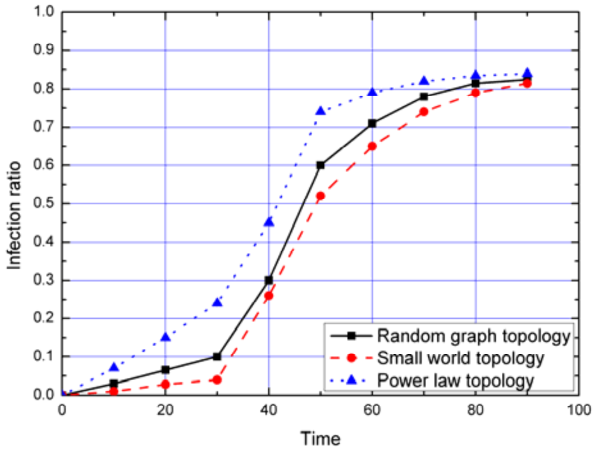

Fig. 2 worm propagation in different P2P topologies.

4. P2P software diversity model and analysis

In this section, we present the P2P software diversity model and analyze the performance of our work on P2P worm prevention. In this model, we provide algorithms which assign software packages to nodes on the P2P network in order to limit the total number of peers an attacker can compromise using a limited attack toolkit. The primary optimization goal would be to reduce the number of neighboring peers running the same software package on the network.

4.1 Introduction

As stated previously, we represent a P2P network using a graph $G=<V,E>$. The number of nodes and edges in the network are denoted by $|V|$ and $|E|$, respectively. The number of neighbors of any given peer is d_i. The set of software packages is denoted by S, and the number of software packages is denoted by k. We need to devise an assignment of software packages, such that the ability of the attacker to compromise the entire network is significantly reduced.

The assignment of k software packages to the graph G is what graph theoreticians would call a coloring of graph G. The assignment of colors in such a way that the number of defective edges, or communication links that exist between two peers of the same color, is minimized is called an optimum coloring. A perfect coloring is an assignment of the minimum number of colors necessary to color a graph such that no two neighboring nodes share the same color. The minimum number of colors required for a perfect coloring is denoted by $\chi(G)$. When $k<\chi(G)$, any color assignment will induce at least one edge where both endpoints are similarly colored. A coloring where such an edge, referred to as a defective edge, is present is called a defective coloring. Determining a minimum number of colors required to achieve a perfect coloring is, in the general case, an NP-Hard problem. Aside from a handful of special cases, determining an optimum coloring with a minimum number of defective edges is also NP-Hard.

In this paper, we assume that there is a set of P2P software packages for us to deploy on the P2P networks. We do not focus on techniques to produce diverse P2P software.

4.2 Distributed algorithm

The first, and most basic, algorithm discussed is the Randomized Coloring algorithm. This provides, on average, $|E|/k$ defective edges. Proving this is a simple exercise: after randomly coloring every node on the graph, select a single edge. The probability that both endpoints have the same color is $1/k$. Summing across all edges, the average number of defective edges is $|E|/k$. The algorithm requires $O(1)$ time to run on each node, and zero communication between the nodes is required. Because of the lack of inter-node communication, the algorithm can be considered extremely secure against attack. The graph coloring provided by the algorithm, however, is sub-optimal. In the worst case, this algorithm performs poorly. A randomized algorithm may lead to every link forming a connection between two identical systems. While the probability of this event occurring is $(1/k)^{|V|-1}$, the result would have a significant impact on system security.

After executing the randomized coloring algorithm, we could optimize the algorithm as follows: After a random delay, each peer performs a local search amongst its immediate neighbors to determine if switching to a new color would decrease the number of locally defective edges. Since each peer must now poll its immediate neighbors to discover their current color, the algorithm requires $O(\Delta(G))$ time to poll the neighbors per cycle, where $\Delta(G)$ is the maximum degree of the graph. After the data is collected, $O(\Delta(G)+k)$ operations must be done to generate a census of the local colors and determine the minority color. Since the peer should check its neighbors and change the software frequently, the software structure of the whole P2P system is not stabile. To solve this problem, we use a threshold parameter p to control the optimizing algorithm. Until the largest count of the same software in a peer's neighbors exceeds p, the peer should not perform the optimizing algorithm.

By the time the optimizing algorithm has converged, total number of defective edges is provably decreased below the average number of defects in the randomized coloring algorithm. The proof is given as follows:

At the point of convergence, each node is connected to at most $\lfloor d_i/k \rfloor$ defective edges. The number of defective edge endpoints is $\sum_{i=1}^{|V|} \lfloor d_i/k \rfloor$. The number of defective edges is therefore $\frac{1}{2}\sum_{i=1}^{|V|} \lfloor d_i/k \rfloor$. In comparison to the randomized algorithm:

$$\frac{1}{2}\sum_{i=1}^{|V|} \lfloor d_i/k \rfloor \le \frac{1}{2}\sum_{i=1}^{|V|} d_i/k = \frac{|E|}{k}.$$

4.3 Simulation Results

In this section, we report the performance results of our proposed P2P software diversity model. To compare with the results in Section 3, we run our model on a random graph network, a small world network and a power law network, respectively. The corresponding simulation results are shown in Fig.3, Fig.4 and Fig.5.

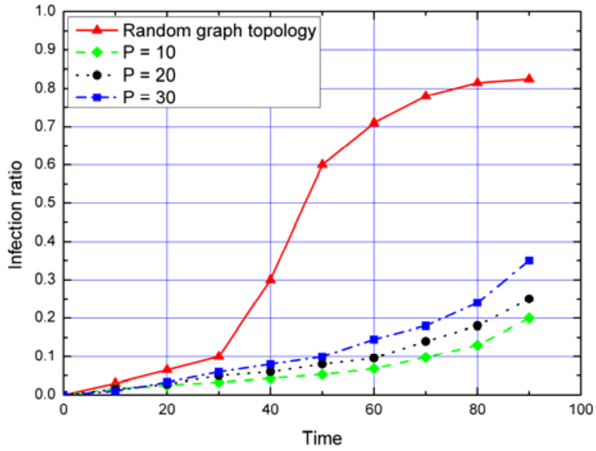

Fig. 3 Effect of distributed algorithm on random graph topology.

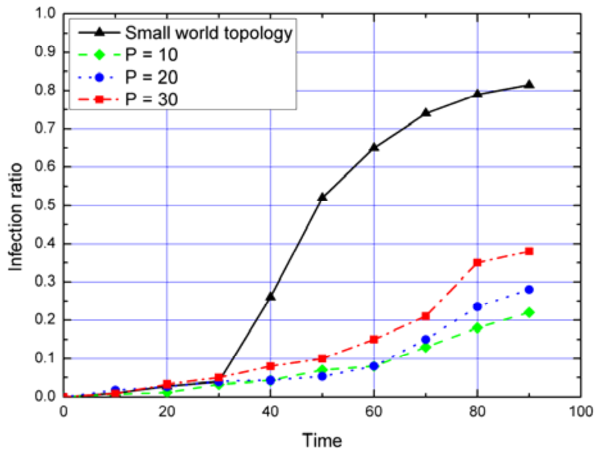

Fig. 4 Effect of distributed algorithm on small world topology.

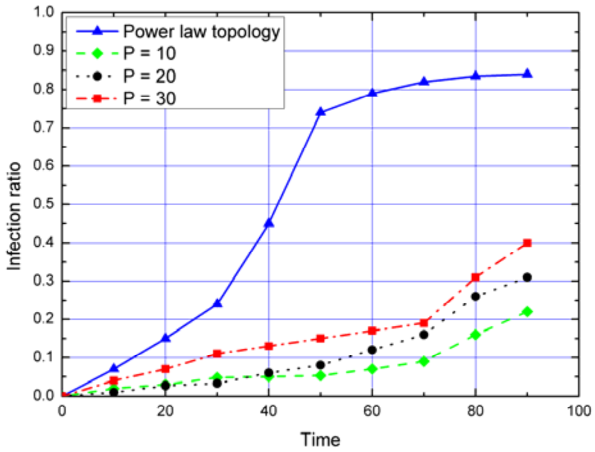

Fig. 5 Effect of distributed algorithm on power law topology.

We can draw two important conclusions from the figures:

1) Although these three topologies are quite diffe rent, the simulation results show that our P2P sof tware diversity model can prevent most of the pe ers from being attacked by P2P worms and the p erformance of the model is not sensitive to the to pology of P2P network.

2) Threshold parameter p of distributed algorith m is a crucial factor in the performance of the mo del. The simulation with lowest value of p achiev es the best performance compared to other simul ations. However, a low value of p would cause th e peers to change their software frequently. Ther efore, the suitable value of p rests with the scale and capability of P2P networks.

5. Conclusions

In the paper, we have presented a P2P software diversity model to study the effect of P2P softwa re diversity on P2P worm propagation. In Section 3, we investigate the worm propagation on P2P networks with different topologies, including rand om graph topology, small world topology and pow er law topology. Simulations of our proposed mod el are presented in Section 4. The results of the simulations demonstrate that our approach has a good performance in restraining the worms over P2P overlay network. To the best of our knowled ge, there has been no research devoted exclusiv ely to studying the effect of software diversity o n the prevention of P2P worm propagation.

In the future, we would like to extend the study o f P2P software diversity to the design of P2P sof tware that could break monoculture automatically.

Acknowledgment

This paper is supported by the National Natural S cience Foundation of China (Grant No. 30400446), the Research Fund for the Doctoral Program of Higher Education of China (Grant No. 200406110 2) and the 863 Program Project of China (Grant No. 2003AA116010).

References

[1] M. Castro, P. Druschel, etc. “Secure routing for structured peer-to-peer overlay networks”, Proceedings of the 5-th USENIX Symposium on Operating Systems Design and Implementation (OSDI), Boston, Massachusetts, December 2002.
[2] N. Daswani, H. G. Molina, “Query-Flood Dos Attack in Gnutella”, Proceedings of the 9th ACM conference on Computer and Communications Security, 2002.
[3] J. O. Kephard, S.R. White, “Directed-graph Epidemiological Models of Computer Virus”, Proceedings of 1991 Computer Society Symposium on Research in Security and Privacy, 1991.
[4] Z. S. Chen, L.X. Gao, K. Kwiat, “Modeling the Spread of Active Worms”, Proeedings of IEEE Infocom, 2003.
[5] C. C. Zou, W. B. Gong, D. Tonsley, “Code Red Worm Propagation Modeling and Analysis”, Proceedings of the 9th ACM conference on Computer and Communications Security, 2002.
[6] S. Staniford, “Containment of Scanning Worms in Enterprise Networks”, Journal of Computer Security, 2004.
[7] W. Yu, C. Boyer, D. Xuan, “Analyzing Impacts of Peer-to-Peer Systems on Propagation of Active Worm attacks”, Technical Report, Computer Science Dept., Texas A&M Univ, 2004.

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, " A Scalable Content Addressable Network", In Proceedings of ACM SigComm., San Diego, 2001.
[9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", In Proceedings of ACM SIGCOMM 2001, San Deigo, CA, August 2001.
[10] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems", In Proceeding of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, November, 2001
[11] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: A Resilient Global-scale Overlay for Service Deployment", IEEE Journal on Selected Areas in Communications, vol. 22, no. 1, pp. 41-53, January 2004.
[12] Streftaris G, Gibson GJ. Statistical inference for stochastic epidemic models. In: Proc. of the 17th Int'l Workshop on Statistical Modelling. Chania, 2002. 609~616.
[13] Frauenthal JC. Mathematical Modeling in Epidemiology. New York: Springer-Verlag, 1980.
[14] Wang Y, Wang CX. Modeling the effects of timing parameters on virus propagation. In: Staniford S, ed. Proc. of the ACM CCS Workshop on Rapid Malcode (WORM 2003). Washington, 2003.
[15] Zou CC, Gong W, Towsley D. Code Red worm propagation modeling and analysis. In: Proc. of the 9th ACM Symp. on Computer and Communication Security. Washington, 2002. 138~147.
[16] R. C. Linger. Systematic generation of stochastic diversity as an intrusion barrier in survivable systems software. In Proceedings of the 32nd Hawaii International Conference on System Sciences, volume 3, page 3062, 1999.
[17] Selvin George, David Evens, Steven Marchette. "A Biological Programming Model for Self-Healing", First ACM Workshop on Survivable and Self-Regenerative Systems (in association with 10th ACM Conference on Computer and Communications Security) October 31, 2003, George W. Johnson Center, George Mason University, Fairfax, VA
[18] C. Collberg, C. Thomborson, and D. Low. Breaking abstractions and unstructuring data structures. In Proc of the IEEE International Conference on Computer Languages, pages 28–38, Chicago, IL,May 1998.
[19] T. Bu and D. Towsley, "On distinguishing between internet power law topology generators," in Proceedings of the IEEE INFOCOM, June 2002.
[20] D. Watts and S. Strogatz. "Collective dynamic of small-world networks," Nature, vol. 393, 1998.
[21] M. Newman, I. Jensen, and R. Ziff, "Percolation and epidemics in a two-dimensional small world,"Phys. Rev. E., vol. 65, no. 021904, 2002.